Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme

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

genehmigte
DISSERTATION

von
Dipl.-Ing. Roman Dumitrescu
aus *Nürnberg*

Tag des Kolloquiums: 22. Dezember 2010
Referent: Prof. Dr.-Ing. Jürgen Gausemeier
Korreferent: Prof. Dr.-Ing. habil. Ansgar Trächtler

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Heinz Nixdorf Institut der Universität Paderborn. Sie ist das Ergebnis meiner Forschungstätigkeiten im Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) sowie an der International Graduate School „Dynamic Intelligent Systems“.

Mein herzlicher Dank gilt Herrn Prof. Dr.-Ing. Jürgen Gausemeier, der mich in meiner Zeit am Lehrstuhl für Produktentstehung stets forderte und förderte. Ich danke ihm besonders für das große Vertrauen in meine Arbeit, und dass er mir ein selbständiges und kreatives Forschen ermöglichte. Seine konstruktive Kritik hat meine Arbeit wesentlich geprägt.

Herrn Prof. Dr.-Ing. habil. Ansgar Trächtler vom Lehrstuhl Regelungstechnik und Mechatronik am Heinz Nixdorf Institut danke ich für die Übernahme des Korreferats.

Allen Kollegen am Lehrstuhl danke ich für die sehr gute Zusammenarbeit und das angenehme Arbeitsklima. Hervorzuheben sind Dipl.-Wirt.-Ing. Sascha Kahl, Dipl.-Wirt.-Ing. Jörg Donoth, Dipl.-Ing. Harald Anacker, M.Sc. Dipl.-Ing. (FH) Frank Bauer und M.Sc. Rafal Dorociak, die mich durch die intensive Auseinandersetzung mit meiner Dissertation unterstützt haben. Ferner danke ich Christian Bremer, Maren Borchers, Vitali Voth sowie allen weiteren Studierenden, die mich bei meiner Arbeit durch ihre Studien- und Diplomarbeiten oder durch ihre studentische Hilfstätigkeit unterstützt haben.

Den Forschern aus dem SFB 614 und insbesondere Dr. rer. pol. Benjamin Klöpper, Dipl.-Ing. Christoph Romaus, Dipl.-Ing. Peter Reinold, Dipl.-Math. Herbert Podlogar und Dipl.-Inform. Stefan Dziwok danke ich für die hervorragende interdisziplinäre Zusammenarbeit im Rahmen meiner Dissertation.

Meinen Eltern danke ich für ihre konsequente Unterstützung meiner Ausbildung. Meiner Schwester Patricia danke ich für die motivierenden Gespräche. Meiner Verlobten Nicole danke ich für ihr Verständnis und ihre Geduld, da sie in den letzten Jahren oft zurückstecken musste. Dir gilt mein größtes Dankeschön!

Paderborn, im Januar 2011 Roman Dumitrescu

Liste der veröffentlichten Teilergebnisse

[DAG10] DUMITRESCU, R.; ANACKER, H.; GAUSEMEIER, J.: Specification of Solution Patterns for the Conceptual Design of Advanced Mechatronic Systems. In: Proceedings of the 2010 International Conference on Advances in Mechanical Engineering (ICAME2010), December 2-5, Shah Alam, Malaysia, 2010

[DGD+09] DUMITRESCU, R.; GAUSEMEIER, J.; DANGELMAIER, W.; KLÖPPER, B.: Solution Patterns for the Development of Self-Optimizing Systems. In Klöpper, B.; Dangelmaier, W. (Eds.): Self-x in Engineering. MV-Verlag, Münster, 2009

[DGK10] DUMITRESCU, R; GAUSEMEIER, J; KAHL, S.: Tool-based approach for the development of self-optimizing systems with solution patterns. In: Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2010), August 15-18, Montreal, Canada, 2010

[DGR09] DUMITRESCU, R.; GAUSEMEIER, J; ROMAUS, C.: Towards the Design of Cognitive Functions in Self-Optimizing Systems Exemplified by a Hybrid Energy Storage System. In: Proceedings of the 10th International Workshop on Research and Education in Mechatronics (REM2009), September 10-11, Glasgow, UK, 2009

[DK09] DUMITRESCU, R.; KLÖPPER, B: Towards Social-Software for the Efficient Reuse of Solution Patterns for Self-Optimizing Systems. In: Proceedings of International Conference on Knowledge Engineering and Ontology Development (KEOD2009), October 6-8, Madeira, Portugal, 2009

[Dum09] DUMITRESCU, R.: Design Systematics for the Integration of Cognitive Functions in Intelligent Mechatronic Systems. In: Proceedings des gemeinsamen Workshops der Informatik-Graduiertenkollegs und Forschungskollegs, Dagstuhl, 2009

[Dum10] DUMITRESCU, R.: Developing Cognitive Functions in Self-Optimizing Systems with Solution Patterns. In: Proceedings des gemeinsamen Workshops der Informatik-Graduiertenkollegs und Forschungskollegs, Dagstuhl, 2010

[GDD+10] GAUSEMEIER, J; DONOTH, J.; DUMITRESCU, R.; TRÄCHTLER, A.; REINOLD, P.: Self-Optimization – An Approach for Intelligent Mechatronics Exemplified by an X-by-wire Vehicle. In: Proceedings of 8th IEEE International Conference on Industrial Informatics (INDIN2010), July 13-16, Osaka, Japan, 2010

[GDP08] GAUSEMEIER, J.; DUMITRESCU, R.; PODLOGAR, H.: Implementing Cognitive Functions with Active Pattern in Self-Optimizing Systems. In: Proceedings of the 3rd Asia International Symposium on Mechatronics (AISM2008), Hokkaido University, August 27-31, Sapporo, Japan, 2008

Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme

Inhaltsverzeichnis

Seite

Anhang

1 Einleitung

Diese Arbeit entstand im Rahmen des Sonderforschungsbereichs 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) der Universität Paderborn. Ziel des SFB sind Systeme, die ihr Verhalten selbstständig und bestmöglich auf das aktuelle Systemumfeld einstellen. Die vorliegende Arbeit ordnet sich in den Bereich der Entwurfsmethodik für derartige Systeme ein und beschreibt eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*.

1.1 Problematik

Die Rolle technischer Systeme im Leben der Menschen hat seit der industriellen Revolution stetig an Bedeutung gewonnen. Die Technik durchdringt Alltag und Wirtschaft derart, dass sie nicht mehr wegzudenken ist. Kommunikation, Produktion und Transport sind die wesentlichen Felder, in denen der Mensch auf technische Systeme immer mehr angewiesen sein wird. Selbstredend verändern sich aber auch die technischen Systeme. Neue Einsatzbereiche bringen neue Anforderungen, die erfüllt werden müssen.

Zukünftige Produktinnovationen im Maschinenbau und anderen technikorientierten Branchen, wie der Automobilindustrie oder der Medizintechnik, werden aber nicht nur durch rein ingenieurwissenschaftliche Ansätze entstehen. Die Informationstechnik und auch nicht-technische Disziplinen, wie die Kognitionswissenschaft oder die Neurobiologie, bringen eine Vielfalt an Methoden, Techniken und Verfahren hervor. Mit diesen können sensorische, aktorische und kognitive Funktionen in technische Systeme integriert werden, die bislang nur von biologischen Systemen erfüllt wurden. Insbesondere kognitive Funktionen versetzen technische Systeme in die Lage, sich noch besser in ihre Umgebung einzubinden und in dieser selbständig zu agieren. Ziel sind intelligente adaptive Systeme, die in ihrer Funktionsweise flexibler sowie robuster als auch auf den Umgang mit dem menschlichen Benutzer abgestimmt sind [Eur09, S. 27], [BW10, S. 1f.], [VMS07, S. 151]. Vier wesentliche Trends prägen diese faszinierende Perspektive:

- **Miniaturisierung der Elektronik:** Das Mooresche Gesetz besagt, dass sich die Prozessorleistung sowie die Transistorfunktionen in einem Zeitraum von 18 bis 24 Monaten verdoppeln[1]. Diese Gesetzmäßigkeit gilt zwar seit der Jahrtausendwende nur noch eingeschränkt, der technologische Fortschritt in der Mikroelektronik ermöglicht dennoch immer kleinere und leistungsfähigere Rechner. Das Parallelisieren der Informationsverarbeitung durch Multikernprozessoren, die immense Steigerung der Speicherkapazität und die Reduzierung des Energiebedarfs charakterisieren

[1] Benannt nach G. MOORE, einem der drei Mitbegründer der Firma Intel. MOORE erkannte und propagierte diesen Zusammenhang ab Mitte der 1960er Jahre.

fortschreitende Entwicklungen für die kognitive Hardware technischer Systeme [HS09, S. 76f.], [Kno03, S. 187].

- **Etablierung der Softwaretechnik:** Software ist ein wesentlicher Bestandteil technischer Systeme. Enorme Fortschritte sind bei der methodischen Entwicklung der Software für technische Systeme zu verzeichnen. So ist das Paradigma der objektorientierten Programmierung im Bereich eingebetteter Systeme und Echtzeitsysteme voll etabliert. Ferner werden Entwicklungsumgebungen und Programmiersprachen immer leistungsfähiger und robuster. Die Softwaretechnik ist als eigenständige Disziplin akzeptiert, ihr Wissen wurde bspw. im sog. SWEBOK[2] standardisiert. Die Anzahl an Algorithmen für technische Anwendung steigt immer weiter an und erfolgreich eingesetzte Softwaremodule können teilweise wiederverwendet werden [SEH+10, S. 3f.], [Kno03, S. 187f.].
- **Vernetzung von Informationssystemen:** Das „Internet der Dinge“, „Ubiquitous Computing“, „Pervasive Computing“ oder „Ambient Intelligence“ sind aktuelle Forschungsfelder, die sich mit der elektronischen Vernetzung von informationsverarbeitenden Systemen befassen. Ziel ist die konsequente und allgegenwärtige Durchdringung des privaten und beruflichen Alltags. Der Anwendungsbereich ist beachtlich –Güterlogistiksysteme bis hin zu intelligenten Prothesen sind in Planung und teilweise mit heute verfügbaren Technologien realisierbar. Die Vernetzung technischer Systeme wird das Umfeld des Menschen auch in Zukunft stark prägen, dabei interessiert insbesondere die Art und Weise der Koordination der vernetzten Systeme [Ram06, S. 2ff.], [HS09, S. 75f.].
- **Fortgeschrittene Mechatronik:** Mechatronische Systeme stellen eine neue Klasse von technischen Systemen dar. Ihre Leistungsfähigkeit geht über die rein mechanischer Systeme deutlich hinaus. Sie basieren auf dem engen Zusammenwirken von Mechanik, Elektrik/Elektronik, Regelungstechnik und Software. Aufgrund der sich abzeichnenden Entwicklung der Kommunikations- und Informationstechnik werden fortgeschrittene mechatronische Systeme realisierbar, die ein intelligentes Verhalten zeigen und über sog. „Self-X“-Fähigkeiten verfügen wie z.B. Selbstoptimierung, Selbstkoordination oder Selbstheilung [Trä09, S. 4].

Trotz dieser Trends und den offensichtlichen Nutzenpotentialen durch die Integration kognitiver Funktionen konnten bislang nur wenige Anwendungen realisiert werden. Bei der Mehrzahl handelt es sich um prototypische Entwicklungen aus dem Bereich der Robotik. Es fehlt in erster Linie an einer systematischen Verzahnung der für die Erforschung kognitiver Funktionen relevanten Disziplinen, wie der künstlichen Intelligenz, der Neurobiologie oder der Kognitionswissenschaft mit der ingenieurwissenschaftlichen Vorgehensweise in der Produktentwicklung, wie sie bspw. die klassische Konstrukti-

[2] SWEBOK: Guide to the Software Engineering Body of Knowledge [ISO19759].

onsmethodik lehrt. Die beiden wesentlichen Herausforderungen für die Entwicklung derartiger Systeme sind die verstärkte Interdisziplinarität und die steigende Komplexität der zu entwickelnden Systeme selbst. Beide führen zu einer erhöhten Komplexität der Entwicklung.

Diese Problematik führt zu der Notwendigkeit, die Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme methodisch zu unterstützen. Handlungsfeld ist der frühzeitige und disziplinübergreifende Entwurf eines Lösungskonzepts, das die prinzipielle Wirkungsweise des zu entwickelnden Systems beschreibt. Hier fehlt nicht nur ein systematisches und idealisiertes Vorgehen, sondern auch eine Technik zur Identifikation und Spezifikation der kognitiven Funktionen innerhalb der Architektur der Informationsverarbeitung fortgeschrittener mechatronischer Systeme. Weiterhin fehlt eine Methode zur Dokumentation und Wiederverwendung von erfolgreich eingesetztem Lösungswissen für die Implementierung und Umsetzung kognitiver Funktionen. Summa summarum fehlt eine Entwicklungssystematik, die die Entwicklungstätigkeiten strukturiert und entsprechende Hilfsmittel bereitstellt.

1.2 Zielsetzung

Ziel der Arbeit ist eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*. Sie soll so konzipiert sein, dass sie in den frühen Phasen der Entwicklung ein interdisziplinäres Entwicklerteam unterstützt.

Die Entwicklungssystematik soll – ausgehend von den Systemanforderungen, ersten Systemzielen und einer funktionalen Systembeschreibung – potentielle Lösungen für die Realisierung kognitiver Systemfunktionen identifizieren sowie derart beschreiben, dass eine systematische Integration in die Spezifikation des Gesamtsystems erfolgen kann. Kern bildet ein strukturiertes sowie in den Gesamtentwicklungsprozess integriertes Vorgehen und die Dokumentation von Lösungen in einer wiederverwendbaren Form. Als Resultat liegt die frühzeitige Spezifikation der kognitiven Informationsverarbeitung eines fortgeschrittenen mechatronischen Systems vor. Diese dient als Ausgangspunkt für die Systemkonkretisierung und -implementierung in den nachfolgenden Entwicklungsphasen.

Die durchgängige Anwendbarkeit der Entwicklungssystematik soll anhand der Konzipierung eines Demonstrators des SFB 614 nachgewiesen werden. Hier handelt es sich um ein hybrides Energiespeichersystem des autonomen Schienenfahrzeugs RailCab.

1.3 Vorgehensweise

In **Kapitel 2** wird die einleitende Problematik konkretisiert. Dazu wird das Forschungsfeld der Arbeit abgesteckt. Ausgangspunkt ist eine Beschreibung der Funktionsweise mechatronischer Systeme. Auf die Betrachtung adaptiver Systeme folgt eine Einführung in die Grundlagen der Selbstoptimierung und entsprechender Systeme als Beispiel für

fortgeschrittene mechatronische Systeme. Da die Realisierung der Informationsverarbeitung derartiger Systeme Funktionen voraussetzt, die dem Phänomen der Kognition zuzuschreiben sind, wird dieses näher betrachtet. Ferner werden die Entwicklung fortgeschrittener mechatronischer Systeme und daraus resultierende Herausforderungen behandelt. Der Fokus liegt dabei auf den frühen Phasen der Entwicklung, der sog. Konzipierung. Vor diesem Hintergrund schließt die Problemanalyse mit der Ableitung von Anforderungen an die geforderte Entwicklungssystematik.

Kapitel 3 gibt einen Überblick über den Stand der Technik. Hierfür werden zunächst methodische Ansätze zur Entwicklung kognitiver Funktionen in technischen Systemen beschrieben. Danach werden Methoden zur Beschreibung von Funktionen im Rahmen der Entwicklung untersucht. Strukturelle Ansätze zur Integration kognitiver Funktionen setzen die Analyse des Stands der Technik fort. Nachdem Techniken zur ganzheitlichen Beschreibung der kognitiven Informationsverarbeitung analysiert werden, stehen Muster zur Dokumentation wiederkehrender Lösungen in den frühen Phasen der Entwicklung im Fokus. Zusätzlich werden zwei Konzepte für IT-Werkzeuge zur Unterstützung der Entwickler bei der Suche und Ablage von Lösungswissen analysiert. Schließlich wird anhand der in Kapitel 2 ermittelten Anforderungen der Stand der Technik bewertet und der Handlungsbedarf identifiziert.

Kern der Arbeit bildet das **Kapitel 4**. In diesem wird die Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme beschrieben. Das Kapitel startet mit einem Überblick über die Entwicklungssystematik. Danach folgt die Herleitung einer gemeinsamen Begriffswelt der Mechatronik und der Kognitionswissenschaft, die die Grundlage der Entwicklungssystematik bildet. Anschließend werden die einzelnen Bestandteile der Entwicklungssystematik vorgestellt.

Die Validierung der Arbeit erfolgt in **Kapitel 5**. Anhand eines Demonstrators wird die durchgängige Anwendung der erarbeiteten Entwicklungssystematik gezeigt. Der Demonstrator ist so gewählt, dass die Erfüllung der aufgestellten Anforderungen und der Nutzen der Entwicklungssystematik validiert werden können.

Kapitel 6 besteht aus einer Zusammenfassung und einem Ausblick auf zukünftige Arbeiten. Der **Anhang** umfasst ergänzende Informationen zum Stand der Technik sowie zu der erarbeiteten Entwicklungssystematik.

2 Problemanalyse

Ziel der Problemanalyse sind Anforderungen an eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*. Hierfür werden in Kapitel 2.1 wesentliche Begriffe definiert und die Arbeit im Kontext des Sonderforschungsbereichs 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) eingeordnet. Danach werden in Kapitel 2.2 Aufbau und Funktionsweise fortgeschrittener mechatronischer Systeme vorgestellt. Im Rahmen dieser Arbeit werden mechatronische Systeme, adaptive Systeme und selbstoptimierende Systeme des Maschinenbaus unterschieden. Im Fokus von Kapitel 2.3 stehen sog. kognitive Funktionen, die die Basis für eine Weiterentwicklung mechatronischer Systeme hin zu fortgeschrittenen mechatronischen Systemen sind. In Kapitel 2.4 wird auf die Entwicklung fortgeschrittener mechatronischer Systeme und deren Herausforderungen näher eingegangen. Kapitel 2.5 umfasst eine Problemabgrenzung aus der in Kapitel 2.6 die Anforderungen an die Entwicklungssystematik resultieren.

2.1 Begriffsdefinitionen und Einordnung der Arbeit

Ziel dieser Arbeit ist eine Entwicklungssystematik, die Entwickler in den frühen Phasen des Entwurfs unterstützt. Der Begriff (Produkt-)**Entwicklung** steht im Kontext dieser Arbeit für die Gesamtheit an Tätigkeiten zur Definition, Lösung und Realisierung einer technischen Aufgabe, deren Resultat ein technisches System ist. Sie grenzt sich hierdurch von der Forschung ab, die neue Lösungen zur Umsetzung der technischen Aufgabe erarbeitet. Der zugehörige Entwicklungsprozess erfolgt *vom Markt zum Markt*, also von der Marktforschung, die neue Kundenanforderungen identifiziert, zum Absatzmarkt, der das entwickelte System nutzt [Sau06, S. 19].

Der Begriff **Systematik** ist insbesondere in der Biologie etabliert und beschreibt die Teildisziplin Biosystematik. Diese beschäftigt sich maßgeblich mit der systematischen Benennung und Bestimmung von Lebewesen und Pflanzen [LL06]. In der Brockhaus Enzyklopädie ist eine Systematik grundsätzlich als eine *einheitliche Darstellung und Gliederung nach sachlichen und logischen Gesichtspunkten* definiert [Bro03, S. 1942].

In den fünfziger Jahren des letzten Jahrhunderts wurde in der Firma Carl Zeiss Jena der Begriff **Konstruktionssystematik** geprägt, der in den Folgejahren in der damaligen DDR auch in der Wissenschaft verwendet wurde. Getrieben von einem Mangel an Führungs- und Fachkräften in der Konstruktionsabteilung war es das Ziel, die Konstruktionstätigkeiten durch ausführliches Analysieren der in der Konstruktion ablaufenden Vorgänge zu optimieren. Wesentliche Ergebnisse der Konstruktionssystematik sind die Strukturierung der Konstruktionstätigkeiten in einem logischen und idealisierten Konstruktionsprozess sowie geeignete Hilfsmittel für dessen effektive Durchführung [Höh02]. Der damalige Konstrukteur bei Carl Zeiss Jena und spätere Professor an der TU Ilmenau F. HANSEN definierte die Konstruktionssystematik als das *planmäßige, wis-*

senschaftliche Kombinieren der Einzelerkenntnisse der Technik zum Aufbau eines technischen Gebildes [Han55, S. 36].

In diesem Sinne beschreibt eine **Entwicklungssystematik** ein universelles Rahmenwerk, das ein Vorgehensmodell sowie dedizierte Hilfsmittel zur erfolgreichen Umsetzung der Entwicklung technischer Systeme bereitstellt. Sie ermöglicht weder ein automatisiertes Entwickeln noch ist sie ein Ersatz für die kreative Leistung des Anwenders. Das **Vorgehensmodell** strukturiert den Entwicklungsprozess nach aufgabenspezifischen Gesichtspunkten. **Hilfsmittel** können bspw. Methoden, Richtlinien, Spezifikationstechniken/Modellierungssprachen, Konstruktionsprinzipien, Entwurfsmuster oder Werkzeuge sein.

Der Begriff Entwicklungssystematik grenzt sich in zwei Punkten von dem Begriff **Entwicklungsmethodik**[3] ab. Zum einen fließen in eine Entwicklungsmethodik denk- und arbeitspsychologische Untersuchungen ein. Zum anderen betrachtet die Entwicklungsmethodik zusätzlich organisatorische Aspekte [PBF+07, S. 10f.]. Beide Punkte fehlen in einer Entwicklungssystematik.

Demzufolge adressiert die hier zu erarbeitende Entwicklungssystematik ein strukturiertes Vorgehen und dedizierte Hilfsmittel zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme. Sie soll in den frühen Phasen der Entwicklung eingesetzt werden. Notwendige bestehende Hilfsmittel sind dabei in die Entwicklungssystematik zu integrieren, neue sind dann zu entwickeln, falls es keine geeigneten gibt.

Im Rahmen des **Sonderforschungsbereichs 614 (SFB 614)**, dessen Gegenstand *maschinenbauliche Erzeugnisse von morgen* sind [SFB08, S. 9], schlägt die Arbeit eine Brücke zwischen den Projektbereichen „Grundlagen und Potentiale der Selbstoptimierung“ und „Entwurfsmethoden und -werkzeuge“. Sie stellt eine Schnittstelle zwischen Grundlagenforschung und ingenieurwissenschaftlicher Aufbereitung des Wirkparadigmas der Selbstoptimierung dar und trägt so zu einer neuen Schule des Entwurfs fortgeschrittener mechatronischer Systeme bei.

2.2 Fortgeschrittene mechatronische Systeme

Die rasante und fortschreitende Entwicklung der Mikroelektronik, Kommunikations- und Informationstechnik ergeben für den modernen Maschinenbau und verwandten Branchen, wie der Automobilindustrie oder der Medizintechnik, neue Möglichkeiten für erfolgsversprechende Produktinnovationen. Bereits heute ist absehbar, dass durch eine weiter ansteigende Integration von informationsverarbeitenden Komponenten in technische Systeme, neue Systemfunktionen realisierbar sein werden [HS09, S. 75].

[3] In der Literatur wird oftmals der Begriff Konstruktionsmethodik verwendet. Sie sind als identisch anzusehen.

Die Mechatronik und auf ihr basierende Systeme drücken diese Perspektive in besonderem Maße aus. Sie ermöglicht neue Erfolgspotentiale für die Gestaltung von innovativen Systemen, wie die Funktions- und Verhaltensverbesserung, die Reduzierung von Baugröße, Gewicht und Kosten, die Realisierung völlig neuer Funktionen und die Entwicklung intelligenter, autonomer Systeme [Möh04, S. 6], [Ise08, S. 21]. Vor diesem Hintergrund erfolgt in den folgenden Abschnitten[4] eine Beschreibung fortgeschrittener mechatronischer Systeme in drei aufeinander aufbauenden Systemklassen. Dies verdeutlicht die Entwicklung mechatronischer Systeme hin zu intelligenten technischen Systemen, die auf der Durchdringung der Ingenieurwissenschaft durch die Informatik und Mathematik basieren.

2.2.1 Mechatronische Systeme

Der Begriff Mechatronik ist ein Kofferwort aus Mechanik und Elektronik, das 1969 die Japaner T. MORI und K. KIKUCHI erschufen und daraufhin von Yaskawa Electric als Handelsname eingetragen wurde [Mor69], [Com94], [Roh08-ol]. Ursprünglich drückte es die Erweiterung mechanischer Systeme um elektronische Funktionen aus. Heute existiert eine Vielzahl an Definitionen, deren Kern im Wesentlichen HARASHIMA, TOMIZUKA und FUKUDA prägten. Sie betonten als erste das synergetische Zusammenwirken von Mechanik, Elektrik/Elektronik, Regelungs- und Softwaretechnik und schließen darüber hinaus in ihrer Definition der Mechatronik die Entwicklung und die Produktion ein [HTF96, S. 1]. Die VDI-Richtlinie 2206 „Entwicklungsmethodik für mechatronische Systeme“ bietet eine Übersetzung der im Original engl. Definition.

> *„Mechatronik bezeichnet das synergetische Zusammenwirken der Fachdisziplinen Maschinenbau, Elektrotechnik und Informationstechnik beim Entwurf und der Herstellung industrieller Erzeugnisse sowie bei der Prozessgestaltung.“ [VDI2206, S. 14].*

Mechatronische Systeme bestehen in der Regel aus vier Einheiten: einem Grundsystem, einer Sensorik, einer Aktorik und einer Informationsverarbeitung. Diese vier Einheiten bilden einen systeminternen Regelkreis. Grundsätzlich werden diese Einheiten in einer Umgebung betrieben. Nach außen bestehen grundsätzlich zwei Schnittstellen: eine Mensch-Maschine-Schnittstelle über die der Mensch Einfluss auf das System nehmen kann und einem Kommunikationssystem für den Informationsaustausch mit anderen (technischen) Systemen. Die Leistungsversorgung kann sowohl extern als auch intern erfolgen [VDI2206, S. 14ff.]. Bild 2-1 zeigt die Grundstruktur eines mechatronischen Systems in Anlehnung an die VDI-Richtlinie 2206.

[4] Der Aufbau der Arbeit unterscheidet in Anlehnung an die DIN-Norm 1421 nicht zwischen Kapiteln und Abschnitten. Die Begriffe Unterkapitel und Unterabschnitt werden nicht verwendet [DIN1421].

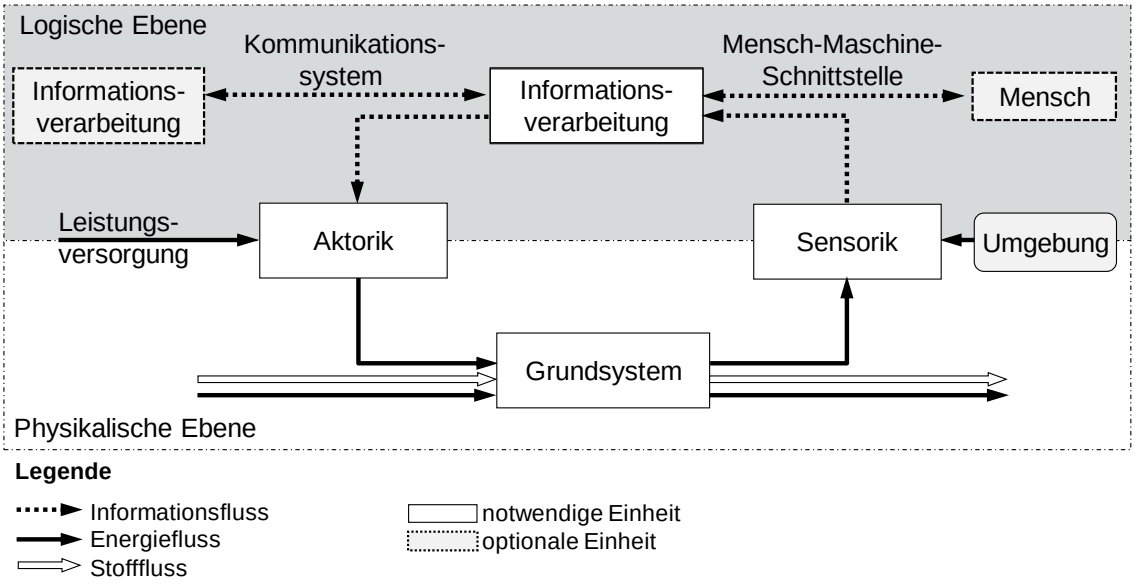


Bild 2-1: Grundstruktur eines mechatronischen Systems nach [VDI2206, S. 14]

Das **Grundsystem** bildet eine mechanische, elektro-mechanische, hydraulische oder pneumatische Grundstruktur. Technologische Weiterentwicklungen ermöglichen auch Kombinationen dieser Strukturen. Es ist die zentrale Einheit der *physikalischen Ebene*. Die **Sensorik** erfasst und bestimmt ausgewählte Zustandsgrößen[5] des Grundsystems als auch äußere Einflüsse der Umgebung. Die ermittelten Messwerte gehen im Anschluss als Eingangsgrößen an die Informationsverarbeitung. Zentrales Element auf der *logischen Ebene* ist die **Informationsverarbeitung**. Sie ermittelt alle relevanten Einwirkungen und bestimmt die Stellgrößen für die Aktorik. Dabei müssen nicht nur systemintern ermittelte Informationen berücksichtigt werden, sondern über die externen Schnittstellen können Informationen anderer technischer Systeme oder des Benutzers erfasst werden. Die **Aktorik** hat die Aufgabe, die Zustandsgrößen des Grundsystems in geeigneter Weise zu beeinflussen. Durch den technologischen Fortschritt erweitern sich die Einsatzfelder von Aktoren stetig.

Die Einheiten des mechatronischen Systems sind über Flüsse miteinander verbunden. Nach PAHL/BEITZ sind in einem technischen System drei Flussarten zu unterscheiden [PBF+07, S. 43]:

- **Stofffluss:** Stoffflüsse beschreiben den Transport und den Austausch von Gasen, Flüssigkeiten oder festen Körpern (z.B. Kühlflüssigkeit).
- **Energiefluss:** Mechanische, thermische und elektrische Energie und entsprechende Kenngrößen (z.B. Kraft oder Strom) werden durch Energieflüsse festgelegt.

[5] Zustandsgrößen eines technischen Systems sind physikalische Größen, durch deren Wert zu einem beliebigen Zeitpunkt t_0 der Ablauf des Systems für $t > t_0$ eindeutig bestimmt ist, sofern die Eingangsgrößen des System für $t > t_0$ gegeben sind [Föl08, S. 389].

- **Informationsflüsse:** Werden z.B. Messgrößen zwischen elektronischen Einheiten ausgetauscht, ist die Verbindung als Informationsfluss zu kennzeichnen.

Die Bandbreite mechatronischer Anwendungen ist groß. GAUSEMEIER unterscheidet hier zwei grundsätzliche Klassen mechatronischer Systeme. Systeme der Klasse 1 sind mechanisch-elektronische Baugruppen, die auf der räumlichen Integration von Mechanik und Elektronik beruhen. Klasse 2 umfasst Mehrkörpersysteme mit kontrolliertem Bewegungsverhalten. In Bild 2-2 sind einige Beispiele der beiden Klassen zu sehen.

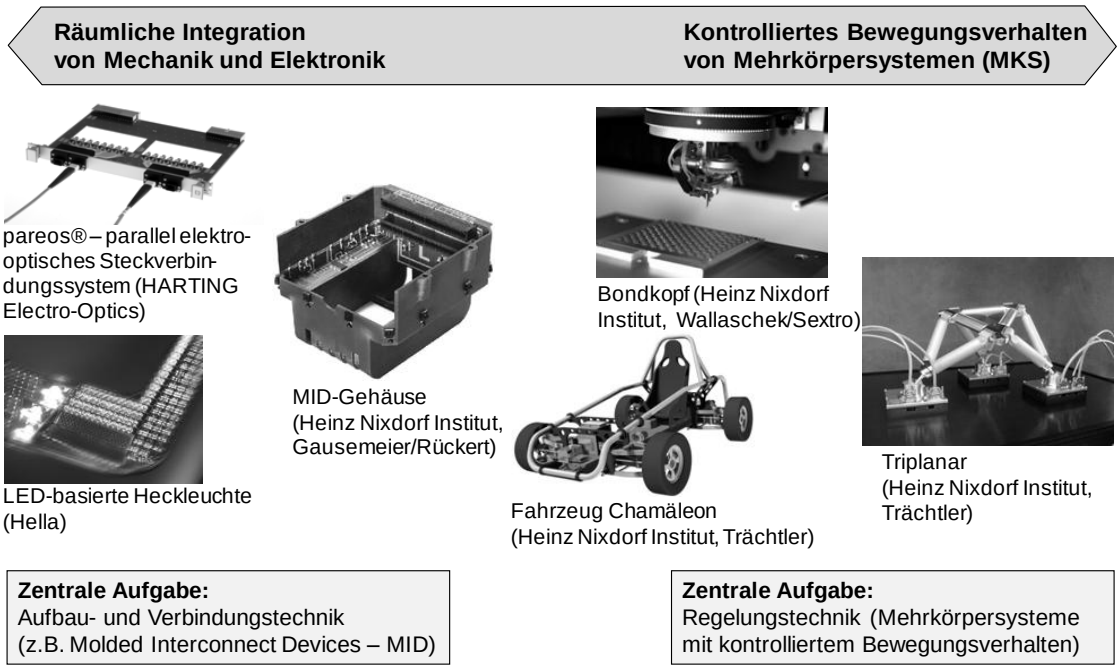


Bild 2-2: Klassen mechatronischer Systeme [Gau10, S. 15]

Ziel der **Klasse 1** ist eine hohe Integration mechanischer und elektronischer Funktionsträger auf kleinem Bauraum. Wesentliche Erfolgspotentiale liegen in der Miniaturisierung, Funktionsintegration, der höheren Zuverlässigkeit und den geringeren Herstellkosten. Zentrale Aufgabe ist die Aufbau- und Verbindungstechnik z.B. auf Basis innovativer Technologien wie MID (Molded Interconnect Devices). Hervorzuheben ist, dass zwischen dem Produkt und dem zugehörigen Produktionssystem starke Abhängigkeiten herrschen und diese daher parallel und integrativ zu entwickeln sind [GF06, S. 3f.].

Fokus der **Klasse 2** liegt auf der Verbesserung des kontrollierten Bewegungserhaltens von Mehrkörpersystemen. Systeme dieser Klasse können gemäß der oben beschriebenen Grundstruktur mechatronischer Systeme selbstständig auf Veränderungen ihrer Umgebung reagieren. Im Vordergrund steht der Entwurf der Regelung als Basis für das kontrollierte Bewegungsverhalten. Die Regelung ist ein Vorgang bei dem fortlaufend eine zu regelnde Größe, die Regelgröße, erfasst, mit einer anderen Größe, der Führungsgröße, verglichen und an diese in vorgeschriebener Abhängigkeit angeglichen wird. Resultat ist ein geschlossener Wirkungskreis, bei dem die Regelgröße durchge-

hend sich selbst beeinflusst. Die Regelung besteht grundsätzlich aus der Regeleinrichtung (bzw. dem Regler als Teil der Regeleinrichtung) und der Regelstrecke [GF06, S. 3], [DIN19225, S. 2].

In der Praxis treten beide Klassen mechatronischer Systeme oftmals in Kombination auf. Nicht selten realisieren mechatronische Systeme der Klasse 1 Elemente der mechatronischen Systeme der Klasse 2. Beispiele hierfür sind integrierte Sensoren oder miniaturisierte Elektroantriebe. Im Rahmen der vorliegenden Arbeit liegt der Schwerpunkt auf den Systemen der Klasse 2, die auch die Basis für fortgeschrittene mechatronische Systeme in diesem Kontext sind. Daher wird im Folgenden unter einem mechatronischen System stets die Klasse 2 verstanden.

2.2.2 Adaptive Systeme

Es existiert keine anerkannte Definition des Begriffs Adaption[6]. Eine Ursache ist seine generische Verwendung. So beschreibt die Adaption in der Medizin die grundsätzliche Anpassungsfähigkeit von Lebewesen oder Organismen, während sie in der Informatik bspw. die Anpassung eines Datenformats für eine Hardwaremigration kennzeichnet. Einigkeit besteht darin, dass Adaption das Anpassen eines Verhaltens an neue Umstände beschreibt. Im Kontext mechatronischer Systeme versteht man unter einem adaptiven System ein geregeltes System, das sein Verhalten an die Änderungen der Prozessdynamik und der Störgrößen anpassen kann. Kern ist eine adaptive Regelung, die verstellbare Parameter besitzt und in der Lage ist, diese zu ändern [AW95, S. 1].

Adaptive Systeme sind grundsätzlich von adaptronischen Systemen abzugrenzen. Der Begriff Adaptronik wurde vom VDI Technologiezentrum kreiert. Er beschreibt eine der Mechatronik zwar ähnliche, aber klar von ihr zu unterscheidende Disziplin. Sie befasst sich mit geregelten Systemen, deren Funktionalität auf Multifunktionswerkstoffen und deren Struktur sowie Verhalten beruht. Diese integrieren die Sensorik und Aktorik. Beispiele hierfür sind Piezokeramiken oder Formgedächtnislegierungen [Jan07].

Nach der geschilderten Charakterisierung passen adaptive Systeme ihr Verhalten im Hinblick auf die Stabilität, Dynamik und Statik gemäß vordefinierter Maßnahmen an veränderte Betriebsbedingungen an. Im Normalfall erfolgt dies durch Einstellen oder Ausbessern der Regelung bzw. der Regelparameter. Die hierfür notwendige Adaptionseinrichtung kann das Grundsystem derart beeinflussen, dass es bei unvorhersehbaren, zeitlich unabhängigen Ereignissen oder Störungen, bestmöglich im Hinblick auf vorgegebene Ziele arbeitet (Bild 2-3) [SR88, S. 18f.].

---

[6] Auch als Adaptation, Adaptierung, Adaptivität oder Adaptabilität bezeichnet. Alle Bezeichnungen stammen vom lateinischen Verb *adaptare, adapto* = anpassen, passend herrichten ab [Lan91].

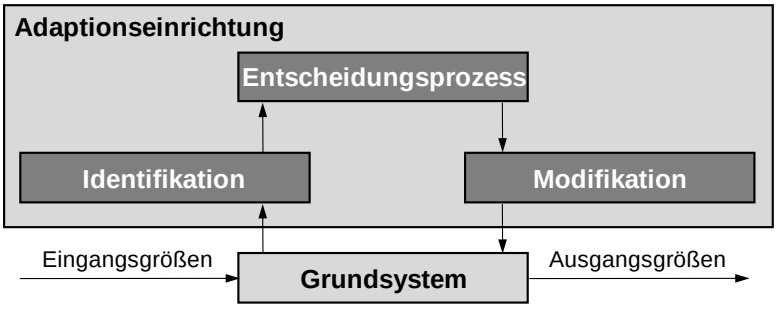


Bild 2-3: Grundstruktur eines adaptiv geregelten Systems nach [SR88, S. 18]

Das Grundsystem stellt das System dar, dessen Parameter nachzustellen sind. Die Adaptionseinrichtung führt hierzu drei für die Adaption charakteristische Teilprozesse aus, die gleichzeitig die grundlegenden Fähigkeiten eines adaptiven Systems beschreiben [SR88], [CS04], [Nau00, S. 8f.]:

1) Identifikation: Von veränderlichen Größen werden kontinuierlich die Istwerte erfasst und daraus Kennwerte des Grundsystems abgeleitet. Ziel ist die Wahrnehmung einer Änderung im Systemumfeld. Dazu können Schätzverfahren oder statistische Verfahren eingesetzt werden.

2) Entscheidungsprozess: Für eine vorher ermittelte Änderung im Systemumfeld erfolgt mittels Optimierungsverfahren eine Berechnung von Stellsignalen für den adaptiven Systemeingriff.

3) Modifikation: Die Durchführung des Systemeingriffs erfolgt durch das Nachstellen der Parameter der Regelung des Grundsystems.

Folglich ist die Adaption einer klassischen Regelung übergelagert und kann diese modifizieren. Adaptive Systeme können auf diese Weise Prozesse regeln, auch wenn detaillierte Kenntnisse hinsichtlich des Betriebs zum Zeitpunkt der Systementwicklung nicht vorhanden waren. Dabei kann die Adaption entweder direkt oder indirekt erfolgen (Bild 2-4) [SR88].

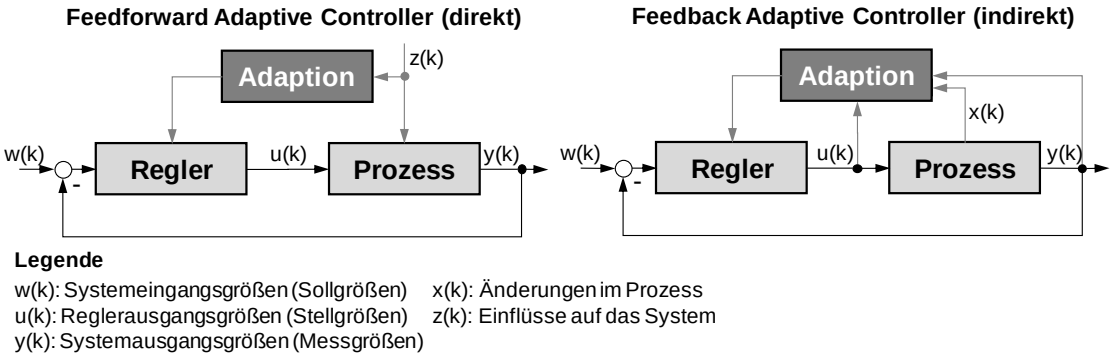


Bild 2-4: Direkte und indirekte adaptive Regelung nach [ILM92]

Die direkte adaptive Regelung wird durch einen **Feedforward Adaptive Controller** realisiert. Hierfür ist es notwendig, dass der Einfluss der Änderung auf den Prozess bekannt ist. Erfolgt der adaptive Eingriff per Rückführung der Systemausgangsgrößen nach der Bestimmung der Einflüsse auf den Prozess mittels eines Modells, spricht man von einer indirekten adaptiven Regelung. Sie wird als **Feedback Adaptive Controller** bezeichnet. In beiden Fällen werden die Zielgrößen bereits beim Reglerentwurf vom Entwickler festgelegt und können nicht mehr vom System modifiziert werden. Die Regelparameter werden allerdings im laufenden Betrieb an die jeweiligen Einflüsse angepasst, um die vorgegebenen Ziele zu erfüllen [ILM92], [IS95], [SB89].

Die Umsetzung adaptiver Regelungen begann bereits in den 1950er. Ziel waren adaptive Autopiloten unter anderem für Überschallflugzeuge. Diese ändern ihr Systemverhalten bei zunehmender Fluggeschwindigkeit stark [IS95]. Mit dem Einzug der Digitaltechnik und der Fly-by-Wire-Technologie verschmolz der Autopilot mit der Fluglagesteuerung. Seither wird an der adaptiven Flugsteuerung (adaptive flight control), insbesondere für unbemannte Luftfahrtzeuge gearbeitet [RC98], [JK02].

Da adaptive Systeme sich grundsätzlich auf unterschiedliche Betriebssituationen einstellen können, eignen sie sich für Anwendungen, in denen A-priori-Informationen fehlen. Dort können Systemeigenschaften hinsichtlich Sicherheit, Funktionalität oder Lebensdauer verbessert werden. Adaptive Systeme sind als Erweiterung geregelter, mechatronischer Systeme zu verstehen. Diese Erweiterung drückt sich in erster Linie durch die übergeordnete adaptive Regelung aus, die das Verhalten des mechatronischen Systems optimiert. Daher gibt es in der Mechatronik neben der Flugzeugtechnik eine Vielzahl weiterer Beispiele adaptiver Systeme auf die im Rahmen dieser Arbeit nicht näher eingegangen werden kann.

2.2.3 Selbstoptimierende Systeme

Die moderne Informations- und Kommunikationstechnik ermöglicht bereits heute fortgeschrittene mechatronische Systeme, die über intelligente Systemfunktionen verfügen. Im SFB 614 wird hierfür der Begriff Selbstoptimierung verwendet und derartige Systeme als selbstoptimierend bezeichnet:

> *„Unter **Selbstoptimierung** (self-optimization) eines technischen Systems wird die endogene Anpassung der Ziele des Systems auf veränderte Einflüsse und die daraus resultierende zielkonforme autonome Anpassung der Parameter und ggf. der Struktur und somit des Verhaltens dieser Systeme verstanden. Damit geht Selbstoptimierung über die bekannten Regel- und Adaptionsstrategien wesentlich hinaus; Selbstoptimierung ermöglicht handlungsfähige Systeme mit inhärenter „Teilintelligenz“, die in der Lage sind, selbständig und flexibel auf veränderte Betriebsbedingungen zu reagieren.“ [ADG+09, S. 5].*

In Bild 2-5 sind die wesentliche Aspekte eines selbstoptimierenden Systems und deren Zusammenwirken dargestellt.

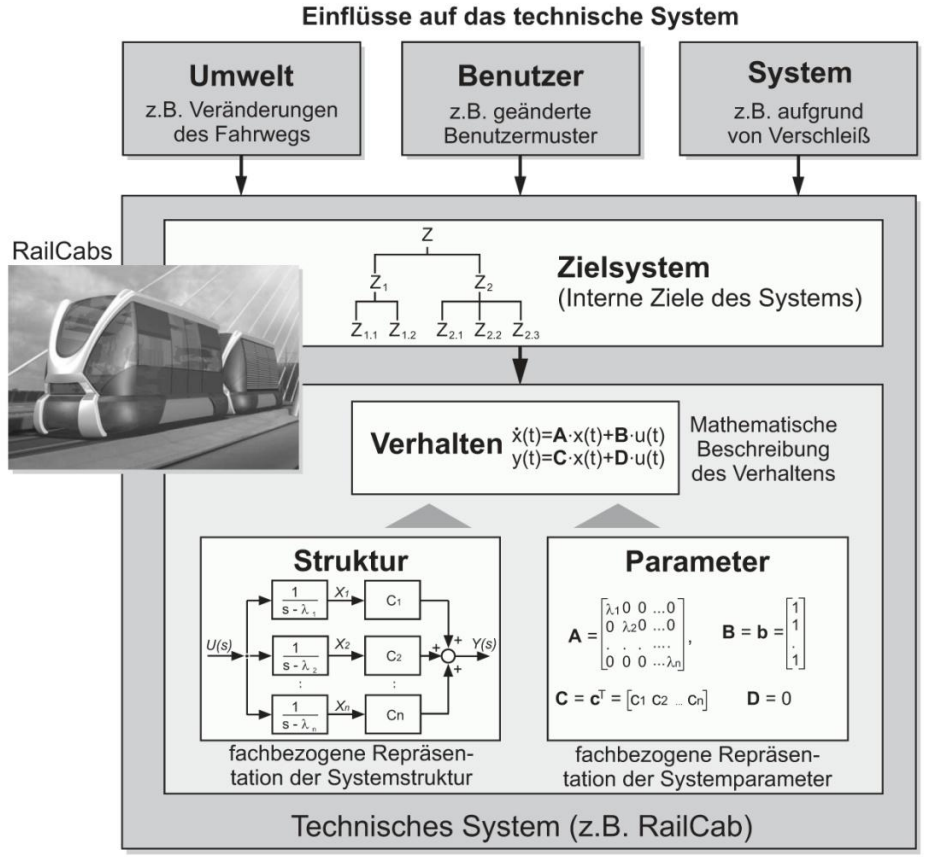


Bild 2-5: Aspekte eines selbstoptimierenden Systems – Einflüsse auf das System bewirken eine Veränderung der Ziele und eine entsprechende Anpassung des Systemverhaltens [SFB08, S. 18]

Grundsätzlich handelt es sich bei einem selbstoptimierenden System um ein **technisches System**. Ein System besteht im Kontext der Selbstoptimierung und in Anlehnung an die DIN 19226 aus Elementen, die wiederum selbst Systeme sein können und in Beziehung zueinander stehen [DIN19226]. Eine Klasse von Beziehungen wurde mit den Flussarten nach PAHL/BEITZ bereits eingeführt (vgl. Kap. 2.2.1).

Auf das selbstoptimierende System wirken verschiedene **Einflüsse** – unterstützend oder störend. Potentielle externe Einflussquellen sind die Umwelt, der Benutzer oder andere technische Systeme. Vom System selbst können aber auch Einflüsse stammen bzw. kann es wiederum auf die externen Einflussquellen wirken.

Ziele beschreiben die geforderten, gewünschten oder zu vermeidenden Systemeigenschaften. Sie werden während der Entwicklung definiert. Die situationsbedingte und selbstständige Ausprägung der Ziele erfolgt aber erst im Betrieb. Grundsätzlich sind drei Arten von Zielen zu unterschieden. Von außen an das System herangetragene Ziele werden *externe Ziele* genannt. *Inhärente Ziele* definieren den Entwurfszweck des Systems und beschreiben die zu entwickelnde Funktionalität. Die zu einem bestimmten

Zeitpunkt aktiven Ziele werden *interne Ziele* des Systems genannt. Sie leiten sich durch Auswahl, Kombination als auch Gewichtung aus den externen und den inhärenten Zielen ab und bilden das interne **Zielsystem**. Je nach Komplexität handelt es sich um einen Zielvektor, eine Zielhierarchie oder einen Zielgraphen [ADG+09, S. 20f.].

Das geregelte (Bewegungs-)**Verhalten** eines selbstoptimierenden Systems kann analog zu einem mechatronischen System in einem mathematischen Modell beschrieben werden. Es besteht aus allen in einer bestimmten Situation verfügbaren Aktionen, um von einem Zustand in einen anderen zu gelangen. Eine Veränderung der internen Ziele resultiert in einer Anpassung des Verhaltens. Eine Verhaltensanpassung kann auf zwei Arten erfolgen (Bild 2-6).

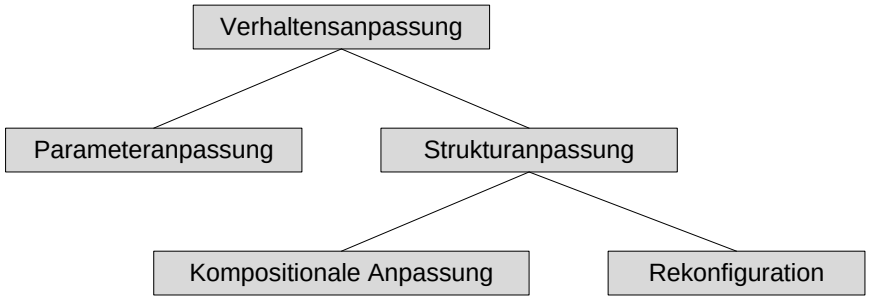


Bild 2-6: Möglichkeiten von Verhaltensanpassungen [ADG+09, S. 7]

Unter einer **Parameteranpassung** ist die Anpassung eines Systemparameters zu verstehen. Parameter sind Größen, deren Werte das Verhalten eines Systems bzw. seiner einzelnen Elemente bei gegebener Struktur beschreiben. Eine **Strukturanpassung** umfasst die Änderungen in der Anordnung und Beziehungen zwischen den Systemelementen. Eine Struktur stellt die Gesamtheit sämtlicher Beziehungen zwischen den Elementen in einem System dar. Zwei Arten der Strukturanpassung werden unterschieden. Die **kompositionale Anpassung** umfasst das Einfügen, die Entnahme, die Aktivierung und die Deaktivierung neuer oder bestehender Systemelemente. Die **Rekonfiguration** führt zu einer Änderung der Anordnung und Beziehungen bestehender Systemelemente [ADG+09], [DIN19226].

Die Selbstoptimierung erfolgt als eine Serie drei aufeinander folgender und wiederkehrender Aktionen, dem **Selbstoptimierungsprozess**. Dieser führt zu einem Zustandsübergang, wenn sich Einflüsse auf das System ändern, und gibt somit das Anpassungsverhalten eines selbstoptimierenden Systems wieder [SFB08, S. 19f.], [ADG+09, S. 6].

1) **Analyse der Ist-Situation (Situationsanalyse):** Die erste Aktion umfasst neben der Aufzeichnung relevanter Zustandsgrößen des Systems auch die Wahrnehmung äußerer Einflüsse. Informationen über das Umfeld können dabei entweder direkt mittels der Sensorik vom System oder indirekt per Kommunikation mit anderen Systemen gewonnen werden. Eine Voraussetzung für eine intelligente und lernende Verhaltensanpassung ist das Speichern sämtlicher Systemzustände, um einen Abruf

zurückliegender Aufzeichnungen zu ermöglichen. In diesem Schritt erfolgt auch die Überprüfung der Erfüllung der gegenwärtig aktiven Ziele für die aktuelle Situation.

2) **Bestimmung der Systemziele (Zielbestimmung):** Während der zweiten Aktion werden die aktiven Ziele des Systems bestimmt. Als Entscheidungsgrundlage dienen die in der Situationsanalyse gewonnenen Informationen. Die Zielbestimmung kann durch Auswahl, Anpassung und Generierung von Zielen eintreten. Bei der Auswahl von Zielen wird aus einer endlichen, zuvor festgelegten Menge von möglichen Zielen eine entsprechende Alternative selektiert. Die Anpassung von Zielen beinhaltet die schrittweise Veränderung der Ausprägung. Die Generierung beschreibt die Erstellung von neuen, bislang unbekannten Zielen.

3) **Anpassung des Systemverhaltens (Verhaltensanpassung):** Infolge der zuvor ermittelten Systemziele wird in der dritten Aktion das Systemverhalten angepasst. Bild 2-6 verdeutlicht die Möglichkeiten der Verhaltensanpassung. Die Ergebnisse werden mittels der Informationsverarbeitung weitergeleitet, woraus letztendlich eine Umsetzung der Verhaltensanpassung über die Aktorik erfolgt.

Kern der Selbstoptimierung ist folglich die fortlaufende, endogene Anpassung der Ziele des Systems im Sinne des geschilderten Selbstoptimierungsprozesses. Im Vergleich zur klassischen Regelung und adaptiven Regelung besitzen selbstoptimierende Systeme eine selbstoptimierende Regelung, die wesentlich über bekannte Regel- und Adaptionsstrategien hinaus geht (Bild 2-7).

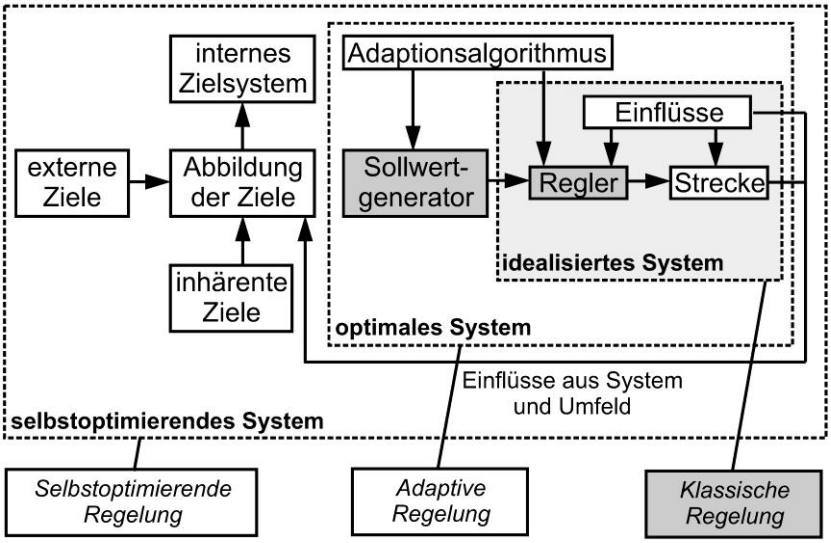


Bild 2-7: Erweiterung klassischer und adaptiver Regelungen zur selbstoptimierenden Regelung [ADG+09, S. 27] bzw. [BSK+06]

Die **selbstoptimierende Regelung** ist nicht als Alternative zur klassischen oder adaptiven Regelung zu verstehen, sondern als Erweiterung. Die klassische Regelung, bestehend aus Regeleinrichtung und Regelstrecke, führt zu einem hinsichtlich des Systemverhaltens idealisierten und gegenüber bekannten Einflüssen robusten System. Bei sich

ändernden Einflüssen, aber festen Zielen kann das Verhalten durch eine Adaptionsstrategie optimiert werden. Ergebnis ist ein optimales System. Eine Abbildung der Ziele wird notwendig, wenn sich die Ziele im Betrieb ändern können. Hierfür ist dann ein selbstoptimierendes System zu implementieren [BSK+06], [ADG+09, S. 27f.].

Die Strukturierung selbstoptimierender Systeme kann auf Basis der von LÜCKEL aufgestellten hierarchischen Strukturierung komplexer mechatronischer Systeme erfolgen. Diese orientiert sich an der Grundstruktur mechatronischer Systeme und erweitert diese um den Aspekt der Modularisierung und Strukturierung [LHL01], [Hes06] (Bild 2-8).

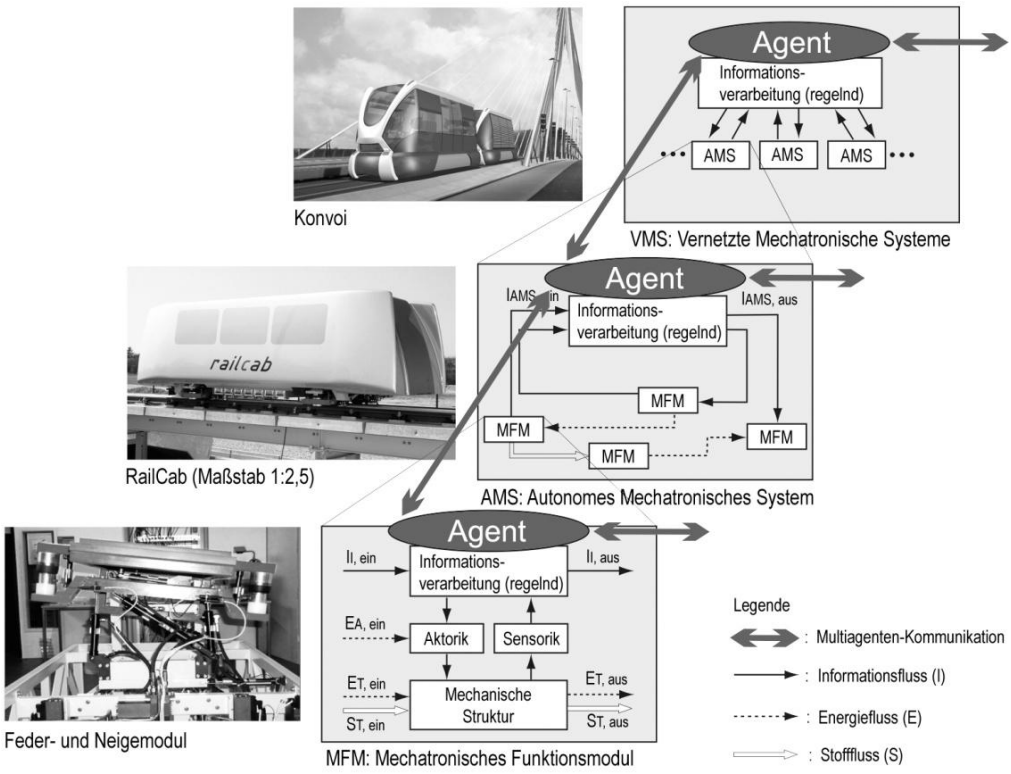


Bild 2-8: Struktur eines fortgeschrittenen mechatronischen Systems mit inhärenter Teilintelligenz [ADG+09, S. 11]

Grundsätzlich werden drei Strukturelemente unterschieden, die aus einer funktionalen Dekomposition des Bewegungsverhaltens eines mechatronischen Systems resultieren [Hes06]. Dementsprechend ergeben sich drei Hierarchieebenen[7]:

- **Mechatronisches Funktionsmodul (MFM):** Dieses Strukturelement entspricht der Grundstruktur mechatronischer Systeme mit seinen vier Grundeinheiten (vgl. Bild 2-1). Es bildet die Basis eines komplexen mechatronischen Systems und setzt in der Regel eine Teilfunktion der Hauptbewegungsfunktion um. Ein MFM kann Schnittstellen zu anderen MFM haben und ein anderes MFM als Aktor integrieren.

[7] Generisch besitzen technische Systeme „n“ Hierarchieebenen. Dennoch ist die Strukturierung in MFM, AMS und VMS für eine Vielzahl von technischen Systemen passend [ADG+09, S. 12].

- **Autonomes Mechatronisches System (AMS)**: Werden mehrere MFM in einem System verbaut, ergibt die Vernetzung dieser ein AMS. Das AMS kann selbstständig Handlungen ausführen und bedient sich hierfür der Aktorik der MFM. Das AMS erfüllt die Hauptbewegungsfunktion. Beispiele für ein AMS sind ein PKW oder ein autonomes Schienenfahrzeug.
- **Vernetztes Mechatronisches System (VMS):** Das oberste Strukturelement ist ein VMS. Es resultiert aus der rein informationstechnischen Kopplung mehrerer AMS. Auf diese Weise können übergeordnete Aufgaben der Informationsverarbeitung realisiert werden. Beispiele für ein VMS sind kooperierende Roboter oder ein autonomer Fahrzeugverbund.

Die Erweiterung mechatronischer Systeme zu selbstoptimierenden Systemen erfolgt durch die Ergänzung der regelnden Informationsverarbeitung der unterschiedlichen Strukturelemente durch eine selbstoptimierende Informationsverarbeitung. Das mechatronische Gesamtsystem sowie die einzelnen Teilsysteme werden so mit einer inhärenten Teilintelligenz ausgestattet und können autonom als auch kooperativ agieren. Aus informationstechnischer Sicht handelt es sich um ein Multiagentensystem, in dem einzelne Agenten miteinander kommunizieren und kooperieren können [ADG+09, S. 17], [RN07, S. 21]. Ein **Agent** ist im Rahmen des SFB 614 eine informationsverarbeitende Einheit, die

- vordefinierte Funktionen erfüllt und entsprechende Ziele verfolgt,
- hierfür notwendige Fähigkeiten in Form von Programmen besitzt,
- Informationen über das Umfeld sammelt und mit anderen Agenten Information austauscht und
- autonom sowie situationsgerecht das Systemverhalten anpasst.

Eine wesentliche Rolle für die Realisierung der Selbstoptimierung spielen Optimierungsverfahren, die auf den Agenten laufen. Es existieren modellorientierte und verhaltensorientierte Verfahren. Die Optimierung mittels **modellorientierter Verfahren** erfolgt anhand ingenieurwissenschaftlicher (physikalischer) Modelle. Die Modelle werden mathematisch durch Differentialgleichungen beschrieben. Die Optimierung ist dabei von der Regelung zeitlich entkoppelt, und optimierte Parameter oder andere Änderungen werden nach erfolgreicher Verifikation an die Regeleinheit weitergereicht.

Verhaltensorientierten Verfahren liegt nicht so ein physikalisches Modell zu Grunde. Der eigentliche Prozess wird daher in der Regel als Blackbox betrachtet. Auf diese Weise erfolgt eine Abbildung der als relevant identifizierten Eingangsgröße hinsichtlich der entsprechenden Ausgangsgröße. Damit geht häufig eine Diskretisierung des Optimierungsproblems einher. Entsprechende Modelle basieren auf Erfahrungswissen des Entwicklers oder durch Lern- oder Explorationsstrategien gebildet werden, sind verhaltens-

orientierte Verfahren weniger genau als modellorientierte. Im Gegenzug können aber längere Planungshorizonte berücksichtigt werden.

Hybride Verfahren kombinieren Ansätze beider Verfahrensarten. Ein Beispiel ist die hybride Planung. Diese wurde entwickelt, da verhaltensorientierte Verfahren, insbesondere die Planungsverfahren, Zustandsübergänge nur diskret betrachten. Da selbstoptimierende Systeme auf mechatronischen Systemen basieren, sollten zumindest Teile des Systems und die dort ablaufenden Prozesse kontinuierlich beschrieben werden. Dies betrifft vor allem die Hierarchieebene der MFM [EAS+08], [AK09]. Eine weitere Verfahrenskombination ist die verhaltensbasierte Entwurfspunktselektion. Diese unterstützt bei der situationsabhängigen Optimierung mechatronischer, kontinuierlicher Prozesse, für die mittels einer Mehrzieloptimierung eine Menge optimaler Kompromisse errechnet wird. Die situationsabhängige Auswahl einer Lösung aus dieser Menge basiert auf Kriterien, die nicht nur rein physikalisch modellierbar sind. Daher können verhaltensorientierte Suchstrategien die Auswahl präzisieren [SFB08, S. 282].

Beispiele selbstoptimierender Systeme sind Anwendungen des SFB 614. Bild 2-9 zeigt eine Auswahl an Demonstratoren, an denen die Methoden und Verfahren zur Entwicklung selbstoptimierender Systeme erprobt werden.

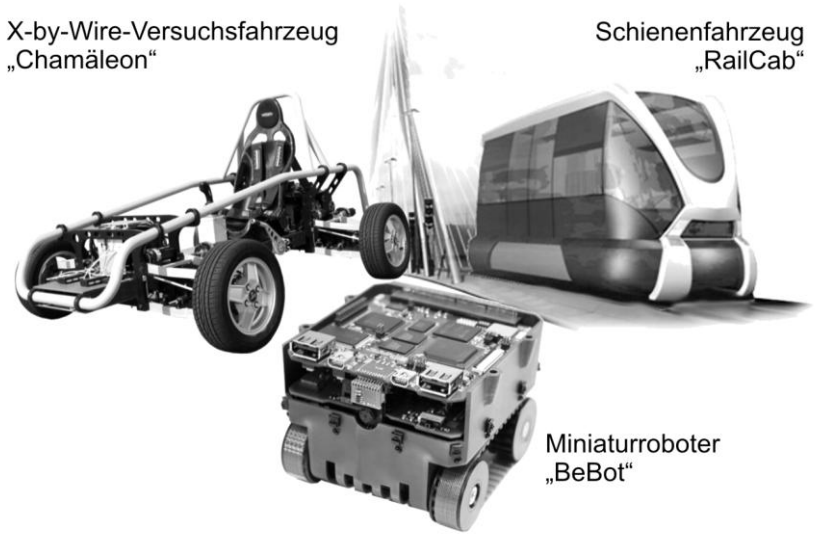


Bild 2-9: Demonstratoren des SFB 614 – Anwendungen der Selbstoptimierung

Die Bahntechnik ist ein wichtiger Anwendungsbereich im SFB 614. Kern bildet ein innovatives Bahnsystem mit autonomen Schienenfahrzeugen (**RailCabs)** für den Personen- und Güterverkehr[8]. RailCabs fahren bedarfsgerecht und nicht nach einem Fahrplan. Sie zeigen proaktives Verhalten indem sie selbstständig einen Konvoi zur Reduzierung des Energiebedarfs bilden. Ferner können sie Streckencharakteristika lernen und so

[8] Website des Projekts „Neue Bahntechnik Paderborn/RailCab“ der Universität Paderborn: http://nbp-www.upb.de/

bspw. den Fahrkomfort deutlich verbessern [TMV06]. Das RailCab-System veranschaulicht die Entwicklung von konventionellen hin zu fortgeschrittenen mechatronischen Systemen. Die im Rahmen dieser Arbeit zu erstellende Entwicklungssystematik wird an der Entwicklung eines Teilsystems des RailCabs validiert (vgl. Kap. 5.1).

Der autonome Miniaturroboter **BeBot** ist ein avantgardistisches Basissystem für die Forschung in den Bereichen dynamisch rekonfigurierbarer Systeme, Multiagentensysteme sowie Schwarmintelligenz[9]. Die Architektur der Informationsverarbeitung sowie die Vielzahl an Kommunikationsschnittstellen ermöglichen die Umsetzung der Selbstoptimierung auf verschiedenen Hierarchieebenen – insbesondere im kooperativen Verbund. Umgesetzte Szenarien sind bspw. ein mobiles Ad-hoc-Netzwerk aus mehreren BeBots, die ein robustes Kommunikationsnetzwerk realisieren, oder ein heterogenes Roboterteam bestehend aus BeBots und Robotern mit völlig anderer physischer Hardware (insb. Sensorik und Aktorik) [WEH+08], [AKH+10]. Das Wirkparadigma der Selbstoptimierung wird hier eingesetzt, um eine dezentrale und roboterbezogene Aufgabenverteilung im Roboterteam zu realisieren [ADG+09, S. 108ff.].

Der dritte abgebildete Demonstrator ist das voll aktive X-by-Wire-Versuchsfahrzeug **Chamäleon**. Mit diesem Elektrofahrzeug kann die Vernetzung von Fahrdynamikregelsystemen getestet werden. Hierzu werden neuartige Lenk- und Bremsstrategien entwickelt und am realen System validiert [RNJ+09], [NJT08]. Für weitere Anwendungen selbstoptimierender Systeme sei auf [ADG+09] verwiesen.

2.3 Kognitive Funktionen in technischen Systemen

Im Gegensatz zu konventionellen maschinenbaulichen Erzeugnissen weisen fortgeschrittene mechatronische Systeme die Möglichkeit für eine erhebliche Funktionssteigerung auf. Ansatzpunkt ist die Informationsverarbeitung, die zwischen Sensorik und Aktorik agiert. Während konventionelle mechatronische Systeme eine rein reaktive, regelnde Informationsverarbeitung besitzen, kann durch das Aufbrechen und Erweitern dieser Informationsverarbeitung ein Verhalten realisiert werden, das weit über reaktive und starre Verhaltensweisen hinausgeht. Diese Perspektive, mechatronische Systeme in die Lage zu versetzen, intelligent und flexibel auf veränderte Betriebsbedingungen zu reagieren, wird durch die Integration kognitiver Funktionen möglich.

Der Begriff Kognition, der sich von dem lateinischen Verb *cognoscere* bzw. im Griechischen *gignoskein* erkennen, wahrnehmen und wissen, ableitet, entstand im 19. Jahrhundert. In der Psychologie wurde ein Begriff gesucht, der die internen informationsverarbeitenden Mechanismen zur Verhaltenssteuerung ausdrückt [Len02, S. 9ff.]. Bis heute gibt es in der Wissenschaft keine einheitliche Definition der Kognition, die ihre

[9] Website des Projekts „Miniaturrobter BeBot“ am Heinz Nixdorf Institut in Paderborn: http://wwwhni.uni-paderborn.de/schwerpunktprojekte/miniaturroboter/

Grenzen klar absteckt. Daraus ergibt sich eine Unsicherheit in der Verwendung des Begriffs, die LENGELER et al. wie folgt beschreiben:

> *„Kognition wird als Komplex von Eigenschaften besonders höher entwickelter Lebewesen fest angenommen, aber wenig Genaues ist über die Voraussetzungen bekannt, die erfüllt sein müssen, um einem Lebewesen Kognition zusprechen zu können." [LMD99, S. 7].*

Einzig im Punkt, dass die Kognition eine bestimmte Art von Informationsverarbeitung darstellt und es unterschiedliche Formen von informationsverarbeitenden Prozessen gibt, herrscht Übereinstimmung [Str95], [Str06]. Aus diesem Grund wird in den folgenden beiden Abschnitten zunächst der Kognitionsbegriff weiter konkretisiert und die wesentlichen Grundlagen einer kognitiven Informationsverarbeitung beschrieben. Anschließend werden im letzten Abschnitt die Herausforderungen für die Integration kognitiver Funktionen in technische Systeme dargelegt.

2.3.1 Disziplinspezifische Sichten auf die Kognition

Das Phänomen Kognition und seine Erforschung ist aufgrund seiner vielen unterschiedlichen Aspekte Gegenstand mehrerer Fachdisziplinen. Die sog. kognitiven[10] Wissenschaften, denen die Kognitionswissenschaft übergeordnet ist, haben die Kognition als zentralen Forschungsgegenstand. Je nach kognitiver Wissenschaft wird ein anderer Schwerpunkt gelegt [Str96], [Str95]. Sie werden im Rahmen dieser Arbeit nicht explizit behandelt, sondern summiert in der Kognitionswissenschaft betrachtet. Dennoch ist es unausweichlich, das Phänomen Kognition aus Sicht verschiedener Fachdisziplinen zu beleuchten, um wichtige Punkte für die Realisierung kognitiver Funktionen in technischen Systemen hervorzuheben (Bild 2-10).

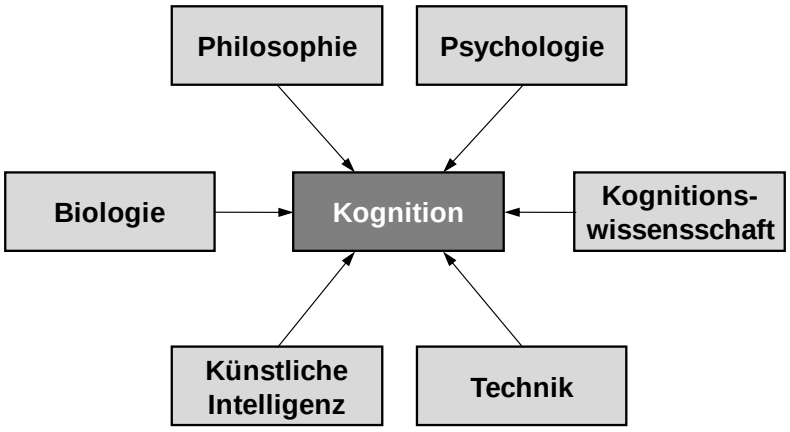


Bild 2-10: Wichtige disziplinspezifische Sichten auf die Kognition

[10] „Kognitiv" ist hier als Kurzform von „kognitiv orientiert" zu lesen. STROHNER unterscheidet dabei folgende sechs kognitive Wissenschaften, die interdisziplinäre Verbindungen aufweisen: Kognitive Philosophie, Kognitive Psychologie, Kognitive Linguistik, Kognitive Anthropologie, Kognitive Neurowissenschaft und Kognitive Informatik [Str95, S. 10ff].

In der **Biologie** untersucht insbesondere die Physiologie, wie Kognition in Lebewesen zustande kommt. Gegenstand sind sog. präkognitive Prozesse, die der eigentlichen Kognition vorausgehen. Dabei handelt es sich um vorbewusste und physiologische Abläufe, die die Grundbausteine der Kognition sein können. Grundlage der Forschung sind neurobiologische Untersuchungen des Gehirns. Neuere Ergebnisse konnten bspw. die Hirnregionen identifizieren, die für die kognitiven Funktionen bzw. zumindest deren Steuerung zuständig sind [RM96], [KTM+08].

Eine andere Sicht auf den Kognitionsbegriff, die auch der Biologie zuzuordnen ist, wurde von den Chilenen H. MATURANA und F. VARELA geprägt. Unter dem Leitsatz *life is cognition* setzten sie Kognition mit dem Prozess des Lebens gleich und definierten eine Systemtheorie der Kognition[11] [MV80, S. 13]. Diese besagt, dass lebende und sich selbst regelnde Systeme automatisch Kognition besitzen. MATURANA und VARELA prägten hierfür den Begriff Autopoiesis, der die Fähigkeit eines Systems zur Selbsterhaltung beschreibt [MV80], [MV90].

Die **Philosophie** befasst sich mit konzeptuellen und theoretischen Problemen der Kognitionsforschung. Sie wird meist als Philosophie des Geistes bezeichnet. Im Fokus steht neben erkenntnistheoretischen Untersuchungen das sog. Gehirn-Geist-Problem[12]. Kern ist die Frage, in welchem Zusammenhang mentale Zustände, insbesondere das Bewusstsein, und physische Zuständen, bei Lebewesen das Gehirn, stehen. Zwei klassische Lösungen wurden hierfür formuliert. Der dualistische Ansatz wird R. DESCARTES zugeschrieben. Er trennt strikt die mentale von der physischen Ebene. Im Gegensatz dazu steht die monistische Perspektive. Diese sieht Geist und Gehirn als eine Einheit und untersucht dessen Interaktion mit der Umwelt [Str95].

Eng mit dem Gehirn-Geist-Problem ist die Frage verknüpft, ob Maschinen Bewusstsein besitzen. Einen aktuellen Ansatz aus der Philosophie liefert hier die Selbstmodell-Theorie der Subjektivität (SMT) nach METZINGER. Im Kern besagt diese, dass ein System, das selbstständig ein internes Modell von sich konstruiert, per se Bewusstsein besitzt. Das Selbstmodell in biologischen Systemen ist in Folge eines „kognitiven Wettrüstens“ entstanden, das für das Überleben notwendig war. METZINGER verknüpft in seiner Theorie Ansätze aus weiteren Disziplinen und überträgt diese teilweise auch auf technische Systeme [Met00], [Met05].

Wie zuvor erwähnt hat der Begriff Kognition seinen Ursprung in der **Psychologie** und drückt dort die Introspektion des Verhaltens aus. Der Kognitivismus nahm eine Gegenposition zum Behaviorismus[13] ein, dem bis in die Mitte des 20. Jahrhunderts anerkann-

[11] MATURANA und VARELA nannten diese nach deren Geburtsstadt Santiago-Theorie. Sie bildet die Grundlage für die biologische Erkenntnistheorie.

[12] Je nach Fokus auch Leib-Seele-Problem oder Körper-Geist-Problem genannt.

[13] Die psychologische Schule des Behaviorismus, abgeleitet von dem amerikanisch-englischen Wort *behavior* für Verhalten, wurde zu Beginn des 20. Jahrhunderts von J.B. WATSON begründet [BE01].

ten wissenschaftstheoretischen Standpunkt in der Verhaltensforschung. Im Behaviorismus werden Organismen und deren Abläufe als Black Boxes beschrieben. Die zwischen eingehenden Reizen und gelernten Reaktionen ablaufenden Prozesse werden nicht erforscht. Folglich ist der Mensch ein Produkt seiner Umwelt, der als *tabula rasa*[14] zur Welt kommt und jegliches Verhalten unabhängig von seinen Erbanlagen erlernt. Einzig die Reiz-Reaktions-Mechanismen stehen daher im Vordergrund [BE01].

Der Ansatz des Kognitivismus hingegen fokussiert die Bedeutung der internen Zustände und Prozesse, die für eine Reaktion auf einen Reiz aus der Umwelt verantwortlich sind. Die interne Repräsentation der Umwelt ist dabei von entscheidender Bedeutung. Dieser Ansatz ist weitestgehend in der kognitiven Psychologie[15] aufgegangen, die die psychischen Vorgänge der menschlichen Kognition untersucht. Der Mensch wird als ein informationsverarbeitendes System gesehen, für das ein Berechnungsmodell aus Reiz, Reaktionen und internen Zuständen gilt [Str96], [And01].

Der Standpunkt der **Kognitionswissenschaft** gleicht dem der kognitiven Psychologie, beschränkt sich aber nicht auf das menschliche Denken. Gegenstand sind somit sämtliche lebende als auch künstliche kognitive Systeme. Ziel ist es, die kognitiv orientierten Ergebnisse aus den verschiedenen Fachdisziplinen methodisch in einen komplementären Einklang zu führen. Hauptaufgaben sind die wissenschaftstheoretische Fundierung, der Entwurf formaler Modelle sowie die empirische Untersuchung der Kognition [Str95]. Grundlage für die kognitive Modellierung ist die Annahme, dass Kognition als Informationsverarbeitung berechenbar sein muss. LENZEN beschreibt prägnant diese These, die als Informationsparadigma bekannt wurde:

> *„Kognitive Prozesse sind Prozesse der Datenverarbeitung, mentale Repräsentationen sind ihre Daten: Das ist die klassische Annahme der Kognitionswissenschaft.“ [Len02, S. 91]*.

Die Kognitionswissenschaft versteht unter Kognition aber nicht nur komplexe geistige Funktionen wie das Denken. Auch die Sensorik und Aktorik dürfen nicht getrennt von der Kognition betrachtet werden. Vielmehr sind sie in weiten Teilen für die Verarbeitung der Informationen mitverantwortlich oder hängen von dieser ab. Hinzu kommen emotionale und motivationale Aspekte ohne die die Kognition in ihrer Gesamtheit nicht erklärbar ist. STROHNER schließt daher auf folgende Definition des Kognitionsbegriffs in der Kognitionswissenschaft:

---

[14] Latein für unbeschriebene Tafel bzw. unbeschriebenes Blatt.

[15] Der Name kognitive Psychologie wurde von U. NEISSER in dem gleichnamigen Buch eingeführt und kennzeichnet die sog. kognitive Wende weg von der behavioristischen hin zur kognitiven Sicht in der Psychologie [Str96], [Nei74].

> *„Kognition ist jede Art von Informationsverarbeitung durch das Zentralnervensystem von Lebewesen oder eine entsprechende Informationsverarbeitung in künstlichen Systemen.“ [Str95, S. 7].*

Das Zentralnervensystem (ZNS) des Menschen, bestehend aus Rückenmark und Gehirn, ist *das interaktive Verarbeitungszentrum für alle Körperfunktionen sowie das Verhalten* [BE01, S. 44ff.]. Zusammen mit dem peripheren Nervensystem bildet es das Nervensystem[16], wobei das periphere die Informationen für das ZNS empfängt und aussendet. Die Aufgaben des ZNS sind die Integration, Koordination und Assoziation der Informationssignale zu einer einheitlichen Leistung. Diese steuert dann die Körpertätigkeit. Das ZNS bildet somit den Kern der kognitiven Informationsverarbeitung, greift aber über diese deutlich hinaus [Sch05], [Str95].

Die Entwicklung der Kognitionswissenschaft weist viele Parallelen zu der der **künstlichen Intelligenz (KI)** auf – sicherlich eine Konsequenz des Informationsparadigmas[17]. Insbesondere die kognitive Modellierung bedient sich der Methoden der KI. Ziel der KI-Forschung ist es, künstliche Systeme zu erschaffen, die mit intelligenten Fähigkeiten ausgestattet sind. In der Regel werden reine Softwaresysteme entwickelt. Die Kernaufgabe besteht darin, rationales, menschenähnliches Denken sowie Handeln zu realisieren. Sie ist ein Teilgebiet der Informatik und bedient sich ihrer Techniken [GN03]. RUSSEL/NORVIG unterscheiden sechs Hauptarbeitsrichtungen, denen die Methoden der KI zugeordnet werden können [RN07, S. 12f.]:

- **Problemlösen**: Hierunter fallen in erster Linie Suchverfahren, wie z.B. die Zustandsraumsuche. Die Hauptaufgabe übernehmen sog. Suchalgorithmen, die den Suchraum nach den gewünschten Merkmalen filtern. Grundsätzlich gibt es blinde (uninformierte) Suchverfahren, zu denen die Breiten-, Tiefen- und bidirektionale Suche gehören, und heuristische (informierte) Suchverfahren, wie bspw. die Greedy- und die A*-Suche [GN03].
- **Wissensrepräsentation und logisches Schließen:** Eine zentrale Annahme der KI ist, dass Wissen logische Schlussfolgerungen (Inferenzen) ermöglicht. Hierdurch kann wiederum Wissen erworben werden. Die zwei grundlegenden Inferenzen sind die Deduktion, die vom Allgemeinen zu Speziellen schließt, und die Induktion, die genau umgekehrt verläuft. Grundlage ist die formale Logik, wie die Prädikatenlogik erster Stufe, oder die Fuzzy-Logik [Kel00].

---

[16] Es handelt sich um eine morphologische Einteilung. Die Funktionen des zentralen und peripheren Nervensystems sind in Wirklichkeit eng miteinander vernetzt. Daher wird meist funktionell das somatische und das vegetative Nervensystem unterschieden [Sch05, S. 687].

[17] Auf eine Darstellung der verzahnten Historie beider Disziplinen wird verzichtet. LENZEN zeigt diese in [Len02] vollständig auf.

- **Planen:** Beim Planen werden geeignete Aktionsfolgen generiert, um ein System in einen gewünschten Zustand zu versetzen. Mithilfe von Planungssprachen wie der Planning Domain Definition Language (PDDL) werden die Planungsprobleme formalisiert. Gerade auf dem Gebiet der Planung konnten in den letzten Jahren große Erfolge erzielt werden [DB09].
- **Unsicherheit:** Eine Idealisierung durch strikte logische Formalisierung stößt im praktischen Einsatz an ihre Grenzen. Hierfür werden Methoden für den Umgang mit Unsicherheiten entwickelt, wie z.B. wahrscheinlichkeitsbehaftete Verfahren, die Theorie von Dempster-Shafer oder die Fuzzy-Logik [GN03].
- **Lernen und Data Mining:** Lernende Systeme werden prinzipiell durch Verfahren des maschinellen Lernens realisierst. Eng damit verbunden ist der Vorgang der Wissensentdeckung und -klassifizierung, dem sog. Data Mining. Hier steht der Umgang mit großen Datenmengen im Vordergrund.
- **Kommunizieren, Wahrnehmen und Handeln:** RUSSEL/NORVIG fassen die Bereiche der Kommunikation, Perzeption und Handlung bzw. Handlungsplanung zusammen. Konkrete Forschungsfelder sind bspw. die Spracherkennung für die Kommunikation, das Bildverstehen für die Wahrnehmung und die kognitive Robotik für die Handlung [GN03].

Bei der theoretischen Fundierung der KI ist in erster Linie die uneinheitliche Definition des Intelligenzbegriffs[18] hinderlich. Die KI-Pioniere A. NEWELL und H. SIMON schlugen bereits eine Namensänderung in „komplexe Informationsverarbeitung“ vor, die sich aber nicht durchsetzen konnte [Bor94, S. 115]. Die Folge war, dass innerhalb der KI-Bewegung zwei gegensätzliche Thesen definiert wurden. Vertreter der *starken KI* sind der Überzeugung, dass Maschinen dieselbe Intelligenz haben können, die bislang nur bei Menschen nachweisbar ist. Denken und Bewusstsein sind demnach reine Rechenprozeduren. Einige KI-Visionäre glauben sogar, dass die Rechenleistung zukünftiger Computer genügen wird, um das menschliche Gehirn durch Simulatoren aus Silizium operativ zu ersetzen und der Mensch auf seine sterbliche „Hülle“ verzichten könnte. Die *schwache KI* hingegen zielt auf eine anwendungsspezifische Simulation der Intelligenz ab. Künstliche Systeme dienen eher als hilfreiches Werkzeug zur Erforschung der menschlichen Denkvorgänge als zu deren Nachbildung [Bro02], [RN07].

Die kognitive Robotik ist ein Ansatz aus der KI-Forschung. Sie hat den Anspruch kognitionsrelevante Ergebnisse aus den hier vorgestellten Disziplinen in die **Technik** zu überführen. Ziel sind Mechanismen zur Erhöhung der Autonomie und zur Gewinnung von adaptiven Verhalten von Robotern. Autonomie beschreibt die Fähigkeit eines Ro-

[18] Eine oft zitierte Definition hat der Erfinder des Intelligenzquotienten W. STERN aufgestellt: *„Intelligence is a general capacity of an individual consciously to adjust his thinking to new requirements: it is general mental adaptability to new problems and conditions of life.“ [Ste12, S. 3]*.

boters, sein Verhalten selbst zu bestimmen oder wenigstens sein aktuelles Verhaltensmuster per Interaktion mit der Umgebung entsprechend zu ergänzen, zu verändern oder es gänzlich durch ein neues auszutauschen. Adaptivität in der Robotik meint die Fähigkeit des Roboters, sich mittels eigenen kognitiven und motorischen Funktionen einer dynamischen Umwelt anzupassen und diese zielgerichtet zu beeinflussen (vgl. Kap. 2.2.2) [Chr99], [KC03, S. 31].

Eine systematische Integration kognitiver Funktion in technische Systeme so wie sie im Fokus dieser Arbeit steht, stellt sich aber bislang weitaus schwieriger dar. LENGELER et al. führen hierfür den Begriff des *mechatronischen Kurzschlusses* ein, der ausbleibt bzw. nicht trivial umzusetzen ist:

> *„Auch Forschungsaktivitäten im Bereich der Künstlichen Intelligenz bzw. Kognitiven Robotik […], die einen technischen rekonstruktiven Anspruch erheben, haben zwar gelegentlich zu einer Präzisierung der Problemstellungen geführt, aber kaum Durchbrüche erzielt. Dies hängt möglicherweise grundsätzlich damit zusammen, daß ein technischer Erfolg nicht ohne weiteres durch einen mathematisch-informatischen bzw. mechatronischen "Kurzschluß", d.h. ohne genaue Berücksichtigung der Struktur- und Funktionsübergänge zwischen den verschiedenen Organisationsebenen von Lebewesen […] erreichbar ist." [LMP99, S. 7f.]*.

Kognition in technischen Systemen scheint daher mehr eine softwaretechnische Zielrichtung als eine bereits realisierbare Eigenschaft. Es fehlen Beweise für die Gültigkeit des Informationsparadigmas und somit für die Analogie zwischen menschlichem Geist und berechenbarer Computerumsetzung. Die Grundlage für die vollständige Realisierung der Kognition in der Technik fehlt [Pri98, S. 6f.]. Insbesondere die Unterscheidung, ob ein (technisches) System als kognitiv bezeichnet werden kann, ist aufgrund der ungenauen Abgrenzung des Kognitionsbegriffs nicht möglich [Str96]. Die Notwendigkeit einer technischen Reaktivierung der Kognition lässt sich aber auch nicht von der Hand weisen. MÜLLER führt hierfür den Begriff *Beweltigung* ein:

> *„Weil ohne Kognition die Komplexität der Umwelt und der darin zu lösenden Aufgaben für autonome Systeme wie Menschen, Tiere, Roboter nicht zu bewältigen wäre. Kognition ist der Mittel- und Verfahrensvorrat zur Bewältigung der Welt (Beweltigung)." [Mül98, S. 7]*.

Daher ist es nicht das primäre Ziel der hier zu erarbeitenden Entwicklungssystematik, den Entwurf eines kognitiven Systems zu unterstützen, das über das ganze Repertoire der Kognition verfügt. Vielmehr steht die ingenieurwissenschaftliche Extension der Informationsverarbeitung mechatronischer Systeme um kognitive Funktionen im Mittelpunkt. Diese kann aber nur unter Berücksichtigung der hier vorgestellten Sichten erfolgen.

2.3.2 Grundlagen kognitiver Informationsverarbeitung

Zwar fehlt eine einheitliche Definition der Kognition, doch ist in der Wissenschaft anerkannt, dass sie in irgendeiner Art und Weise *zwischen der Reizaufnahme und dem Verhalten interveniert* [Str96, S. 2]. Sie ist all das, was zwischen dem Erhalt von Informationen aus der Umgebung und der entsprechenden Aktion eines Systems stattfindet. Sie steht somit für die **kognitive Informationsverarbeitung**. Ein entscheidender Aspekt der Kognition ist, dass sie auf systeminternes Wissen zurückgreift und neues Wissen abspeichert. Dabei ist sie eng mit der Peripherie eines Systems verzahnt. So sind die Weiterverarbeitung sensorischer Daten als auch die Planung und Steuerung von Bewegungen Bestandteile der kognitiven Informationsverarbeitung [Str95], [Str96]. **Kognitive Funktionen** sind die Grundlage für die Umsetzung einer kognitiven Informationsverarbeitung. In der Neurophysiologie sind sie folgendermaßen definiert:

> *„Unter kognitiven Funktionen verstehen wir alle bewussten und nicht bewussten Vorgänge, die bei der Verarbeitung von organismusexterner oder -interner Informationen ablaufen, z.B. Verschlüsselung (Kodierung), Vergleich mit gespeicherter Information, Verteilung der Information, Problemlösung und Entschlüsselung und sprachlich-begriffliche Äußerung." [BS06, S. 449]*.

Kognitive Funktionen umfassen also alle Vorgänge, die mit der Aufnahme von Informationen, ihrer Verarbeitung und Speicherung im Gedächtnis sowie ihrer Nutzung und Anwendung verbunden sind. Auf psychologischer Ebene können beim Menschen folgende kognitive Funktionen identifiziert werden, die der vorgehenden Definition gerecht werden [Str96, S. 1]:

- das Wahrnehmen und Erkennen,
- das Enkodieren,
- das Speichern und Erinnern,
- das Denken und Problemlösen,
- das Steuern der Motorik und
- der Gebrauch von Sprache.

Die Umsetzung dieser Funktionen kann erheblich variieren, so kann eine einfache Sprachfunktion bereits durch das Unterscheiden von „ja" und „nein" realisiert werden und nicht erst durch den Gebrauch natürlicher Sprache [SP05]. In jedem Fall sind die kognitiven Funktionen für eine flexible Kopplung zwischen Wahrnehmung und Handlung verantwortlich. Systeme die diese Funktionen in ihrer Gesamtheit besitzen, werden als **kognitive Systeme** bezeichnet. STRUBE leitet aus den kognitiven Funktionen drei spezifische Merkmale für kognitive Systeme ab [GRS03, S. 19]:

- Aktive Einbindung in die Umgebung und die Fähigkeit mit ihr Informationen auszutauschen.
- Flexible und umgebungsadaptive Handlungssteuerung durch die Repräsentation systemrelevanter Informationen der Umwelt.
- Lern- und Antizipationsfähigkeit der integrierten Informationsverarbeitung.

Das Bild 2-11 veranschaulicht die Abgrenzung zwischen kognitiven und nicht kognitiven Systemen aufgrund ihrer Informationsverarbeitung zwischen der Reizaufnahme und der Aktionsausführung.

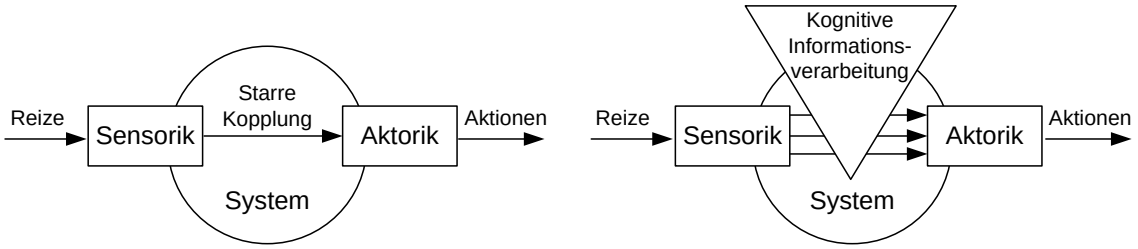


Bild 2-11: Gegenüberstellung eines nicht kognitiven Systems und eines kognitiven Systems nach [Str98, S. 6]

Kognitive Systeme sind nicht nur in der Lage nach fest vorimplementierten Verhaltensmustern auf Reize zu reagieren, sondern können zudem die Kopplung zwischen sensorischer Eingabe und aktorischer Ausgabe modifizieren. Die kognitive Informationsverarbeitung ermöglicht eine flexible und intelligente Anpassung des systemeigenen Verhaltens entsprechend der subjektiv wahrgenommen externen sowie internen Zuständen [Str98].

Die Veränderung des Systemverhaltens aufgrund der kognitiven Informationsverarbeitung geht immer mit dem Phänomen **Lernen** einher [Str06, S. 9]. Lernen ist ein Prozess der Verhaltensanpassung basierend auf dem Erwerb von neuem Wissen oder der Umstrukturierung von bereits vorhandenem Wissen [ZG04, S. 243]. JÖRGER stellt den Lernprozess schematisch als Flussdiagramm dar, dem er entsprechende kognitive Funktionen zuordnet (vgl. Bild 2-12) [Jör89]. Das Flussdiagramm ist von links nach rechts zu lesen. Der Input ist der Reiz aus der Umwelt, der im sensorischen Register wahrgenommen und anschließend im Kurzzeitspeicher als Information verarbeitet wird. Bei erfolgreicher Verarbeitung und Kategorisierung mit bestehenden Informationen, können neue Informationen im Langzeitspeicher abgelegt werden (Enkodierung). Auf deren Basis können problemspezifische Modifikationen des Systemverhaltens im Sinne einer kognitiven Informationsverarbeitung geplant werden (Dekodierung). Der jeweilige Vorgang des Zuordnens und Abrufens von Informationen bzgl. der vorhandenen Informationen und Strukturen wird subjektive Organisationstendenz genannt. Die Umsetzung in der Umwelt und das Ergebnis des Lernprozesses zeigen sich in der Leistung [Kel00], [ZG04].

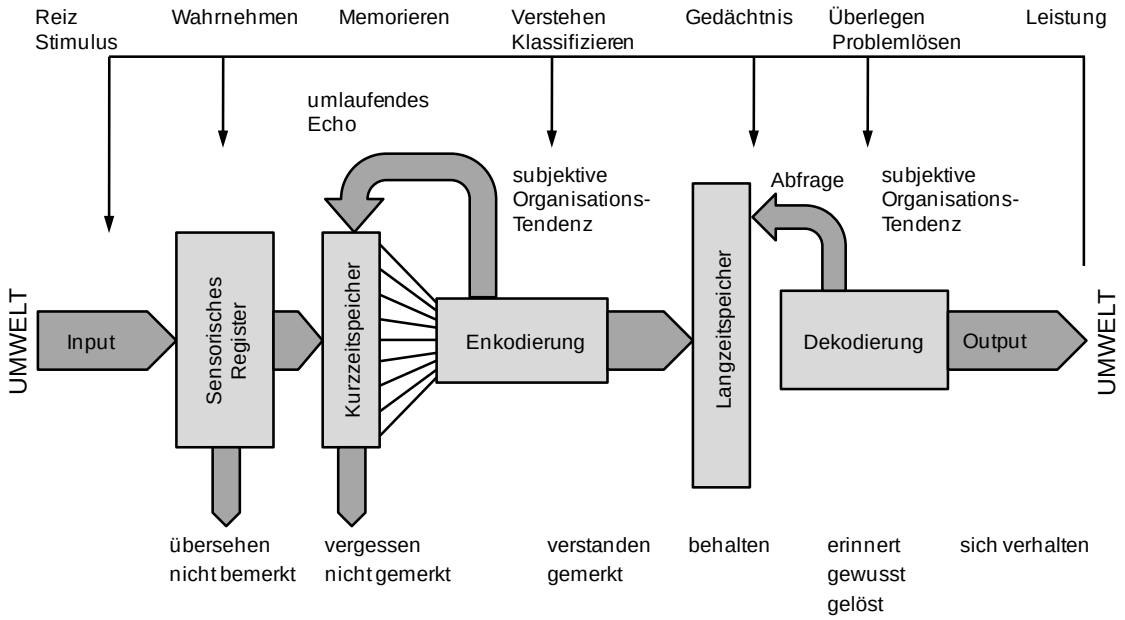


Bild 2-12: Lernprozess als Flussdiagramm nach [Kel00, S. 365]

Nicht-kognitive Systeme sind nicht lernfähig und können nur fest vorprogrammiert werden. Von entscheidender Bedeutung ist, dass die kognitive, lernfähige Informationsverarbeitung die reaktiven Kopplungen nicht vollständig ersetzt, sondern mit diesen koexistieren muss. Reaktive Kopplungen sind deswegen so wichtig, weil die meisten existentiellen Systemmechanismen rein reaktiv und reflexartig ablaufen. STRUBE schlägt daher das **Dreischichtenmodell für die Verhaltenssteuerung** lernfähiger, kognitiver Systeme vor. Den jeweiligen Schichten können unterschiedliche Lernformen zugeordnet werden, die in ihrer Gesamtheit eine kognitive Informationsverarbeitung charakterisieren (Bild 2-13) [Str98].

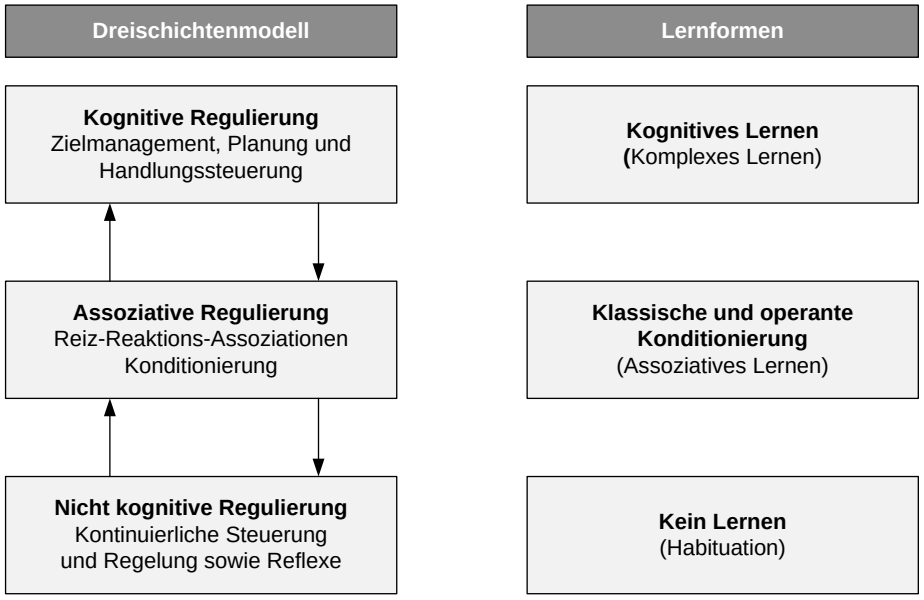


Bild 2-13: Dreischichtenmodell für die Verhaltenssteuerung nach STRUBE [Str98, S. 9] und Zuordnung der Lernformen nach [SP05]

Die unterste Ebene des Dreischichtenmodells umfasst die **nicht kognitive Regulierung**. Aufgrund der starren Kopplung zwischen der Sensorik und der Aktorik findet kein Lernen bzw. nur das Lernen, bestimmte Reize zu ignorieren, statt. In der Lernpsychologie wird dies als *Habituation* oder Gewöhnung bezeichnet und zählt teilweise als einfachste Form des Lernens. Ihr Ziel ist die Verhinderung einer Reizüberflutung und die Erhöhung der Aufmerksamkeit [Ede00]. Hier laufen einfache kontinuierlich arbeitende Steuerungen, rückgekoppelte Regelkreise und Reflexe ab. Dabei ist der Aktionsraum der jeweiligen Steuer- und Regelungseinheiten begrenzt.

Der Lernprozess in der mittleren Schicht, der **assoziativen Regulierung**, basiert auf Reiz-Reaktions-Assoziationen und erfolgt durch klassische oder operante Konditionierung[19]. Beide Lernformen zählen zum *assoziativen Lernen*. Die klassische Konditionierung geht auf den Psychologen PAVLOV zurück. Es handelt sich um eine Art des Lernens, bei der eine unkonditionierte Reaktion durch einen konditionierten Reiz hervorgerufen wird, der seine Wirkung durch Assoziation mit einem unkonditionierten Reiz erlangte. THORNDIKE zeigte zudem, dass sich diese Reiz-Reaktions-Assoziationen durch Wiederholung und positive Bestätigung verstärken lassen[20]. Beim operanten Konditionieren[21] nach SKINNER wird die Auftretenswahrscheinlichkeit des Verhaltens durch die darauf folgenden Konsequenzen bestimmt. Wird das Verhalten verstärkt, tritt es zukünftig wiederholt auf. Folgt hingegen eine Bestrafung, wird es zukünftig unterlassen. So können komplexe Verhaltensweisen aus elementaren Lernschritten aufgebaut werden [ZG04, S. 246ff.].

Auf der obersten Schicht des Modells befindet sich die **kognitive Regulierung**. Sie realisiert eine zielorientierte und planende Verhaltenssteuerung. Die entsprechende Lernform ist das *kognitive Lernen*. Diese bezieht sich auf die Konstruktion von Wissen, die Herausbildung spezifischer Fähigkeiten und das schlussfolgernde Denken und Urteilen. Im Unterschied zur Konditionierung müssen weder die eigene Reaktion noch die Konsequenzen direkt wahrgenommen werden. Kognitives Lernen ermöglicht es, Wissen in neuartigen Situationen anzuwenden und Probleme flexibel zu lösen [See03].

Das vorgestellte Dreischichtenmodell verdeutlicht die hohen Anforderungen an die Umsetzung einer kognitiven Informationsverarbeitung. Insbesondere durch die Verwendung der Techniken der KI, die Verfahren zur Integration kognitiver Funktionen bereit-

---

[19] Konditionierung ist die Art und Weise, wie Ereignisse, Reize (Stimuli) und Verhalten miteinander assoziiert werden. Es handelt sich dabei um Reiz-Reaktions-Theorien, die das Lernen als Kopplung eines bestimmten Umweltreizes mit einer Verhaltensweise definieren [ZG04].

[20] Für das Lernverhalten, welches dem sog. „Lernen durch Versuch und Irrtum“ entspricht, definierte THORNDIKE das „Law of Effect“, das im Kern besagt, dass eine Handlung entsprechend einer positiven oder negativen Konsequenz wieder bzw. nicht mehr ausgeführt wird [Pri98].

[21] Das operante und instrumentelle Konditionieren unterscheiden sich nur marginal und können im Rahmen dieser Arbeit als identisch angesehen werden.

stellt, erhöht sich die Komplexität. Hier wird ein einheitliches Strukturkonzept für technische Systeme benötigt.

2.3.3 Herausforderungen bei der Integration kognitiver Funktionen

Zahlreiche Herausforderungen ergeben sich bei der Integration kognitiver Funktionen in technische Systeme. Diese sind insbesondere auf die interdisziplinäre Herangehensweise bei der Entwicklung und die Komplexität des zu entwickelnden Systems zurückzuführen. Beide Aspekte wurden in den vorhergehenden Abschnitten analysiert. Doch gerade die Systemkomplexität ist noch nicht vollständig durchdrungen. RIEGLER identifiziert vier wesentliche Herausforderungen [Rie07, S. 3ff.]:

- **PacMan:** Technische Systeme sind PacMan-Systeme, die nur darauf ausgerichtet sind, bestimmte Parameter ohne Berücksichtigung des dahinter stehenden Ziels zu optimieren. Sie interagieren nur mit vom Systementwickler vordefinierten Objekten. Ihre Bedeutung kennen sie nicht; noch sind sie in der Lage, diese herauszufinden.
- **Black-Box:** Technische Systeme benötigen ein internes Modell ihrer Umgebung und ihrer selbst, um nicht in der Black-Box-Darstellung des Behaviorismus gefangen zu bleiben. Was die Psychologie im Rahmen des Kognitivismus klar gestellt hat, dass der Mensch mehr als ein Reiz-Reaktions-System ist, ist in der Technik bislang nicht angekommen.
- **Binarität (Symbol-Grounding):** Die symbolorientierte Informationsverarbeitung in technischen Systemen trifft in der Regel binäre Klassifikationen. Sie wird nicht dem Umstand gerecht, dass sich die reale Systemumgebung kontinuierlich ändert und sich durch einen gewissen Grad an Unschärfe auszeichnet. Das technische System kann zwar somit die Bedeutung bestimmter Symbole lernen, aber nicht im Betrieb mit diesen neue Assoziationen auch hinsichtlich der Umgebung ziehen.
- **Anthropomorphe Sicht:** Bisherige Verfahren, wie z.B. künstliche neuronale Netze, sind zwar biologisch fundiert, jedoch kann deren kognitive Leistung anfangs nur schwer und bei steigender Komplexität kaum noch nachvollzogen werden. Rückschlüsse werden auf Basis des beobachteten Verhaltens getroffen, die dann in Relation zu dem (gewünschten) menschlichen Verhalten gesetzt werden. Die innere, kognitive Funktionsweise des Systems tritt in den Hintergrund.

Aus diesen Herausforderungen und der notwendigen interdisziplinären Vorgehensweise ergeben sich nach RIEGLER vier Grenzbedingungen für die Integration kognitiver Funktionen in technische Systeme [Rie07, S. 12f.]:

- **Evolution:** Kognition beschreibt die Fähigkeit, dass ein System durch Organisation und Evolution von komplexen Verhaltensstrukturen seine Umwelt bewältigt und sich in dieser selbst erhält. Diese Fähigkeit kann daher nicht vollständig vordefiniert werden, sondern muss sich teilweise eigenständig weiterentwickeln können.

- **Hierarchische Struktur:** Die Entwicklung der Kognition ist ein inkrementaler Bottom-up Prozess, der auf der reaktiven sensor-motorischen Ebene beginnt. Jede höher entwickelte Ebene baut auf der vorherigen Ebene auf. Die verschiedenen Funktionen zur Umsetzung der Kognition verteilen sich auf den unterschiedlichen Ebenen. Es ergibt sich eine streng hierarchische Form. Diese führt jedoch nicht zu getrennt arbeitenden Modulen, sondern ist eine Voraussetzung für ein robustes, vorhersehendes und lernendes Verhalten.
- **Geschlossenheit:** Die kognitive Informationsverarbeitung sollte als geschlossenes System realisiert sein, denn das System selbst kann nicht zwischen internen und externen Zuständen unterscheiden. Dies setzt voraus, dass dessen Prozesse nicht in semantischer Darstellung und Abhängigkeit der Umwelt definiert werden, wie dies bei physikalischen Systemen der Fall ist.
- **Untersuchbarkeit:** Weil kognitive Prozesse auf einem abstrakten und funktionalen Level erklärbar sein müssen, stehen konnektionistische Ansätze, wie neuronale Netze, nicht zur Auswahl. Hierarchisch organisierte Ansätze hingegen können aufgrund ihrer linearen Zerlegbarkeit schichtweise entkoppelt und untersucht werden.

2.4 Entwicklung fortgeschrittener mechatronischer Systeme

Im Folgenden wird in Kapitel 2.4.1 mit der VDI-Richtlinie 2206 eine etablierte Methodik zur Entwicklung mechatronischer Systeme vorgestellt. Sie fasst den minimalen Konsens der Fachwelt zusammen. Anschließend wird die Entwicklung selbstoptimierender Systeme im SFB 614 beschrieben (Kap. 2.4.2). Die dort entstehende Entwicklungsmethodik stellt eine Weiterführung der VDI-Richtlinie 2206 für fortgeschrittene mechatronische Systeme dar. Komplettiert wird das Kapitel durch eine grundlegende Darstellung des in der VDI-Richtlinie 2206 bezeichneten Teilschritts des Systementwurfs in Kapitel 2.4.3. Hierdurch wird das Handlungsfeld der hier zu erarbeitenden Entwicklungssystematik aufgezeigt.

2.4.1 Entwicklung mechatronischer Systeme

Die steigende Systemheterogenität und die Beteiligung mehrerer Fachdisziplinen sowie der damit einhergehende Abstimmungsaufwand sind die wesentlichen Herausforderungen bei der Entwicklung mechatronischer Systeme[22]. Bestehende und etablierte Vorgehen zum Entwurf technischer Systeme, wie z.B. die Mechanik-Entwicklung nach PAHL/BEITZ [PBF+07], werden diesem Anspruch nicht gerecht. Die VDI-Richtlinie 2206 „Entwicklungsmethodik mechatronischer Systeme“ [VDI2206] beschreibt ein

[22] Im Fokus dieser Arbeit stehen mechatronische Systeme der 2. Klasse (vgl. 2.2.1). Systeme der 1. Klasse werden maßgeblich vom Produktionsprozess determiniert. GAUSEMEIER/FELDMANN schlagen hierfür ein Vorgehen in Anlehnung an die VDI-Richtlinie 2206 vor [GF06, S. 118].

generelles, fachdisziplinübergreifendes Vorgehen zur Entwicklung mechatronischer Systeme. Sie versteht sich als Ergänzung zu den VDI-Richtlinien 2221 „Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“ und VDI-Richtlinie 2422 „Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik“ [VDI2221], [VDI2422]. Sie beschreibt ein flexibles Vorgehen, das drei elementare Bestandteile umfasst [VDI2206, S. 26ff.]:

1) **Problemlösungszyklus auf der Mikroebene:** Der Problemlösungszyklus auf der Mikroebene unterstützt den Entwickler bei der Bewältigung vorhersehbarer aber auch unerwarteter Probleme. Er wird in Anlehnung an das Systems Engineering als Abfolge elementarer Schritte verstanden [DH02, S. 47ff.]. Beispiele für solche elementaren Schritte sind Situationsanalyse, Lösungssynthese, Analyse, Bewertung und Entscheidung. Die Zusammenstellung der Schritte lässt sich flexibel an die jeweilige Entwicklungsaufgabe anpassen.

2) **Das V-Modell auf der Makroebene:** Das Vorgehen auf übergeordneter Ebene greift das in der Softwaretechnik etablierte V-Modell auf [BD95]. Es wurde an die Anforderungen der Entwicklung mechatronischer Systeme angepasst. Je nach Komplexität der Aufgabe sind die Phasen mehrmals zu durchlaufen (Bild 2-14):

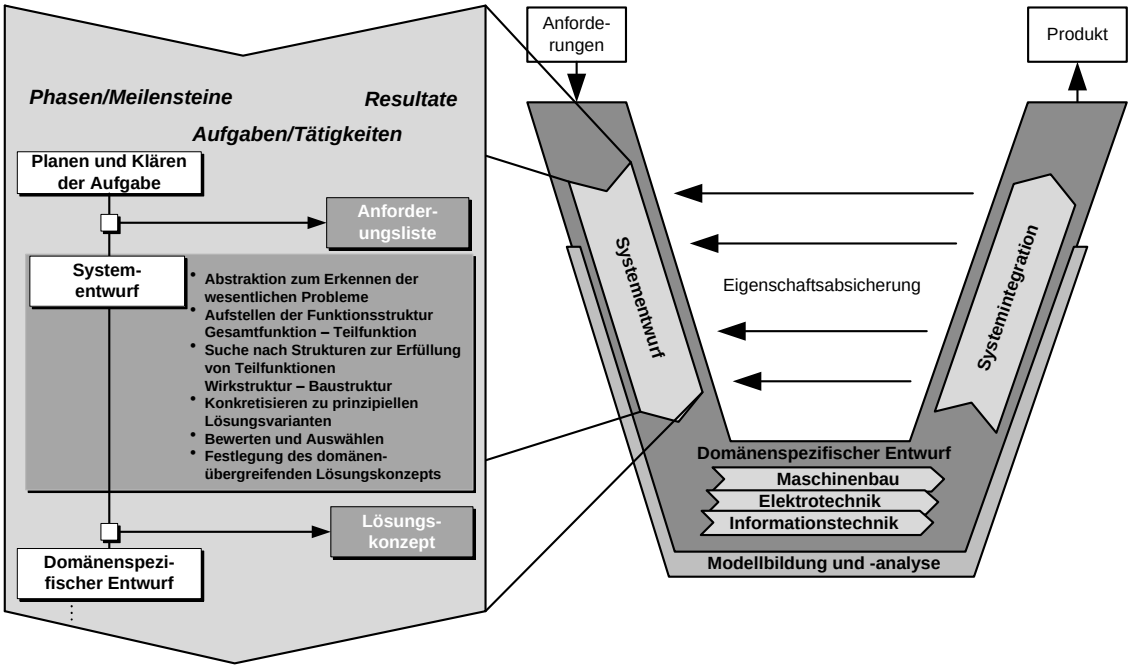


Bild 2-14: Das V-Modell als Makrozyklus (rechts) und der Prozessbaustein für den Systementwurf nach [VDI2206, S. 32]

- **Anforderungen:** Für einen konkreten Entwicklungsauftrag sind die Anforderungen an das Produkt in Form einer Anforderungsliste zu definieren. Anhand dieser erfolgt die Bewertung des späteren Produkts.

- **Systementwurf:** An dieser Stelle wird die wesentliche physikalische und logische Wirkungsweise des Systems fachdisziplinübergreifend spezifiziert. Ergebnis ist das Lösungskonzept.
- **Domänenspezifischer Entwurf:** Es folgt die Konkretisierung des Lösungskonzepts. In der Regel arbeiten die Fachdisziplinen[23] zu diesem Zeitpunkt parallel zueinander. Detaillierte Berechnungen und Auslegungen sollen das Lösungskonzept sicherstellen.
- **Systemintegration:** Die fachdisziplinspezifisch erarbeiteten Ergebnisse werden zu einer Gesamtlösung zusammengeführt. Diese wird auf korrektes Zusammenwirken überprüft.
- **Eigenschaftsabsicherung:** Während der Systemintegration werden die Ergebnisse an den Anforderungen und dem Lösungskonzept überprüft.
- **Modellbildung und -analyse:** Fortlaufend werden die Systemeigenschaften rechnergestützt modelliert und analysiert.

3) **Prozessbausteine für wiederkehrende Arbeitsschritte**: Wiederkehrende Tätigkeiten während der Teilschritte des V-Modells können in Prozessbausteinen hinterlegt werden. Die VDI-Richtlinie 2206 definiert Prozessbausteine für die Teilschritte Systementwurf, Modellbildung und -analyse, domänenspezifischer Entwurf, Systemintegration und Eigenschaftsabsicherung. Bild 2-14 zeigt den Prozessbaustein für den Systementwurf und die darin definierten Tätigkeiten.

2.4.2 Entwicklung selbstoptimierender Systeme

Im SFB 614 wurden die Entwicklungsprozesse für die verschiedenen selbstoptimierenden Demonstratoren aufgenommen und mit der Prozessmodellierungssprache OMEGA[24] modelliert. Diese umfassen die durchzuführenden Tätigkeiten, deren Ergebnisse in Form von Entwicklungsobjekten sowie einzusetzende Methoden und Werkzeuge. Der gesamte Entwicklungsprozess des Demonstrators RailCab bspw. umfasst ca. 850 Prozessschritte und fast 1000 Entwicklungsobjekte [GRD+09], [KGD10], [DK10]. Nach einer ausgiebigen Analyse konnte ein generischer Entwicklungsprozess sowie ein generisches Vorgehen für die frühen Phasen der Entwicklung, der sog. Konzipierung, formuliert werden [GFD+08b].

---

[23] Im Rahmen dieser Arbeit sind die Begriffe Fachdisziplin und Domäne synonym.

[24] OMEGA (Objektorientierte Methode zur Geschäftsprozessmodellierung und -analyse) ist eine Methode zur vollständigen Modellierung einer Ablauforganisation in einem Unternehmen. Mit ihr können Leistungserstellungsprozesse, wie bspw. Entwicklungsprozesse visualisiert werden [GPW09, S. 281ff.].

Der generische **Entwicklungsprozess** selbstoptimierender Systeme ist mit dem vorgestellten V-Modell der VDI-Richtlinie 2206 vergleichbar. Der wesentliche Unterschied besteht darin, dass eine explizite Integrationsphase nach der fachdisziplinspezifischen Ausarbeitung entfällt. Vielmehr wird diese als integraler Bestandteil dieser Phase verstanden. In der Folge ergibt sich ein generischer Entwicklungsprozess bestehend aus zwei Phasen (Bild 2-15).

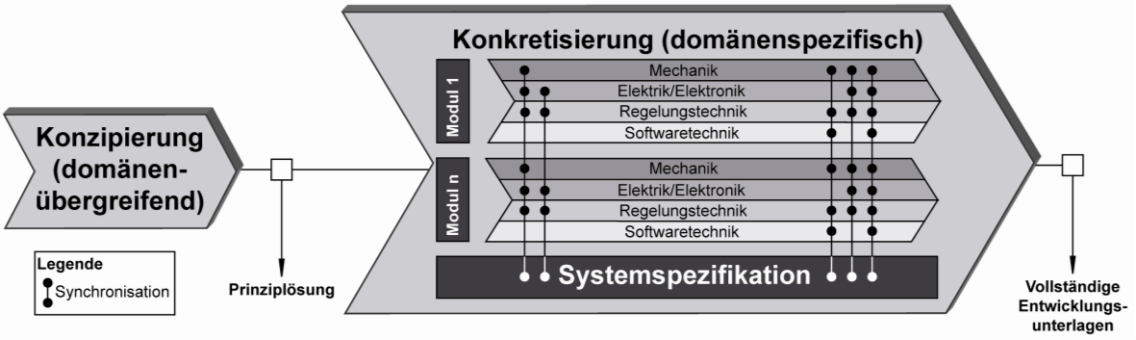


Bild 2-15: Entwicklungsprozess selbstoptimierender Systeme nach [GSD+09]

In der domänenübergreifenden **Konzipierung** erarbeitet ein interdisziplinäres Team aus Entwicklern aller beteiligten Fachdisziplinen die sog. Prinziplösung[25] des zu entwickelnden Systems. Diese beschreibt die grundsätzliche Lösung für eine Entwicklungsaufgabe. Ferner legt diese die physikalische und logische Wirkungsweise und Anordnung der benötigten Systembestandteile fest. Diese müssen hierfür noch nicht detailliert ausgestaltet sein [VDI2221], [PBF+07], [Fra06]. Um die Komplexität des Gesamtsystems zu beherrschen, erfolgt zu diesem Zeitpunkt eine erste Aufteilung in Module. Ziel ist eine entwicklungsorientierte Produktstruktur, die die in der Modularisierung übliche funktionsorientierte und gestaltorientierte Sicht vereint. Zu diesem Zweck müssen nicht nur die Eigenschaften und Beziehungen der möglichen Systembestandteile, sondern auch das resultierende Systemverhalten betrachtet werden. Auf diese Weise können für die identifizierten Module selbst Prinziplösungen konzipiert werden, die bei Bedarf auswechselbar sind [Ste07], [GSD+09].

Die Prinziplösung und die darin festgelegte Produktstruktur sind die Grundlage für die domänenspezifische **Konkretisierung**. Das System wird nun schrittweise ausgestaltet, modelliert, simuliert, sowie analysiert. Schließlich werden die vollständigen Entwicklungsunterlagen erstellt. Die bis dahin weitestgehend sequentielle Entwicklung wird nun entsprechend der spezifizierten Module in parallele Entwicklungsstränge aufgeteilt. Ferner werden diese Modulstränge durch die beteiligten Domänen entwickelt. Diese treiben die Systementwicklung mit ihren domänenspezifischen Methoden und Werkzeugen voran. Ein übergeordneter Entwicklungsstrang, der die domänenübergreifende Systemspezifikation aus der Prinziplösung weiterführt, sichert durch gezielte, aber kon-

[25] Synonyme Begriffe in der Entwicklungsmethodik sind prinzipielle Lösung [PBF+07] oder Lösungskonzept [VDI2206], [Ehr07].

tinuierliche Synchronisation der Entwicklungsergebnisse die Gesamtintegration ab. Die Abstimmung der beteiligten Domänen muss modulintern als auch modulübergreifend erfolgen.

Im Vordergrund der Arbeiten des SFB 614 standen aus entwicklungsmethodischer Sicht bislang die frühen Phasen der Entwicklung. Da sich die vorliegende Arbeit dort einordnet, wird im Folgenden das detaillierte Vorgehensmodell für die Konzipierung nach GAUSEMEIER et al. kurz erläutert (Bild 2-16).

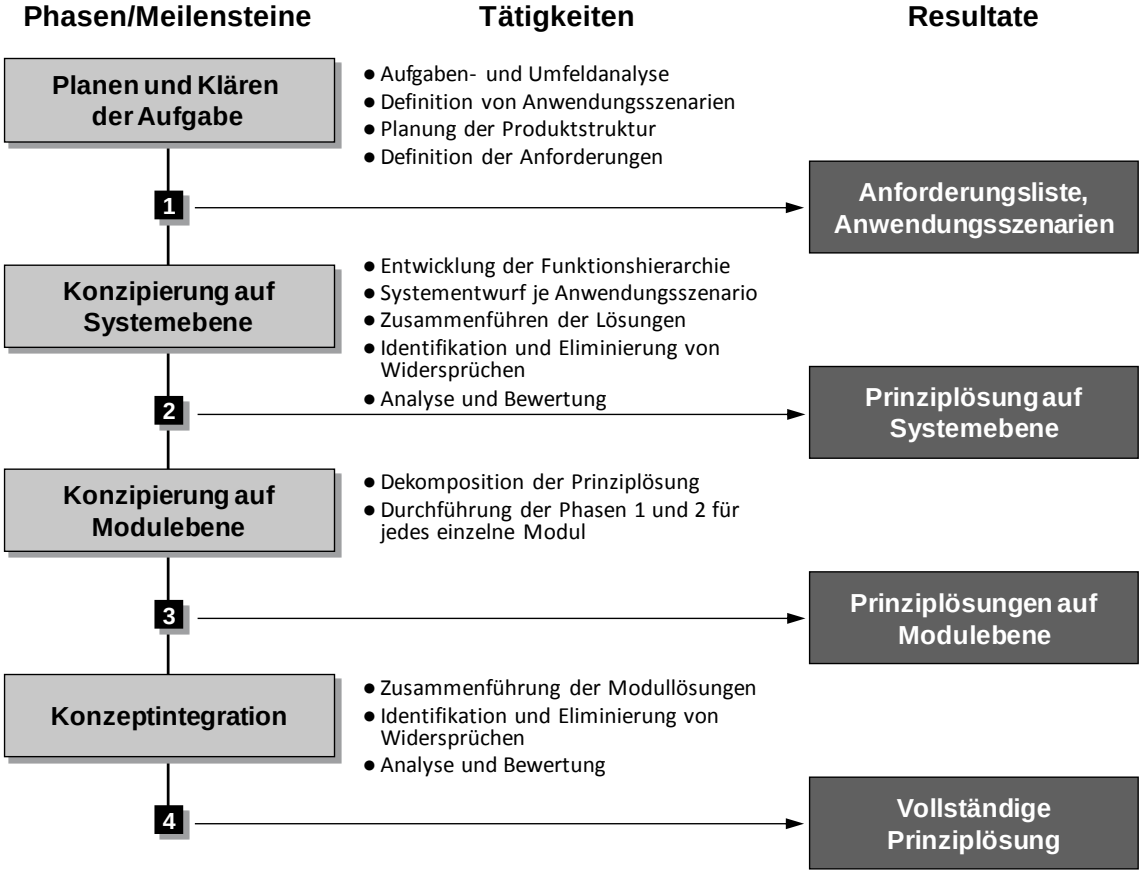


Bild 2-16: Vorgehensmodell für die Konzipierung selbstoptimierender Systeme nach [GFD+08b, S. 97]

Das **Vorgehensmodell zur Konzipierung** besteht aus vier grundlegenden Phasen und beschreibt deren idealisierten Ablauf inkl. auszuführender Tätigkeiten und Resultate. Das Vorgehen ist dabei nicht als stringente Folge zu verstehen, sondern enthält in der Anwendung zahlreiche Iterationen in Folge des Entwurfsobjekts, den organisatorischen Randbedingungen und des individuellen Problemlösungsvorgehens des Entwicklers [GFD+08b, S. 96f.], [ADG+09, 167ff.], [SFB08, S. 411ff.].

In der Phase **Planen und Klären der Aufgabe** wird zunächst die Aufgabe auf die wesentliche Entwicklungsaufgabe abstrahiert und ein Umfeldmodell des zu entwickelnden Systems erstellt. Dieses beschreibt das System als eine Black-Box und spezifiziert nur Elemente aus der Umgebung des Systems und deren Einflüsse auf das System. Die

Kombination konsistenter Einflüsse resultiert in Situationen, für die das gewünschte Systemverhalten in Anwendungsszenarien zunächst prosaisch beschrieben wird. Mit dem Verfahren zur Produktstrukturierung nach STEFFEN wird die Produktstruktur geplant und hierfür relevante Entwurfsregeln identifiziert [Ste07]. Abschließend werden die Ergebnisse als funktionale und nicht-funktionale Anforderungen[26] sowie Wünsche in der Anforderungsliste festgehalten.

Ausgehend von den zuvor ermittelten Anforderungen wird während der **Konzipierung auf Systemebene** eine lösungsneutrale funktionale Systembeschreibung in Form einer Funktionshierarchie aufgestellt. Für jedes zuvor aufgestellte Anwendungsszenario erfolgt dann die Entwicklung einer Lösung. Diese startet mit einer Modifikation der Funktionshierarchie hinsichtlich des jeweiligen Anwendungsszenarios. Im Anschluss werden für die Teilfunktionen auf der untersten Ebene der Funktionshierarchie Teillösungen gesucht und in einen morphologischen Kasten eingetragen. Teillösungen können entweder auf eine Lösung abstrahierte Muster oder finale Lösungselemente sein (vgl. Kap. 2.4.3). Eine widerspruchsfreie Kombination der Teillösungen im morphologischen Kasten bildet eine Lösungsvariante [Köc04]. In der Regel werden mehrere Lösungsvarianten abgeleitet, von denen die vielversprechendsten weiter konkretisiert werden. Hierfür werden auf Basis der ausgewählten Teillösungen die Wirkstruktur und ein erstes grobes Modell der Gestalt erstellt. Zusätzlich wird das Verhalten des zu entwickelnden Systems in Form von Aktivitäten und Zuständen spezifiziert. Dieses ist die Grundlage für das Vorausdenken der möglichen Systemziele und somit für die Modellierung des internen Zielsystems. Im Anschluss sind die für die einzelnen Anwendungsszenarien entwickelten Lösungen zusammenzuführen und das Ergebnis auf Widersprüche zu untersuchen. Stellt sich heraus, dass sich Widersprüche in der Systemkonfiguration ergeben, die situations- und zielabhängig sind, ergibt sich ein Potential für das Wirkparadigma der Selbstoptimierung. Hierfür muss ein entsprechendes Konzept spezifiziert werden, das den Selbstoptimierungsprozess enthält [GZF+07]. Abschließend erfolgt eine Analyse und Bewertung, die zur Prinziplösung auf Systemebene führt.

Gemäß der geplanten Produktstruktur erfolgt in der Phase **Konzipierung auf Modulebene** eine Dekomposition des Gesamtsystems in die einzelnen Module. Für jedes Modul wird dann wiederum eine Prinziplösung entwickelt. Dabei sind die gleichen Tätigkeiten auszuführen wie bei der Entwicklung der Prinziplösung auf Systemebene einschließlich dem Planen und Klären der Aufgabe. Das Resultat sind die jeweiligen Prinziplösungen der Module.

Während der Phase **Konzeptintegration** werden die Prinziplösungen der Module zu einer Prinziplösung des Gesamtsystems zusammengefügt. Es ist wiederum nach Widersprüchen zu suchen und diese gegebenenfalls mit dem Wirkparadigma der Selbstopti-

[26] Nicht-funktionale Anforderungen beschreiben entweder eine Funktion (z.B. die Geschwindigkeit der Funktion „fahren“) oder ein Bauteil (z.B. dessen Größe).

mierung aufzulösen. Die erarbeitete Lösung der Integration ist abschließend noch nach technischen und wirtschaftlichen Kriterien zu bewerten [GFD+08b, S. 96]. Um den Übergang von der Konzipierung in der Konkretisierung zu ermöglichen, müssen die Informationen aus der Prinziplösung domänenspezifisch ausgeleitet und in domänenspezifische Modelle überführt werden [ADG+09, S. 177].

2.4.3 Handlungsfeld Systementwurf

Im Weiteren wird der Systementwurf einer domänenübergreifenden Lösung während der Konzipierung genauer vorgestellt. DAENZER et al. verstehen unter Entwurf

> *„… die Erahnung eines Ganzen, eines Lösungskonzepts, das Erkennen bzw. Finden der dazu erforderlichen Lösungselemente und das gedankliche, modellhafte Zusammenfügen und Verbinden dieser Elemente zu einem tauglichen Ganzen.“ [DH02, S. 158].*

Die Definition umfasst nicht nur die gestalterische Festlegung möglicher Lösungsvarianten, sondern auch die prinzipielle. Im Maschinenbau wird diesbezüglich klar unterschieden. Die gestalterische Festlegung wird als Entwerfen und die prinzipielle Festlegung als Konzipieren bezeichnet [PBF+07, S. 194]. Im Rahmen dieser Arbeit wird die Phase der prinzipiellen Festlegung in Anlehnung an die VDI-Richtlinie 2206 als **Systementwurf** bezeichnet und beschreibt den zentralen Prozess in der domänenübergreifenden Konzipierung der Prinziplösung.

Von entscheidender Bedeutung für den Systementwurf sind die funktionale Beschreibung und das Aufstellen der Wirkstruktur des zu entwickelnden Systems. Bild 2-17 zeigt den prinzipiellen Verlauf des Systementwurfs mit anschließender Konkretisierung in Bau- und Komponentenstruktur.

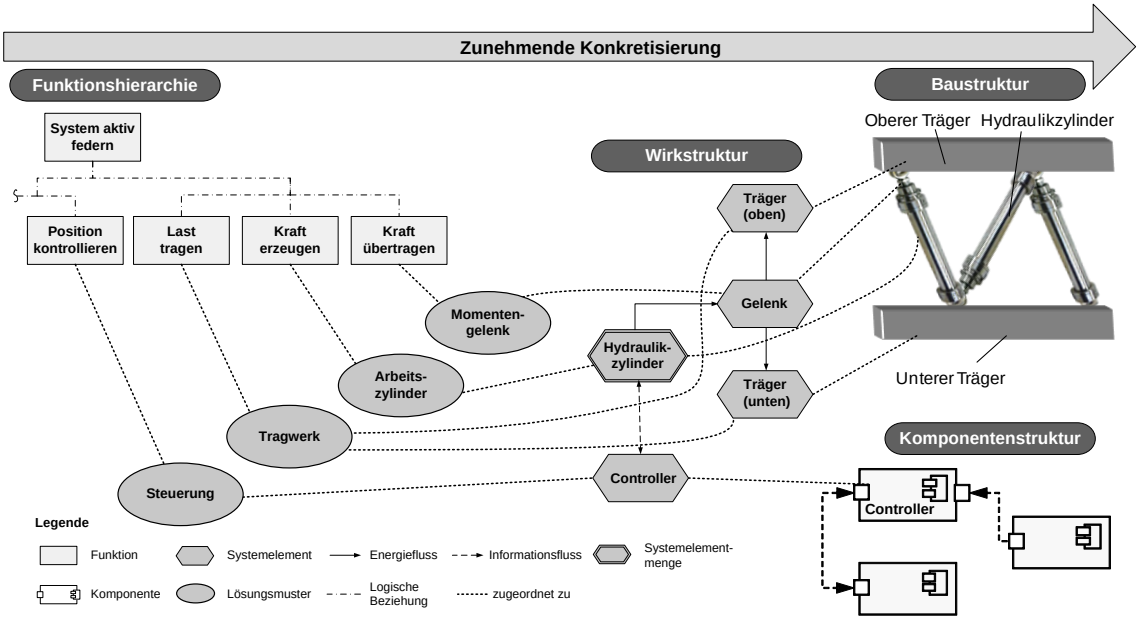


Bild 2-17: Der Systementwurf fortgeschrittener mechatronischer Systeme

Nach PAHL/BEITZ beschreibt eine Funktion *den gewollten Zusammenhang zwischen einem Eingang und einem Ausgang eines Systems, mit dem Ziel eine Aufgabe zu erfüllen* [PBF+07, S. 44]. Eine Funktion wird grundsätzlich durch ein Verb beschrieben und wird in der Regel zusätzlich durch ein Substantiv konkretisiert (z.B. „Energie leiten"). Es ist auf eine lösungsneutrale Beschreibung der Funktion zu achten, um neue Lösungen zu ermöglichen [KK98], [Rot01], [Lan00]. Im Rahmen des Systementwurfs erfolgt die Aufgliederung der gewünschten Funktionalität in einer **Funktionshierarchie**. Auf oberster Ebene befindet sich die Gesamtfunktion. Diese wird auf den unteren Ebenen in Teilfunktionen unterteilt. Funktionen höherer Ebenen werden in mindestens zwei Teilfunktionen unterteilt, die in ihrer Gesamtheit die übergeordnete Funktion erfüllen. Die Unterteilung in Teilfunktionen findet solange statt, bis sich für die Funktionen sinnvolle Lösungen finden lassen [VDI2221, S. 10].

Mit der **Wirkstruktur** werden die Systemelemente sowie deren Merkmale und Beziehungen zueinander modelliert. Ein Systemelement ist ein Konstrukt, das eine Teillösung (z.B. für eine oder mehrere Funktionen) präsentiert, aber noch nicht final ausgestaltet ist. Die Wirkstruktur bildet somit den grundsätzlichen Aufbau und die prinzipielle Wirkungsweise eines fortgeschrittenen mechatronischen Systems feingegliedert ab. Zur besseren Übersichtlichkeit lassen sich mehrere Systemelemente dabei zu übergeordneten logischen oder funktionalen Gruppen zusammenfassen.

Weiter konkretisierte Systemelemente werden als Lösungselemente bezeichnet. Sind diese gestaltbehaftet handelt es sich um Bauteile. Aus deren Anordnung im Raum und logischen Aggregation zu Baugruppen wird der Bauzusammenhang beschrieben. Das Ergebnis ist die **Baustruktur** [PBF+07, S. 56]. Diese ist ein wichtiges Konstrukt zur Ableitung des Produktionssystems (z.B. Montagereihenfolge).

In Anlehnung an SZYPERSKI werden rein informationsverarbeitende Lösungselemente als (Software-)Komponenten[27] bezeichnet. Die **Komponentenstruktur** beschreibt deren Zusammenwirken in einem Gefüge [Oes05]. Zusammen mit der Baustruktur beschreibt die Komponentenstruktur die konkrete Realisierung eines fortgeschrittenen mechatronischen Systems.

Eine zentrale Rolle im Systementwurf nimmt das Konstrukt **Lösungsmuster** ein. In der Literatur wird der Begriff Muster je nach Fachdisziplin unterschiedlich interpretiert. In der Kunst bspw. bezeichnet der Begriff graphische Strukturen, wohingegen sich die Psychologie u.a. auf menschliche Verhaltensmuster konzentriert [Hae04], [Kas03]. Im Kontext des Systementwurfs fortgeschrittener mechatronischer Systeme wird die Definition des Architekturtheoretikers ALEXANDER verwendet [AIS+77]. Er verfolgte die

[27] SZYPERSKi schreibt: *„A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties." [Szy02].*

Grundidee, häufig wiederkehrende Lösungsprinzipien in Form von Mustern festzuhalten und definierte diese Idee wie folgt:

> *„Jedes Muster beschreibt ein in unserer Umwelt immer wieder auftretendes Problem, beschreibt den Kern der Lösung dieses Problems, und zwar so, dass man diese Lösung millionenfach anwenden kann, ohne sich je zu wiederholen.“ [AIS+95, S. 12]*.

Dabei ist es entscheidend, das Problem in Teilprobleme zu zerlegen, für die eine Teillösung ausgearbeitet werden kann. Die anschließende Zusammenführung der Teillösungen soll zur Lösung des übergeordneten Problems führen. Aufgrund der Tatsache, dass in derartigen Mustern lediglich der Lösungskern beschrieben wird, ist eine Anpassung an die jeweiligen Randbedingungen des Gesamtproblems notwendig. Hierzu definiert ALEXANDER vier Kategorien, die jedes Muster enthalten muss [AIS+95]:

1) **Name:** Diese Kategorie besteht lediglich aus einer geeigneten Benennung des Musters und einem Beispielbild.

2) **Kontext**: Hier wird der Kontext beschrieben, in dem das Muster eingesetzt wird. Zusätzlich wird erklärt, wie sich das Muster in übergeordnete Muster einfügen lässt.

3) **Problem**: In dieser Kategorie erfolgt eine detaillierte Beschreibung des Problems für das das Muster anzuwenden ist. Die Aufgabe des Musters muss sich daraus ergeben.

4) **Lösung**: Die Darstellung der Problemlösung ist der wichtigste Teil eines Musters. Die Lösung wird dabei im Sinne einer Anleitung für den Anwender beschrieben. Abschließend folgt eine Erklärung über die Verknüpfung zu untergeordneten Mustern.

Der Ansatz, einmal erfolgreich eingesetzte Lösungen für wiederkehrende Probleme zu verwenden und dieses Entwurfswissen in Form von Lösungsmustern[28] zu repräsentieren, ist in der Entwicklung technischer Systeme grundsätzlich etabliert [Suh93], [Los06]. Einen Vorschlag für eine Klassifikation von bestehenden Lösungsmustern für die Entwicklung selbstoptimierender Systeme und somit auch für fortgeschrittene mechatronische Systeme bieten GAUSEMEIER et al. (Bild 2-18).

[28] Der Begriff wurde von SUHM eingeführt (vgl. Kap. 3.3.1.5). GRABOWSKI et al. verwendeten diesen später im Rahmen der Universal Design Theory. Grundsätzlich werden dort zwei Arten unterschieden. Objektmuster, die die Lösung für physische Erzeugnisse oder auch Software dokumentieren, und Prozessmuster, die den aktiven Gebrauch der Objektmuster definieren [GL00], [Los06].

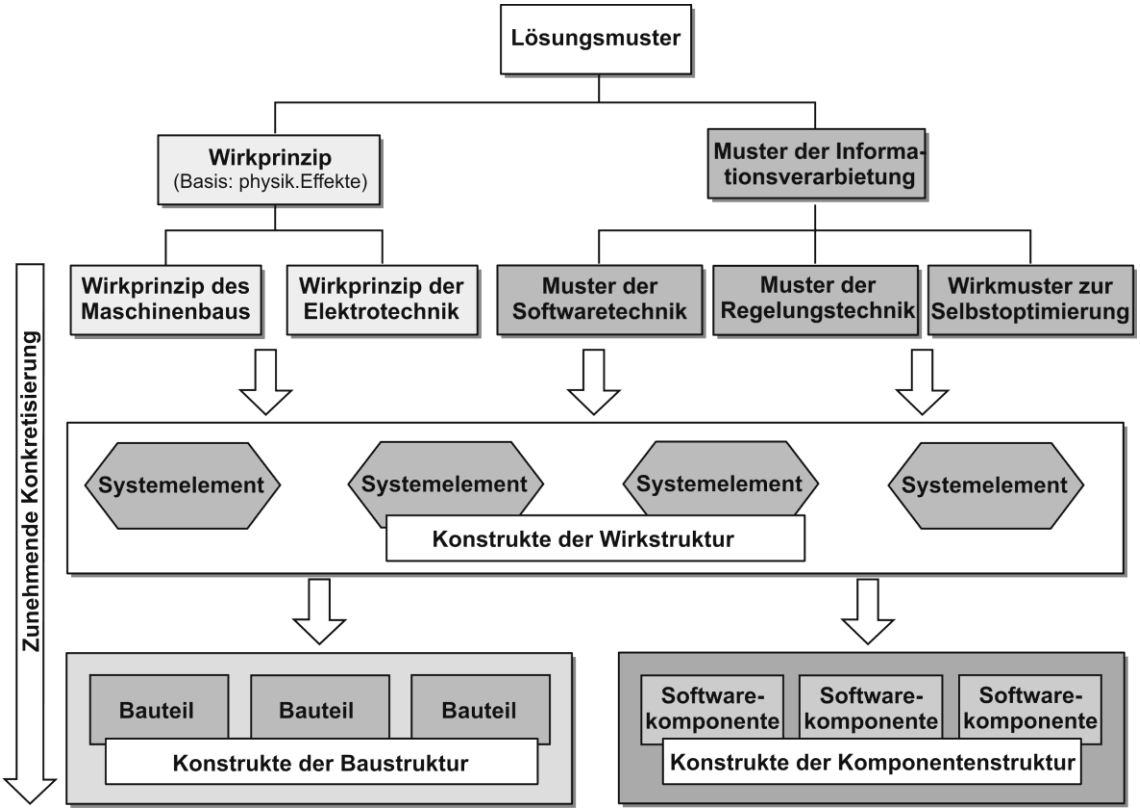


Bild 2-18: Klassifikation von Lösungsmuster [GFD+08a, S. 62]

Die Konstruktionslehre definiert grundlegende Lösungen im Maschinenbau und in der Elektrotechnik als **Wirkprinzipien** [PBF+07, S. 54]. Diese stellen dabei den Zusammenhang zwischen physikalischen Effekt sowie stofflichen und geometrischer Merkmalen her. Beispiele aus dem Bereich des Maschinenbaus sind auf Reibung basierende Passungen oder die Vergrößerung einer Kraft durch den Hebeleffekt (vgl. Bild A-1). Ein Beispiel aus der Elektrotechnik ist das Wirkprinzip Elektromotor, der basierend auf dem physikalischen Effekt der Lorentzkraft elektrische Energie in mechanische Energie wandelt (vgl. Bild A-2) [GHK+06].

Muster der Softwaretechnik werden eingesetzt, um zusammenarbeitende Objekte und Klassen[29] zu speichern, falls diese ein allgemeines Entwurfsproblem lösen. Die Muster enthalten Informationen darüber, wie sie in neuen Situationen genutzt und umgesetzt werden können (vgl. Bild A-3) [GHK+06].

Ein **Muster der Regelungstechnik** hält fest, wie eine Regelstrecke nachgebildet, beeinflusst oder Streckengrößen gemessen oder beobachtet werden können. Diese Muster werden der Informationsverarbeitung zugeordnet (vgl. Bild A-4) [GHK+06].

Wirkmuster zur Selbstoptimierung werden zur Umsetzung des Selbstoptimierungsprozesses genutzt. Sie dokumentieren einen einmal erfolgreich entwickelten Selbstop-

[29] Eine Begriffserklärung erfolgt in Abschnitt 3.2.3.1 bei der Analyse der Modellierungssprache UML.

timierungsprozess. Sie basieren auf Optimierungsverfahren und sind folglich den Mustern der Informationsverarbeitung zugeordnet (vgl. Kap. 3.3.1.4).

2.5 Problemabgrenzung

Mechatronische Systeme beruhen auf dem engen Zusammenwirken von Mechanik, Elektrik/Elektronik, Reglungstechnik und Softwaretechnik. Ein großer und permanent wachsender Teil der Funktionalität mechatronischer Systeme wird durch Softwaremodule realisiert. Aus der fortschreitenden Entwicklung der Informations- und Kommunikationstechnik zeichnet sich eine Systemklasse ab, die über die Mechatronik hinaus geht: fortgeschrittene mechatronische Systeme mit inhärenter Teilintelligenz. Ein Beispiel dieser Systemklasse sind selbstoptimierende Systeme (vgl. Kap. 2.2.3). Derartige Systeme werden in der Lage sein, die Systemumgebung wahrzunehmen, Rückschlüsse auf den eigenen Systemzustand zu ziehen und entsprechend dem Systemzweck ein optimiertes Systemverhalten im Betrieb zu planen und umzusetzen. Es werden sogar Systeme denkbar, die wissen was und wie sie es tun [Bra02, S. 67ff.]. Diese faszinierende Perspektive drücken das Phänomen der Kognition und die damit einhergehende Integration kognitiver Funktionen wie Wahrnehmen, Verstehen, Problemlösen oder Kommunizieren aus. Es zeichnen sich vier wesentliche **Nutzenpotentiale** kognitiver Funktionen in fortgeschrittenen mechatronischen Systemen ab [Eur09, S. 27], [VMS07, S. 153]:

- **Adaptitivität:** Von zentraler Bedeutung ist die Fähigkeit zur Interaktion mit dem Umfeld. So können fortgeschrittene mechatronische Systeme ihr Verhalten als auch ihr Umfeld entsprechend ihres Systemzweckes autonom anpassen. So können sie ihre Existenz langfristig sicherstellen und sich zur Laufzeit in einem vom Entwickler vorgesehenen Rahmen weiterentwickeln.
- **Robustheit:** Fortgeschrittene mechatronische Systeme können auch in einem dynamischen Umfeld flexibel und autonom agieren. Sie besitzen ein robustes Verhalten im Hinblick auf unerwartete und vom Entwickler nicht berücksichtigte Situationen. Unsicherheiten oder fehlende Informationen können bis zu einem gewissen Grad ausgeglichen werden.
- **Effektivität:** Fortgeschrittene mechatronischer Systeme zeigen ein proaktives Verhalten. Zukünftige Einflüsse und mögliche Zustände können antizipiert werden. Gefahren werden frühzeitig vermieden, Ziele werden schneller und in verbesserter Qualität erreicht.
- **Benutzerfreundlichkeit:** Fortgeschrittene mechatronische Systeme berücksichtigen das spezifische Benutzerverhalten. Sie passen sich dem Benutzerverhalten an und verbessern so die Bedienbarkeit. Zudem wird auch das eigene Systemverhalten z.B. auf Basis von historischen Benutzerdaten optimiert.

Die Realisierung eines fortgeschrittenen mechatronischen Systems benötigt neben einem hervorragenden ingenieurwissenschaftlichen Entwurf eine frühzeitige Einbindung der Ergebnisse der höheren Mathematik und der künstlichen Intelligenz wie z.B. Optimierungs-, Lern- oder Planverfahren. Diese müssen den Entwicklern in entsprechender Form bereits während des Systementwurfs zur Verfügung stehen. Ferner müssen im Hinblick auf eine Integration kognitiver Funktionen auch nicht-technische Disziplinen, in erster Linie die Kognitionswissenschaft, berücksichtigt werden. Diese Systeme als auch ihre Entwicklung werden dadurch hochkomplex. Vier grundsätzliche **Herausforderungen** auf dem Weg zur Realisierung kognitiver Funktionen in mechatronischen Systemen sind hervorzuheben:

- **Steigende Systemkomplexität:** RIEGLER nennt vier grundlegende Gründe für die hohe Komplexität kognitiver technischer Systeme (vgl. Kap. 2.3.3). Eine triviale und schnelle Lösung im Sinne eines Kurzschlusses des mechatronischen Regelkreises durch Verfahren der künstlichen Intelligenz wird daher ausbleiben. Hinzu kommt, dass Kognition ein evolutionäres Phänomen ist, das vor dem Betrieb ausgelegt wird, sich aber in diesem stetig weiterentwickelt. Eine Strukturierung und Entwicklung einer kognitiven Informationsverarbeitung kann nur erfolgen, wenn geklärt wird, wie sich eine kognitive Leistung aus elementaren kognitiven Funktionen zusammensetzt und wie diese, z.B. durch welche informationsverarbeitenden Verfahren, realisiert werden. Grundsätzlich hinderlich ist, dass keine allgemein anerkannte Definition der Kognition existiert. Darüber hinaus sind die allgemeinen kognitiven Funktionen nach STRUBE [Str96] für eine Beschreibung technischer Systeme ungeeignet. Eindeutige Beschreibungen für kognitionswissenschaftliche Begriffe im Kontext der Systementwicklung stehen aus. [Ver10, S. 91f.], [Mül98, S. 7].

- **Ungeklärter Informationsbegriff:** Kognitive Theorien, die durch Computermodelle bewiesen werden, gelten als erster Schritt in Richtung zur Entwicklung kognitiver technischer System. Allerdings ist der Informationsbegriff, wie ihn die Kognitionswissenschaft nutzt, nicht unumstritten. BISCHOF hält fest, dass in der Kognitionsforschung der Informationsbegriff für die eigentliche Semantik der Information steht. Diese ist mit den Mitteln der Informationstheorie nicht oder nur eingeschränkt berechenbar. Der Begriff muss daher im Hinblick auf eine Anwendung in der Entwicklung auch in Relation zu den Begriffen Signal, Daten und Wissen konkretisiert werden [Bis09, S. 496ff].

- **Verstärkte Interdisziplinarität:** Die Entwicklung kognitiver Funktionen kann nur durch das Zusammenwirken verschiedener Fachdisziplinen erfolgreich sein. Neben den technischen Disziplinen müssen auch nicht-technische Disziplinen berücksichtigt werden, die sich mit dem Phänomen Kognition beschäftigen. Dies erschwert die ohnehin schon aufwendige Kooperation und Kommunikation innerhalb der Entwicklung mechatronischer Systeme. Konzepte zur frühzeitigen, durchgängigen und gleichberechtigten Beschreibung des zu entwickelnden Systems sind notwendig [HS09, S. 34].

- **Fehlende Entwicklungsmethodik:** Die heute etablierten Entwicklungsmethodiken und -prozesse tragen der verstärkten Interdisziplinarität der Systeme nicht Rechnung. Sie fokussieren in der Regel nur einen bestimmten Aspekt oder eine Disziplin; eine ganzheitliche Systembetrachtung findet – wenn überhaupt – nur ansatzweise statt. Daher wird die Summe der Teillösungen, bei den hier im Fokus stehenden hochkomplexen Systemen, selten die bestmögliche Gesamtlösung ergeben. Doch gerade im Hinblick auf den Entwurf einer kognitiven Informationsverarbeitung ist eine ganzheitliche Betrachtungsweise und eine systematische Vorgehensweise unabdingbar. Neue Entwicklungsansätze und -prinzipien werden für den Entwurf fortgeschrittener mechatronischer Systeme benötigt, die dem frühzeitigen Entwurf der Informationsverarbeitung einen höheren Stellwert geben [Rze03, S. 1034ff.].

Eine Gegenüberstellung der Nutzenpotentiale und der Herausforderungen zeigt, dass die Informationsverarbeitung ein wesentlicher Stellhebel für die Entwicklung fortgeschrittener mechatronischer Systeme ist. Durch die Integration kognitiver Funktionen sind sie gegenüber konventionellen Systemen deutlich leistungsfähiger. Allerdings fehlt es bislang an der methodischen Unterstützung der Entwicklung derartiger Systeme. Bestehende Vorgehen im Entwurf technischer Systeme sowie entsprechende etablierte Methoden sind hier nicht ausreichend und vernachlässigen den frühzeitigen Entwurf der Informationsverarbeitung im Zuge des Systementwurfs. Ferner werden nicht-technische Disziplinen kaum berücksichtigt. Kognitionsrelevante Ergebnisse müssten systematisch für den Systementwurf aufbereitet sein. Aus diesen Gründen besteht ein **Bedarf** für eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*. Diese sollte folgende Bestandteile beinhalten:

- **Strukturiertes Vorgehensmodell:** Kern der Entwicklungssystematik muss ein Vorgehensmodell sein, das den frühzeitigen Entwurf kognitiver Funktionen in fortgeschrittenen mechatronischen Systemen systematisiert. Das Vorgehen soll die notwendigen Tätigkeiten und entsprechende Hilfsmittel integrieren, die im Rahmen des Systementwurfs zur Spezifikation der kognitiven Informationsverarbeitung durchzuführen bzw. einzusetzen sind. Eine idealtypische Darstellung ist ausreichend.

- **Technik zur Systembeschreibung:** Die Entwicklung einer kognitiven Informationsverarbeitung benötigt eine aussagestarke und eindeutige Beschreibung des zu entwickelnden Systems. Insbesondere eine funktionale Sicht ist unerlässlich, die in eine aus Systemelementen bestehende strukturelle Sicht übergeht. Nur so ist eine ganzheitliche und frühzeitige Spezifikation der kognitiven Informationsverarbeitung zu erreichen.

- **Wiederverwendbares Lösungswissen:** Die Informationsverarbeitung bildet den Kern zur Umsetzung kognitiver Funktionen. Die Regelungs- und Steuerungstechnik, die Softwaretechnik, als auch die höhere Mathematik und künstliche Intelligenz sind hierbei die wesentlichen Disziplinen. Deren Methoden und Techniken müssen so aufbereitet sein, dass sie von Nicht-Experten auf diesen Gebieten als Teillösungen

ausgewählt werden können. Die Dokumentation soll in Form von Lösungsmustern erfolgen, für die eine einheitliche und disziplinunabhängige Spezifikation abzuleiten ist. Der Anspruch ist, einen Großteil der frühzeitigen Spezifikation der Informationsverarbeitung durch Kombination bestehender Lösungsmuster sicherzustellen.

- **IT-Konzept zur Unterstützung:** Das wiederverwendbare Lösungswissen muss rechnerintern abgebildet werden und vollständig in einer Wissensbasis hinterlegt werden. Diese muss im Hinblick auf die Interdisziplinarität des Systementwurfs über ein geeignetes Zugriffssystem verfügen, das eine effektive Suche ermöglicht. Das Konzept der Wissensbasis ist prototypisch zu realisieren. Dabei können bestehende Werkzeuge, z.B. zur Modellierung der Lösungsmuster, genutzt werden. Diese sind an die Wissensbasis anzubinden.

2.6 Anforderungen an die Arbeit

Aus der Problemanalyse resultieren folgende Anforderungen an eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*:

A1) Interdisziplinarität: Neben dem Zusammenwirken der mechatronischen Teildisziplinen muss die Entwicklungssystematik auch die höhere Mathematik und künstliche Intelligenz sowie kognitionsrelevante nicht-technische Disziplinen integrieren. Insbesondere der kognitionswissenschaftliche Anspruch steht im Vordergrund.

A2) Systematische Vorgehensweise: Kognition bietet ein enormes, teils unüberschaubares Repertoire für technische Systeme. Die Entwicklungssystematik muss eine zielgerichtete und systematische Entwicklung neuer Lösungen ermöglichen. Das Vorgehen muss sich in das etablierte Vorgehen im Entwurf mechatronischer Systeme einordnen.

A3) Orientierung am Systementwurf: Im Rahmen des Systementwurfs etablierte sich insbesondere die Generierung erster Lösungsansätze in Form von Wirkstrukturen anhand zu erfüllender Funktionen. Die Entwicklungssystematik muss sich an diesem Vorgehen orientieren und die frühen Phasen der Entwicklung fokussieren. Ihre Bestanteile sind entsprechend zu gestalten.

A4) Verständlichkeit: Die Entwicklungssystematik wird in erster Linie von Ingenieuren eingesetzt und muss auch Entwickler unterstützen, die nicht über tiefgehende Fachkenntnisse der höheren Mathematik, der künstlichen Intelligenz oder Kognitionswissenschaft verfügen. Das erfordert eine einfache und verständliche Darstellung der Entwicklungssystematik und ihren Inhalten.

A5) Ganzheitliche Spezifikation der Informationsverarbeitung: In Anlehnung an das Dreischichtenmodell nach STRUBE [Str98] muss neben einer kognitiven zumindest noch eine reaktive Informationsverarbeitung existieren. Die Entwicklungssystematik muss eine geeignete Grundstruktur nutzen und die funktionale Beschreibung der Infor-

mationsverarbeitung inkl. kognitiver Funktionen ermöglichen. In diesem Zuge ist es notwendig, die Begriffe Signal, Daten, Information und Wissen zu unterscheiden.

A6) Technische Relevanz: Beschreibungsmöglichkeiten der allgemeinen Systemtheorie beschreiben zwar komplexe Vorgänge, haben aber kaum technische Relevanz im Hinblick auf die Entwicklung fortgeschrittener mechatronischer Systeme. Die Entwicklungssystematik muss daher eine technikorientierte Beschreibung der kognitiven Funktionen und des zu entwickelnden Systems sicherstellen, die eine ausreichende Beschreibungstiefe bietet.

A7) Lösungswissen für den Entwurf der Informationsverarbeitung: Die Entwicklungssystematik soll die Dokumentation und Wiederverwendung von Expertenwissen sowie erfolgreich eingesetzten Lösungen in Form von Lösungsmustern unterstützen. Diese müssen sich an der Grundstruktur der Informationsverarbeitung orientieren und sind in einem disziplinunabhängigen Schema festzuhalten. Integrale Bestandteile sollten die Kategorien der Musterdefinition nach ALEXANDER [AIS+95] (Name, Lösungs-, Problem- und Kontextbeschreibung) sein, die effektiv, z.B. in Form von Partialmodellen, in die Gesamtsystemspezifikation zu integrieren sind.

A8) Auswahl und Kombination von Lösungswissen: Während des Systementwurfs werden potentielle Teillösungen ausgewählt und zu unterschiedlichen Lösungsvarianten kombiniert. Die zu erarbeitenden Lösungsmuster müssen daher Merkmale für eine sinnvolle Auswahl und Kombination besitzen. Ferner ist die Kompatibilität zu bestehenden Lösungsmustern, wie z.B. den Wirkprinzipien der Konstruktionslehre, durch ein einheitliches Schema sicherzustellen.

A9) Rechnerunterstützung durch eine Wissensbasis: Sowohl die Spezifikation des Systems als auch die Lösungsmuster sind rechnerintern abzubilden. Insbesondere eine Wissensbasis für den Umgang mit dem Lösungswissen wird gefordert. Neben der Sicherung von Lösungsmustern müssen geeignete Filter- und Suchfunktionen konzipiert werden, die eine disziplinübergreifende Auswahl und Kombination unterstützen.

3 Stand der Technik

Dieses Kapitel untersucht bestehende Konzepte zur methodischen Entwicklung fortgeschrittener mechatronischer Systeme. Vor dem Hintergrund der Problemanalyse werden in Kapitel 3.1 erste entwicklungsmethodische Ansätze zur Integration kognitiver Funktionen in technische Systeme analysiert. Kapitel 3.2 befasst sich mit der Spezifikation der kognitiven Informationsverarbeitung fortgeschrittener mechatronischer Systeme. Hierfür werden wesentliche Methoden zur Funktionsbeschreibung, Architekturen zur Strukturierung und Techniken zur Beschreibung der Informationsverarbeitung betrachtet. Kapitel 3.3 stellt Möglichkeiten vor, einmal erfolgreich eingesetztes Lösungswissen wiederzuverwenden. Nachdem Ansätze zur Dokumentation von Lösungen in Mustern vorgestellt werden, stehen zwei grundlegende IT-Konzepte für den Umgang mit Lösungswissen im Fokus. In Kapitel 3.4 wird der Stand der Technik anhand der Anforderungen aus der Problemanalyse bewertet. Aus dieser Bewertung resultiert der Handlungsbedarf für diese Arbeit.

3.1 Methodische Ansätze für die Entwicklung kognitiver Funktionen

Auch wenn die Mechatronik als eigenständige Disziplin noch recht jung ist, bestehen mittlerweile etliche Methodiken und Vorgehensweisen für eine systematische Entwicklung. Den minimalen Konsens der Fachwelt fasst die VDI-Richtlinie 2206 zusammen (vgl. Kap. 2.4.1). Im Folgenden werden daher nur entwicklungsmethodische Ansätze vorgestellt, die die Integration kognitiver Funktionen in mechatronischen Systemen adressieren.

3.1.1 Framework für kognitive Produkte

Im Rahmen des Exzellenzclusters CoTeSys[30] wurde ein Framework für kognitive Produkte entwickelt. Ziel von CoTeSys ist es, technische Systeme mit kognitiven Fähigkeiten wie Wahrnehmen, Lernen oder Planen zu realisieren. Derartige Systeme werden als kognitive technische Systeme (Cognitive Technical Systems – CTS) bezeichnet und sind in Anlehnung an BRACHMAN dadurch gekennzeichnet, dass sie *wissen was sie tun* [Bra02, S. 68], [BBW07, S. 20].

Das Framework soll ein erster Ansatz sein, derartige Systeme systematisch und methodisch zu entwickeln. Ausgangspunkt ist die Definition geeigneter kognitiver Funktio-

[30] Der Exzellenzcluster „Cognition for Technical Systems – CoTeSys“ wurde im Zuge der Exzellenzinitiative des Bundes und der Länder 2005 eingerichtet und wird seit 2006 durch die Deutsche Forschungsgemeinschaft (DFG) gefördert. Insgesamt sind fünf Universitäten bzw. eigenständige Institute im Raum München beteiligt.

nen. Entsprechend der Vorgehensweise in der Konstruktionsmethodik besitzen diese eine Substantiv-Verb Form (Bild 3-1) [MS10, S. 871f.].

kognitive Funktionen		Kaffee Service-roboter	Kognitiver Ofen	Roboter-schwarm Spiel	Carrera Spielzeug-auto	Vernetzte Steckdosen
lernen	Ort/Lage	x		x		o
	Zeitpunkt	x	x			x
	Verhaltensmodell		x		x	
	Umgebungsmodell	x				x
	Ressourceneinsatz	x	x			x
	Benutzermodelle		x	o		x
wissen	Umgebungsmodell	x		x	x	x
	Benutzermodelle	o	x	o		
	eigenes Modell	x	x	x		
begründen	Aufgabenerfüllung					
	alternative Aktionen					
planen	Route	x		x		
	Arbeitsplan	x	x	x		x
	Ressourceneinsatz	x	x			x
wahrnehmen	Ort/Lage	x		x	x	
	Benutzererkennung		x	o		x
	Objekterkennung	o	x	x		o
kommunizieren und interagieren	Benutzeranpassung	o	x	o	x	x
	Anwender lehren			x		
	Rückmeldung geben	x	x			
ausführen	autonomes Bewegen	x		x	x	
...	...	...	...	...	...	...

o = Teil des Konzepts x = im Prototypen implementiert

Bild 3-1: Framework für kognitive Produkte nach [MS10, S. 872]

Insgesamt wurden sieben Funktionsverben identifiziert, die als kognitive Fähigkeiten bezeichnet werden. Das Resultat sind 21 kognitive Funktionen, die für die Entwicklung von fünf kognitiven Produkten im Rahmen von Studierendenseminaren festgelegt wurden [MS10, S. 871f.]:

- **Kaffee Kellner:** Der Roboter serviert in einer erlernten Umgebung eigenständig Kaffee. Entsprechend der Anfragen und der internen Ressourcen berechnet er eine effiziente Ausfahrroute.
- **Kognitiver Ofen:** Der verbessert seine Backparameter auf Basis des Nutzerverhaltens.
- **Roboterschwarm:** Fünf autonome Roboter, die sich auf einer virtuellen Karte bewegen, sollen in Interaktion mit einem Benutzer verschiedene Aufgaben erfüllen.
- **Carrera Autorennbahn:** Dabei handelt es sich um die Programmierung eines virtuellen Carrera Autos, das das Fahrverhalten der menschlichen Gegner erlernt.
- **Vernetzte Steckdosen:** Die vernetzten Steckdosen erlernen den Strombedarf an den einzelnen Steckdosen und optimieren die tägliche Nutzung.

Bewertung: Das Framework zeigt einen ersten Ansatz zur Klassifikation kognitiver Funktionen. Eine Definition von allgemeinen kognitiven Funktionen stellen die identifizierten kognitiven Fähigkeiten dar. Eine fundierte Herleitung dieser Begriffe fehlt. Das Framework dient in erster Linie zur abstrakten Suche nach Innovationen für bestehende Produkte, indem Anwendungsfälle für die Integration einzelner kognitiver Funktionen abgeleitet werden können. Das Framework bietet keinen systematischen Ansatz zur Entwicklung der kognitiven Informationsverarbeitung. Eine Zuordnung von konkreten Lösungsmöglichkeiten innerhalb des Frameworks wäre denkbar, entsprechende Lösungen werden aber nicht aufgeführt.

3.1.2 Entwurf anpassungsfähiger Produkte

Im Rahmen des Verbundprojekts Octopus[31] haben CHMARRA et al. einen methodischen Ansatz zur Entwicklung sog. anpassungsfähiger Produkte (engl. adaptable products) erarbeitet. Diese Anpassungsfähigkeit bezieht sich auf Produkteigenschaften, die in vorgedachten Situationen zur Verbesserung der Leistungsfähigkeit geändert werden können. Die Anpassung kann dabei nur durch das System selbst erfolgen und nicht durch einen Benutzer von außen. Ein wesentliches Ziel ist es, bereits während der Entwicklung sinnvolle Produktkonfigurationen vorauszudenken. Dies kann z.B. durch eine geschickte und flexible Produktmodularisierung erfolgen [CT08].

Fokus der Arbeiten liegt auf dem Entwurf einer geeigneten Systemarchitektur. Hierfür wurde eine generische Schablone mit zwei wesentlichen Komponenten definiert. Das *Steuerungs- und Entscheidungssystem* hat die Aufgabe, aus externen Informationen (z.B. Benutzerwünschen) und intern errechneten Zuständen (z.B. Temperaturmessung) Änderungen zu erkennen und geeignete Handlungen zu bestimmen. Hierfür müssen auf Basis der Produktkonfiguration Aktionspläne vordefiniert sein, die schließlich von der *Funktionseinheit des Produkts* physisch ausgeführt werden. Bei dem Entwurf dieser beiden Komponenten stehen drei wesentliche Fragestellungen im Vordergrund [CAT08]:

- Welche Verfahren sollen implementiert werden, um das System zu steuern und um geeignete Aktionen zu bestimmen?
- Welche Sensoren sind notwendig, um die benötigten Informationen zu erfassen?
- Wie muss die Architektur der Informationsverarbeitung entworfen werden, um eine effektive physische Umsetzung des Grundsystems inkl. Aktorik zu erreichen?

Als ein mögliches Verfahren nennen CHMARRA et al. beispielhaft die mathematische Optimierung mit einer Paretomenge. Für den Entwurf der Informationsverarbeitungsar-

[31] Das Projekt wird vom niederländischen Wirtschaftsministerium im Rahmen des Bsik Programms gefördert.

chitektur sind noch weitere Entscheidungen zu treffen. Es ist die Domäne und Systemebene für die Anpassung festzulegen (z.B. mechanische Regelung). Zudem ist eine Anpassungsstrategie zu definieren und festzulegen, für welche Einflüsse eine Anpassung durchzuführen ist (z.B. wechselnde Benutzer).

Bewertung: Das Konzept der anpassungsfähigen Produkte adressiert fortgeschrittene mechatronische Systeme. CHMARRA et al. beschreiben erste Ansätze zur methodischen Entwicklung, die eine generische Entwurfsschablone und entsprechende Entwurfsentscheidungen umfassen. Letztere sind in ihrer Kernaussage sicherlich korrekt, es werden aber weder konkreten Hilfsmittel noch ein explizites Vorgehen beschrieben. Die Auswahl an einzusetzenden Verfahren ist auf eins beschränkt. Die Umsetzung wird nur beispielhaft an einem anpassungsfähigen Druckergerät gezeigt, ohne die genaue Architektur der Informationsverarbeitung oder des Gesamtsystems zu spezifizieren.

3.1.3 Entwicklung kognitiver technischer Systeme nach PAETZOLD

PAETZOLD charakterisiert die Entwicklung von kognitiven Funktionen in technischen Systemen als eine konsequente Weiterentwicklung mechatronischer Systeme. Kognitiven Funktionen entsprechen dabei den allgemeinen kognitiven Funktionen nach STRUBE (vgl. Kap. 2.3.2). Ebenso wird STRUBES Dreischichtenmodell zur Strukturierung der kognitiven Informationsverarbeitung vorgeschlagen. Eine entsprechende Modifikation der Systemarchitektur soll durch eine Erweiterung der Informationsverarbeitung im mechatronischen Regelkreis erfolgen [Pae06], [SP95].

Die Umsetzung dieser Modifikation ist für die Produktentwicklung eine große Herausforderung. Ein wesentlicher Grund ist die Komplexität solcher Systeme. Der Entwickler muss definieren können, über welche Fähigkeiten das System im Hinblick auf die Informationsverarbeitung, Sensorik, Aktorik und das Grundsystem verfügen muss. Damit einher geht die Frage nach dem Grad der technischen Effizienz, um ein kognitives Überrüsten zu vermeiden. Aber nicht nur das zu entwickelnde System ist komplex, auch die Entwicklung als Folge der interdisziplinären Vorgehensweise [SP05, S. 145].

PAETZOLD schlägt daher einen Bottom-Up-Ansatz für die Suche nach der kleinsten Einheit eines kognitiven technischen Systems vor. So könnte der Architekturansatz überprüft und die Komplexität durch das System beherrscht werden. Der in der Produktentwicklung typische Top-Down-Ansatz sei aber immer noch im Hinblick auf den zu realisierenden Systemzweck von entscheidender Bedeutung. Ferner spielt die Funktionsmodellierung eine tragende Rolle. Zum einen für die Erstellung einer rechnergestützten Simulation des Systemverhaltens und zum anderen zur Integration der beteiligten Fachdisziplinen während des Entwicklungsprozesses [Pae06, S. 970ff.].

Als Handlungsempfehlungen für die Entwicklung kognitiver Funktionen in technischen Systemen, beziehen sich PAETZOLD et al. auf die Leitwerte für selbstorganisierende Sys-

teme nach BOSSEL [Bos94]. Sie eignen sich in erster Linie für eine Beschreibung des Systemzwecks und der systemerhaltenden Funktionen [SP05, S. 146f.]:

- **Existenz:** Das zu entwickelnde System muss nicht nur seinem Systemzweck entsprechend funktionieren, sondern auch so konstruiert sein, dass es in seiner Umgebung überlebensfähig ist. So lange es funktioniert, existiert es auch.
- **Wirksamkeit:** Die Systemeffizienz ist ausschlaggebend für die Systemstabilität. Gefährdet wird diese durch die interne Ressourcennutzung oder die Systemumgebung.
- **Handlungsfreiheit:** Die Ausprägung der kognitiven Funktionen ist entsprechend der Systemumgebung festzulegen. Zudem ist die Robustheit der Systemstruktur sicherzustellen.
- **Sicherheit:** Die Umgebung ist aufgrund ihrer Zeitvarianz ein Unsicherheitsfaktor. Daraus folgt, dass das System von unvorhersehbaren Umwelteinwirkungen unabhängig sein muss, z.B. durch integrierte Diagnoseverfahren, die solche Probleme registrieren, deuten und sofort Gegenmaßnahmen einleiten.
- **Wandlungsfähigkeit**: Soll die Selbstständigkeit des Systems auch in kritischen Situationen gewährleistet werden, müssen entsprechende Strategien zur Wandlungsfähigkeit vorgesehen werden.
- **Rücksichtnahme:** Zur Erfüllung des Systemzwecks im kooperativen Verbund stehen Strategien vom Altruismus bis zum Egoismus zur Verfügung. Aber auch innerhalb eines Systems muss das Prinzip der Rücksichtnahme beachtet werden, denn nicht selten können systeminterne, parallele oder unabhängige Prozesse in Konkurrenz zueinander stehen. Eine unkoordinierte Lösung kann schließlich die Systemexistenz bedrohen.

Bewertung: Der untersuchte Ansatz ist noch sehr abstrakt. Er orientiert sich zwar an fundierten Ergebnissen der Kognitionswissenschaft, eine konkrete Methodik fehlt aber. Die Leitwerte nach BOSSEL sind sehr allgemein formuliert. Sie können nur als erste Orientierungshilfe und grobe Entwicklungskriterien bei der Integration kognitiver Funktionen herangezogen werden. Sie sind zudem nicht konkret in den Kontext der Entwicklung mechatronischer Systeme eingebunden.

3.2 Spezifikation der kognitiven Informationsverarbeitung

Es gibt keinen ganzheitlichen Ansatz zur Spezifikation einer kognitiven Informationsverarbeitung in den frühen Phasen des Entwurfs. Infolgedessen werden zunächst allgemeine Methoden zur funktionalen Beschreibung (vgl. Kap. 3.2.1). Anschließend werden Architekturen für eine kognitive Informationsverarbeitung untersucht (vgl. Kap. 3.2.2). Um die Funktionen als auch die Architektur beschreiben zu können, ist eine Modellie-

rungssprache notwendig. Relevante Sprachen werden daher abschließend analysiert (vgl. Kap. 3.2.3).

3.2.1 Methoden der Funktionsbeschreibung

3.2.1.1 Funktionsbeschreibung in der Produktentwicklung

In der Produktentwicklung sind unterschiedliche Konzepte zur Funktionsbeschreibung verbreitet. Sie ist von zentraler Bedeutung in der Entwicklung technischer Systeme. Allgemeines Ziel der Funktionsbeschreibung ist die ganzheitliche sowie lösungsneutrale Beschreibung der Systemfunktionalität, um die Entwicklungsaufgabe zu konkretisieren und erste Lösungsvarianten zu identifizieren. Hierzu werden grundlegende Funktionen zur Beschreibung technischer Systeme definiert. Die **Konstruktionslehre nach PAHL/BEITZ** schlägt in Anlehnung an KRUMHAUER die fünf allgemein anwendbaren Funktionen *Wandeln*, *Ändern*, *Verknüpfen*, *Leiten* und *Speichern* vor. Sie sind nicht weiter unterteilbar und werden mit den Substantiven *Stoff*, *Energie* und *Signale/Daten* (*Informationen*) verknüpft. Sie beschreiben den Eingang und Ausgang und die Verben die entsprechende Transformation [PBF+07, S. 47f.].

Mehrere **VDI-Richtlinien** greifen das Thema der Funktionsmodellierung auf:

- **VDI-Richtlinien 2221/2222:** Im Fokus stehen die Ermittlung der Funktionen und das Aufstellen einer Funktionsstruktur, in der die Funktionen miteinander verknüpft werden. Das Vorgehen wird auf die Software-Entwicklung übertragen, in der für die Funktionen Algorithmen oder Datenstrukturen zu identifizieren sind [VDI2222, S. 15ff.], [VDI2221, S. 30ff.].
- **VDI-Richtlinie 2803**: Im Vordergrund steht die Funktionsanalyse. Sie enthält Vorschriften bzgl. der Beschreibungsform und des Abstraktionsgrads einer Funktion. Demnach ist eine Verb-Substantiv-Kombination zwar ausreichend, aber sie muss in aktivistischer Form definiert sein. So kann der Interpretationsspielraum reduziert werden (z.B. „Flüssigkeit fördern“ anstatt „Flüssigkeitsförderung ermöglichen“) [VDI2803, S. 2].
- **VDI-Richtlinie 2860:** Diese Richtlinie beschreibt die Montage und Handhabungstechnik. Es werden entsprechende Funktionen (*Speichern*, *Verändern*, *Bewegen*, *Sichern* und *Kontrollieren*) definiert [VDI2860, S. 4f.].

Ziel der **VDW-Richtlinie 02.2002** ist die universelle Beschreibung von Funktionen, um eindeutige und unmissverständliche Unterlagen innerhalb eines Unternehmens erstellen zu können. Es wird explizit eine domänenübergreifende und entwicklungsbegleitende Funktionsbeschreibung adressiert. Von zentraler Bedeutung ist das sog. Funktionsobjekt. Dieses integriert technische Daten als auch Systemfunktionen in einer gekapselten

Einheit. Sie orientiert sich dabei an maschinenbaulichen Funktionen und auch an die Programmierung [VDW02.02, S. 8f.].

LINDEMANN unterscheidet drei Formen der Funktionsmodellierung, die eine hierarchische als auch strukturelle Darstellung unterstützen [Lin09, S.119ff.]:

- **Umsatzorientierte Funktionsmodellierung:** Ziel ist die Darstellung der Zustände der Energie-, Stoff- und Signalumsätze in einem System. Dafür werden Änderungen des Umsatzproduktes funktional beschrieben.
- **Nutzenorientierte Funktionsmodellierung:** Mit dieser Methode kann die Interaktion zwischen dem zu entwickelnden System und dem Nutzer (z.B. Anwender oder Monteur) beschrieben werden. Hierzu werden Anwendungsfälle definiert, um auch ungewollte oder unentdeckte Nutzungsmöglichkeiten zu identifizieren.
- **Relationsorientierte Funktionsmodellierung:** Ziel dieser Modellierungsform ist es, Stärken und Schwächen des Systems aufzuzeigen. So sollen nützliche Funktionen gestärkt und schädliche vermieden oder abgeschwächt werden. Die Funktionen werden hierfür in Relation zueinander gesetzt.

KOLLER/KASTRUP unterscheiden Funktionsverben für Energieoperationen, Stoffoperationen, Grundoperationen zwischen Energie und Stoff sowie für die Datentechnik. Den Funktionsverben werden direkt die Substantive Stoff, Energie, und Daten zugeteilt. Von besonderer Bedeutung für diese Arbeit könnten die nicht physikalischen elementaren Funktionen für die Datentechnik sein [KK98, S. 26]:

- **verknüpfen:** Rechner können Daten verknüpfen.
- **vervielfältigen:** Kopiergeräte vervielfältigen Daten auf den einzelnen Seiten.
- **leiten/isolieren:** Daten sind von einem Ort A nach B zu leiten und vor unberechtigtem Zugriff zu isolieren.
- **umcodieren:** Daten werden umcodiert (analog in digital oder vice versa).
- **speichern:** Daten werden auf einer Festplatte gespeichert oder auch ausgedruckt.

CLAUSSEN/RODENACKER haben technische Funktionsverben definiert, um primär Verknüpfungen oder Trennung verschiedener technischer (Teil-)Systeme zu beschreiben. Ihre Aufstellung beschränkt sich dabei auf die Mechanik [CR98, S. 168].

BIRKHOFER erarbeitete Funktionsverben zur Beschreibung technischer Systeme mit Hilfe eines technischen Wörterbuchs. Dabei wurden im ersten Schritt ca. 1000 technische Verben entnommen. Aus diesen wurden anschließend alle Verben gestrichen, die durch Vorsilben abgeleitet sind und nur eine Präzisierung des Begriffsinhalts darstellen [Bir80, S. 70]. Übrig blieben 222 Verben für technische Funktionen. **ROTH** hat die Funktionsverben von BIRKHOFER übernommen und um die drei Verben *bewegen*, *fügen* und *rückgewinnen* erweitert [Rot94, S. 63]. In Anlehnung an ROTH hat **LANGLOTZ** im

Hinblick auf eine rechnergestützte Konstruktion die allgemeinen Funktionen *speichern, leiten, isolieren, wandeln, umformen, trennen* und *zusammenführen* festgelegt. Diesen wurden weitere Funktionsverben zugeordnet. LANGLOTZ verwendet keine Kombination mit den Substantiven Energie, Stoff und Signale/Daten, sondern mit konkreten Substantiven aus dem technischen Sprachgebrauch [Lan00, S. 109].

Ein Ansatz der sich konkret auf die Funktionsmodellierung mechatronischer Systeme bezieht, ist die **Funktionsmodellierung mit kanonischen Funktionen**. HUANG schlägt zwischen der Modellierung mit allgemeinen Funktionen und der mit speziellen Funktionen eine Übergangsstufe vor. In dieser soll unter Berücksichtigung der zu realisierenden Effekte allgemeine Funktionen in kanonische überführt werden. Diese besitzen den Anspruch eine höhere Anzahl an disziplinunabhängigen Funktionen zu bieten und neutrale sowie handhabungsfähige Funktionsgrößen zu verwenden. Resultat sei eine verifizierbare Funktionsstruktur [Hua02, S. 37].

Bewertung: Die Mehrzahl der untersuchten Methoden eignen sich für die Beschreibung energetischer und stofflicher Prozesse. PAHL/BEITZ, die VDI-Richtlinie 2860, CLAUSSEN/RODENACKER, BIRKHOFFER, ROTH und LANGLOTZ adressieren mechatronische Systeme nicht direkt. Die VDI-Richtlinie 2221 schlägt nur ein beispielhaftes Vorgehen für die Softwareentwicklung vor. Die VDW-Richtlinie führt das Konstrukt Funktionsobjekt ein und umgeht so die klassische Funktionsbeschreibung mit Verben. Die Modellierungsformen nach LINDEMANN sind zwar allgemein anwendbar, aber für die Spezifikation der Informationsverarbeitung ungeeignet. Dieser Bereich wird einzig von KOLLER/KASTRUP und HUANG aufgegriffen. Beide liefern aber nur erste Ansätze. Die vollständige Beschreibung der Informationsverarbeitung fortgeschrittener mechatronischer Systeme erfüllt keiner der untersuchten Ansätze.

3.2.1.2 Funktionsbeschreibung in der Softwaretechnik

Eine funktionale Beschreibung im Sinne einer Substantiv-Verb-Kombination existiert bislang nur in der Konstruktionsmethodik. Dennoch kann die Entwicklung eines Software-Produktes ebenfalls funktional beschrieben werden. BALZERT definiert eine Funktion im Kontext der Softwaretechnik wie folgt:

> *„Eine Funktion beschreibt eine Tätigkeit oder eine klar umrissene Aufgabe innerhalb eines größeren Zusammenhangs. In der Software-Technik ermittelt eine Funktion aus Eingabedaten Ausgebedaten oder bewirkt eine Veränderung des Inhalts oder der Struktur von Informationen.“ [Bal01, S. 124]*.

Hauptfunktionen werden in einem Funktionsbaum in Teilfunktionen zerlegt. Dabei kann diese Beziehung unterschiedlich gewertet werden. In der Definitionsphase setzt sich die Hauptfunktion aus ihren Teilfunktionen zusammen. In der Entwurfsphase ruft die Hauptfunktion ihre Teilfunktionen auf. Die Benennung einer Funktion sollte durch

ein Verb und entweder ein Objekt („verwalte Kunden“) oder ein Subjekt („Kunden verwalten“) erfolgen. Nach der Auswahl einer der beiden Formen sollte diese beibehalten werden [Bal01, S.124f.].

In der Softwaretechnik existieren zudem sog. **Prozessworte**. Diese beschreiben prosaisch die Funktionalität eines Systems und legen so die funktionalen Anforderungen fest. Die Beschreibung wird neben Verben zusätzlich mit Substantiven, Adjektiven und Adverbien konkretisiert. Sie könnten als Infinitive zur Funktionsbeschreibung genutzt werden. Für jedes Prozesswort ist eine semantische Definition zu erarbeiten. Diese Definition ist für das jeweilige Wort eindeutig und verbindlich. Dadurch kann ein einheitliches Verständnis erreicht werden. Um eine eindeutige Formulierung von Funktionen vornehmen zu können, sollte die Menge der Prozessworte auf die für das System relevante Anzahl begrenzt werden [Rup07, S. 238f.]. Tabelle 3-1 zeigt eine definierte Prozesswortliste.

Tabelle 3-1: Prozesswortliste nach [Rup07, S. 240]

Prozesswort	Semantische Definition des Prozessworts	Selbstständige Aktivität	Benutzerinteraktion	Potenzielle Fähigkeit
auswählen	Der NUTZER selektiert eines oder mehrere Elemente aus einer einheitlichen Menge von Elementen (siehe auch "bestimmen").	Nein	Ja	Nein
bestimmen	Das SYSTEM selektiert anhand bestimmter Selektionskriterien aus einer endlichen Menge (zum Beispiel aus einer Datenbank) ein oder mehrere Elemente.	Ja	Nein	Nein
darstellen	Das SYSTEM zeigt dem Nutzer Informationen an.	Ja	Nein	Nein
einfügen	Ausschließlich das SYSTEM gibt neue Daten ein oder überschreibt vorhandene Daten (siehe auch "einfügen", "einsetzen").	Ja	Nein	Nein
eingeben	Ausschließlich der NUTZER gibt neue Daten ein oder überschreibt vorhandene Daten (siehe auch "einfügen", "einsetzen").	Nein	Ja	Nein
einsetzen	Sowohl das SYSTEM als auch der NUTZER geben neue Daten ein oder überschreiben vorhandene Daten (siehe auch "einsetzen", "eingeben").	Ja	Ja	Nein
empfangen	Das System ist IMSTANDE, Daten von einem externen System elektronisch entgegenzunehmen.	Nein	Nein	Ja
erstellen	Sowohl das SYSTEM (selbstständig) als auch der NUTZER erzeugen Objekte.	Ja	Ja	Nein

Bewertung: In der Softwaretechnik wird der Funktionsbegriff ähnlich genutzt wie in der Konstruktionsmethodik. Der Funktionsbaum der Softwaretechnik ähnelt der Darstellungsform einer lösungsneutralen Funktionshierarchie in der Produktentwicklung. Allgemeine Funktionen zur Beschreibung von Software existieren nicht. Ein erster An-

satz sind die Prozessworte nach RUPP. Sie unterstützen bei der prosaischen Formulierung der funktionalen Anforderungen. Mit Hilfe einer entsprechenden Prozesswortliste kann man relevante Funktionen im Projekt gemeinsam und einheitlich definieren. Die funktionale Beschreibung der Informationsverarbeitung technischer Systeme wird nicht adressiert. Eine Erweiterung in diese Richtung ist aber prinzipiell möglich.

3.2.2 Strukturkonzepte für eine kognitive Informationsverarbeitung

3.2.2.1 Klassische kognitive Architekturen

Kognitive Architekturen[32] werden entwickelt, um eine Modellierung von Grund auf zu vermeiden (*from scratch*). Sie stellen im Gegensatz zu vielen anderen kognitiven Theorien, die sich nur auf gewisse Aspekte der Kognition beschränken, einen ganzheitlichen Ansatz für eine kognitive Informationsverarbeitung dar. Kern einer kognitiven Architektur ist die Spezifikation der abstrakten Struktur der kognitiven Informationsverarbeitung bestehend aus nicht physikalischen (informationsverarbeitenden) Elementen, die die kognitiven Funktionen ausführen. Sie ist unabhängig von bestimmten Anwendungen. Durch die Implementierung von anwendungsbezogenem Wissen, das gespeichert, interpretiert, erneuert oder gelöscht werden kann, werden die Architekturen konkretisiert. Eine Wissensbasis ist daher ein integraler Bestandteil einer kognitiven Architektur; das darin abgelegte Wissen nicht [And97, S. 3f.], [LLR08]. NEWELL nennt daher drei Grundleistungen, die eine kognitive Architektur erfüllen muss [New90], [Sch06a]:

- Die Interaktion mit der Umgebung unter Echtzeitbedingungen.
- Der Erwerb und die Repräsentation von Gedächtnisinhalten.
- Die Auswahl von Alternativen an Entscheidungspunkten.

Der Einsatz einer kognitiven Architektur vereinheitlicht die Modellierung der kognitiven Informationsverarbeitung. Idealerweise kann diese auf Basis elementarer Mechanismen der Informationsverarbeitung erfolgen. Ihre Struktur befähigt ein System Inhalte zu speichern und anzuwenden, um vorgegebene Ziele zu verfolgen [Sch06a].

Der Großteil der klassischen kognitiven Architekturen sind sog. Produktionssysteme. Diese bestehen im Kern aus drei Komponenten: Einem Arbeitsspeicher, der aktuelle Daten enthält, einem Langzeitspeicher mit Produktionsregeln und einem Regelinterpreter. Der entscheidende Bestandteil sind die Produktionsregeln. Sie bestehen aus einer Bedingung (*wenn*) und einer Aktion (*dann),* die bei Zutreffen der Bedingung auszuführen ist. Der Arbeitsspeicher fungiert dabei als In-Out-Schnittstelle, um aktuelle Situati-

[32] Der Begriff geht auf NEWELL zurück, der ein einziges System zur Modellierung der Kognition basierend auf einer Vereinheitlichung der psychologischen Theorien forderte (*psychology unified theories of cognition*) [New90, S. 1].

onen mit den Bedingungen vergleichen zu können. Der Regelinterpreter enthält Kontrollwissen zur Priorisierung u.a. von zeitgleich eintretenden Bedingungen und zur Zerlegung in lösbare Teilprobleme [And01, S. 252ff.]. Das erste ganzheitliche Produktionssystem ist der **General Problem Solver (GPS)**. Aus heutiger Sicht kann man ernüchternd feststellen, dass der GPS ein Programm zur Mittel-Zweck-Analyse ist, in seinen Anfangsjahren Mitte der 1950er war der Anspruch ein anderer – er sollte das menschliche Problemlösen simulieren [Len02, S. 43f.].

Eine der ersten kognitiven Architekturen, die sich auch direkt aus dem GPS-Ansatz entwickelte, ist **Soar** (**S**tate, **O**perator **A**nd **R**esult). Sie entstand unter der Leitung NEWELLS, der ebenfalls den GPS mit entworfen hatte, und sollte seiner Forderung nach einer einheitlichen Theorie der Kognition gerecht werden. Bei Soar handelt es sich um ein Produktionssystem, das im Gegensatz zu anderen kognitiven Architekturen das gesamte Wissen als Produktionsregeln speichert. Die Inhalte des Arbeitsgedächtnisses gehen nach einer Zeit wieder verloren – neues Wissen kann nur über die Produktionsregeln generiert und gespeichert werden [New90], [LNR87]. Soar wurde in den letzten Jahren weiterentwickelt. Die aktuelle Version ist Soar 9. Sie erweitert die ursprünglich rein symbolische Repräsentation um eine subsymbolische[33]. Ferner wurde ein zusätzlicher Lernmechanismus und ein weiteres Langzeitgedächtnis implementiert [Lai08].

Eine weitere kognitive Architektur ist **ACT-R** (The **A**daptive **C**haracter of **T**hought – **R**ational Analysis), die aus der ACT-Theorie[34] nach ANDERSON entstand. Sie ist wie Soar ein Produktionssystem. Sie unterscheidet jedoch zwischen deklarativer und prozeduraler Wissensrepräsentation. Hierfür werden zwei unterschiedliche Wissensspeicher und entsprechende Mechanismen zur Verfügung gestellt. Ferner handelt es sich um eine hybride Architektur. Die ersten Anwendungen umfassten reine Computerprogramme. So wurde in den ersten Versionen von ACT-R und seinen Vorläufern eine physische Einbindung mit Sensorik und Aktorik nicht adressiert. Da diese aber eine wesentliche Voraussetzung für Kognition ist, wurde von BYRNE/ANDERSON die Erweiterung **ACT-R/PM** (**P**erceptual **M**otor) entwickelt. Diese erweitert die prozeduralen und deklarativen Speicher und Mechanismen um ein neues Sensor-Aktor-System. Dieses verbindet die kognitive Informationsverarbeitung mit einer Schnittstelle zum Umfeld. Eine direkte Verbindung ist nicht mehr möglich, sondern hängt immer von den Fähigkeiten des Sensor-Motor-Systems ab. Dieses Konzept wurde in die neueren Versionen von ACT-R direkt integriert. Bild 3-2 zeigt eine vereinfachte Darstellung der aktuellsten Architektur von ACT-R [AL98], [ABB+04].

---

[33] Architekturen, die symbolische sowie subsymbolische Repräsentationsformen unterstützen, werden als hybride Architekturen bezeichnet (vgl. Kap. 2.3.1).

[34] Das Akronym ACT wurde von ANDERSON über die Jahre hinweg unterschiedlich interpretiert. So stand es für „Active Control of Thought“, „The Architecture for Cognition“, „The Adaptive Control of Thought“, „The Adaptive Control Theory”, The Adaptive Character of Thought” oder “The Atomic Components of Thought” [AL98, S. 12f.].

ACT-R besteht aus mehreren austauschbaren *sensorischen* und *aktorischen Modulen*, die über spezifische Schnittstellen, sog. *Buffer*, miteinander kommunizieren. Die *zentrale Informationsverarbeitung* kann nur über diese Buffer Informationen empfangen und nicht direkt auf die Module zugreifen. So wird die komplette Kommunikation zentral gesteuert. Das eigentliche Produktionssystem ergibt sich aus dem Zusammenspiel von *prozeduralem Gedächtnis* und *deklarativem Gedächtnis*. In der zentralen Informationsverarbeitung läuft dann ein sequentieller Prozess mit den drei Teilschritten *Konfliktmengenbildung*, *Produktionsauswahl* und *Produktionsausführung* ab. So wird nach einer Produktionsregel gesucht, deren Bedingung einer aktuellen Situation in den Buffern entspricht. Wird eine passende Regel ausgewählt und ausgeführt, ändern sich die Buffer und somit die Ansteuerung des Systems. Zwar kann immer nur eine Produktion ausgeführt werden, doch folgt einer Produktion immer direkt eine weitere. Aus diesem Ablauf ergibt sich dann das Gesamtverhalten [ABB+55], [AL98].

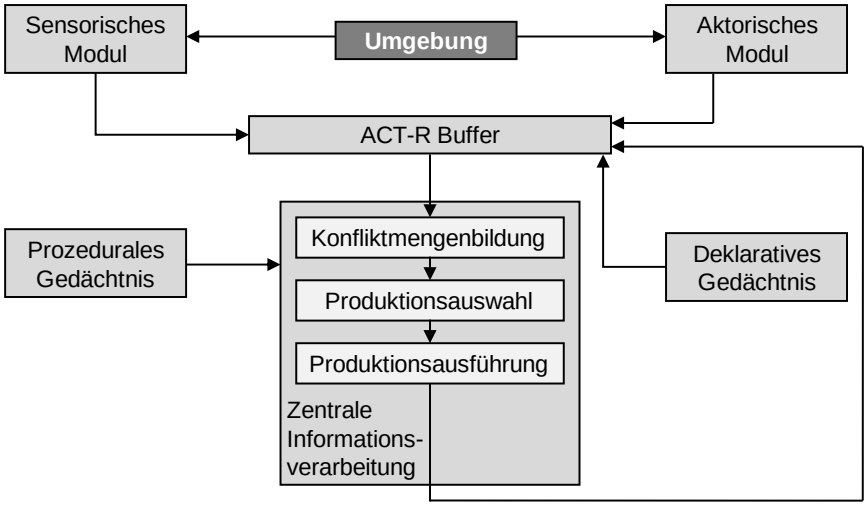


Bild 3-2: Vereinfachte Darstellung von ACT-R in Anlehnung an [ABB+04, S. 1037]

Bewertung: Klassische Architekturen wie die vorgestellten Architekturen Soar oder ACT-R versuchen die menschliche Kognition nachzubilden. Beide Ansätze werden bis heute noch weiterentwickelt und teilweise auch in technischen Systemen eingesetzt. Hervorzuheben ist, dass Erweiterungen, insbesondere von ACT-R, immer auf neue Erkenntnisse in der Kognitionswissenschaft zurückzuführen sind. Dennoch fokussieren sie in erster Linie die Modellierung der kognitiven Ebene und vernachlässigen eine vollständige Betrachtung der informationsverarbeitenden Prozesse wie z.B. die Regelungstechnik. Sie werden somit nur in Ansätzen der Anforderung einer Koexistenz von reaktiver und kognitiver Informationsverarbeitung wie sie STRUBE fordert gerecht.

3.2.2.2 Kognitive Architekturen für die Robotik

Neuere kognitive Architekturen gehen einen anderen Weg als den der vorgestellten klassischen Architekturen. Primär wird nicht versucht, die menschliche Kognition nachzubilden, sondern die Architektur für eine spezielle Anwendung zu optimieren. Dabei

handelt es sich in erster Linie um Anwendungen aus dem Bereich der Robotik. Eine kognitive Architektur, die diesem Ansatz folgt, ist **ADAPT** (**A**daptive **D**ynamics and **A**daptive **P**erception for **T**hought). Sie ist ein Produktionssystem, das für eine Multi-Task-Fähigkeit ausgelegt ist. Im Unterschied zu klassischen Architekturen sollen mehrere Produktionsregeln und Bewegungsabläufe parallel ausgeführt werden. Sensordaten sollen hinsichtlich des aktuell verfolgten Ziels interpretiert werden, um mehrere Ziele gleichzeitig verfolgen zu können. Hierfür werden sog. Sensor-Aktor-Schemata abgelegt. Schemata kombinieren prozedurales und deklaratives Wissen. Sie beinhalten außerdem Angaben über das Zeitintervall in dem eine Aktion ausgeführt werden muss. Ein Schema kann auch andere Schemata starten, was zu einer Vernetzung mehrere Aktionsketten führen kann. Schemata können außerdem andere Schemata löschen, abbrechen oder verändern. Es existiert ein übergeordnetes Schema („Interagiere mit der Umwelt“), das alle Schemata zusammenhält und zur Generierung neuer aktiver Ziele führt [BLL04].

Eine weitere kognitive Architektur für die Robotik entstand im Rahmen des **SFB 588**[35]. Diese wurde vorrangig für humanoide Roboter entwickelt. Es ist eine hybride Architektur mit dem Ziel, die kognitiven Funktionen nach STRUBE in Robotersystemen zu realisieren. Bild 3-3 zeigt den grundsätzlichen Aufbau der Architektur [BMS+05].

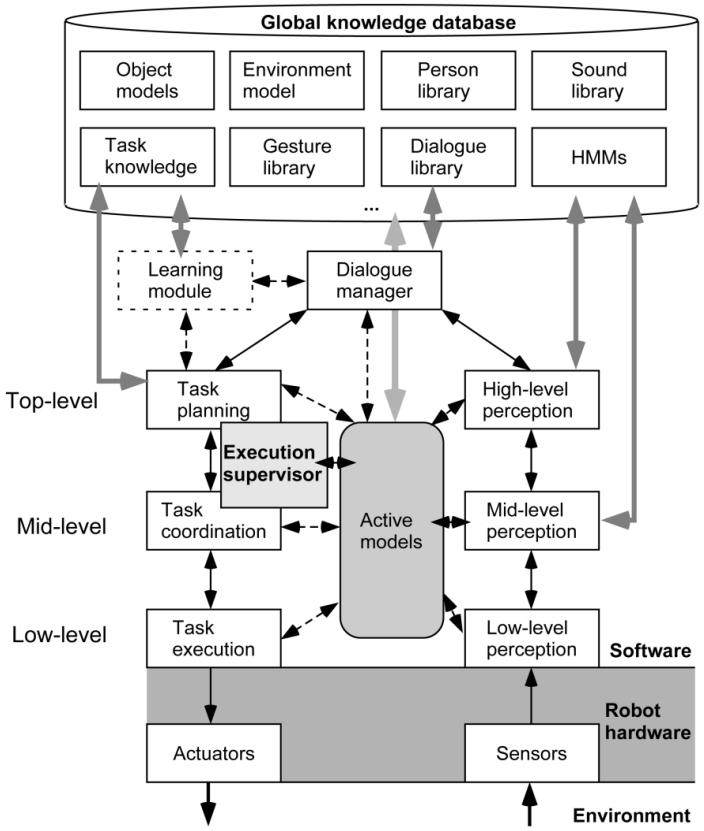


Bild 3-3: Kognitive Architektur für einen humanoiden Roboter [BMS+05]

[35] Der Sonderforschungsbereich 588 „Humanoide Roboter – Lernende und kooperierende multimodale Roboter“ wird von der Deutschen Forschungsgemeinschaft (DFG) seit 2001 gefördert.

Kern sind die sog. *Active Models*. Es handelt sich dabei um eine Art Kurzzeitarbeitsspeicher, der sich verschiedener Modelle aus der *Global Knowledge Database* bedient. Er ist mit der Mehrzahl der anderen Architekturbestandteile verbunden und enthält Daten über das Umfeld und den eigenen Systemzustand sowie die aktuell aktiven Systemziele. Die Informationsverarbeitung gliedert sich in drei Ebenen [BMS+05]:

- **Low-Level:** Hier befinden sich die schnellen, reaktiven Komponenten. Die *Low-Level Perception* beinhaltet schnelle Interpretationsmethoden für die Sensordaten. Die Daten werden für die Regelung des Systems genutzt und über die Active Models an die höhere Ebene übergeben. In der *Task Execution* werden die vorgegeben Bewegungstrajektorien ausgeführt. Hier werden sämtliche regelungstechnischen Aufgaben hinsichtlich der Aktoren ausgeführt.
- **Mid-Level:** Auf dieser Ebene analysiert und fusioniert die *Mid-Level Perception* die vorverarbeiteten Daten. Die fusionierten Daten werden mit bereits vorhandenen Daten aus der globalen Datenbank hinsichtlich Sprach- und Objektwiedererkennung abgeglichen. Die *Task Coordination* überwacht, koordiniert und adaptiert die Abläufe auf dem Low-Level.
- **Top-Level:** Während der Low-Level als auch der Mid-Level unter harter Echtzeit ablaufen, arbeiten die Elemente auf dieser Ebene unter weichen Echtzeitbedingungen[36]. Die *High-Level Perception* führt eine Situationsanalyse aus. Hier werden Aktionen des Benutzers erkannt und Kommunikationssignale wie Gesten oder Sprache interpretiert. Der *Task Planner* kann den Betrieb unterbrechen und eine Neuplanung veranlassen. Zudem delegiert er Entscheidungen direkt an die Elemente des Mid-Levels. Er erstellt globale sowie lokale Pläne für einzelne Subsysteme. Die Pläne regeln, wie die einzelnen Systemkomponenten sich gegenseitig beeinflussen dürfen. Die Schnittstelle zwischen den sensorischen und aktorischen Modulen auf den unterschiedlichen Ebenen ist der *Dialogue Manager*. Er ist in der Lage, den Input des Benutzers zu erfassen und in den Kontext einzuordnen. Die finale Ablaufplanung übernimmt der *Execution Supervisor*. Er bestimmt auch die Ressourcenverteilung. Das *Learning Module* kann in der globalen Datenbank neue Daten ablegen bzw. vorhandene Modelle korrigieren.

Bewertung: Die vorgestellten kognitiven Architekturen zielen stärker auf eine Umsetzung in einem technischen System ab, auch wenn primär die Robotik im Vordergrund steht. ADAPT bietet mit den Sensor-Aktor-Schemata ein interessantes Konzept zur Handlungssteuerung, eine ausgeprägte Struktur inkl. Beschreibung der wesentlichen Elemente für eine kognitive Informationsverarbeitung fehlt. Dagegen erfüllt die kogni-

[36] Der Begriff Echtzeit stammt aus der Informatik und verlangt, dass das Ergebnis einer Berechnung vor einer fest definierten Zeitschranke vorliegt. Harte Echtzeit erfordert das strikte Einhalten der Zeitschranke. Weiche Echtzeit setzt ebenfalls die Einhaltung der Zeitschranke voraus. Im Gegensatz zur harten Echtzeit bleibt die Nichteinhaltung ohne gravierende Konsequenzen [Kop97, S. 2ff.].

tive Architektur des SFB 588 diesen Punkt weitestgehend. Zudem überzeugt die Strukturierung der Informationsverarbeitung, die der kognitionswissenschaftlichen Forderung nach einer Koexistenz von kognitiver und reaktiver Regelung entspricht. Jedoch ist ein Lernprozess nur offline vorgesehen. Auch das Konzept der Active Models hebt die klare Strukturierung der Informationsverarbeitung in Teilen wieder auf.

3.2.2.3 Psychologieorientierte Architekturen

Eine weitere Kategorie kognitiver Architekturen sind sog. psychologieorientierte Architekturen. Ihr Anspruch ist eine umfassende und ganzheitliche Theorie des menschlichen Verhaltens. Die verschiedenen psychischen Prozesse werden nicht separat betrachtet. Denn erst durch die Interaktion und die Abhängigkeit der geistigen Vorgänge sind ausgedehnte Denkprozesse wissenschaftlich erklär- und fundierbar. Dabei handelt es sich im Wesentlichen um motivationale, kognitive und emotionale Abläufe, die in einem perzeptuellen und motorischen Gefüge ganzheitlich abzubilden sind [DSS01, S. 1f.].

Die **PSI-Theorie** (Theorie der **P**ersönlichkeits-**S**ysteme-**I**nteraktionen) verfolgt diesen ganzheitlichen Ansatz. Sie wurde maßgeblich von DÖRNER konzipiert und auch als Computersimulation umgesetzt. Sie besagt, dass zwischen der Perzeption von Informationen aus der Umgebung und der Handlungsausführung eines Systems verschiedene interne Prozesse durchlaufen werden. Dabei geht DÖRNER von mehreren Regelsystemen aus, in denen die systemeigenen Parameter eingestellt werden können. Für den Fall, dass innerhalb einer dieser Regelungen eine Abweichung des Sollwertes vorherrschen sollte, wird versucht dieser entgegenzuwirken. Nach DÖRNER heißt dies, dass ein Bedarf entstanden ist, der nicht von alleine wieder ausgeregelt wird. Dies führt zu einem Motiv, dem die Informationen über den Bedarf als Bedürfnis mitgeteilt werden. Aus Bedarf, Bedürfnis und Motiv entsteht eine Absicht des Systems, die die Handlung entsprechend der eigenen Gedächtnisinhalten steuert. DÖRNER entwirft auf Basis seiner Theorie das künstliche Wesen Ψ (PSI), dessen Struktur und Informationsflüsse in Bild 3-4 dargestellt sind [DSD01], [Dör01].

Bei Ψ handelt es sich ursprünglich um ein Computerprogramm einer „Dampfmaschine“, das in einer Inselsimulation „ausgesetzt“ wird. Ziel ist es, Energieelemente (sog. Nukleotide) zu finden und aufzusammeln. Ψ muss hierfür die Insel erkunden und neben dem Aufsammeln von Nukleotiden dafür sorgen, dass es nicht durch Umwelteinflüsse zerstört wird. Zudem dürfen sich seine existentiellen Bedürfniskessel (Wasser, Energie) nicht komplett leeren. Zu diesem Zweck existieren Wasserstellen zum Auftanken und verschiedene Pflanzen zur Energiegewinnung. Soziale Bedürfnisse (Affiliation) können durch einen „Teddy-Kumpel“ befriedigt werden [DSD01].

Einen Schritt in Richtung technische Umsetzung geht **MicroPsi**. Dies ist eine Architektur für Miniaturroboter und umfasst eine Entwicklungs- sowie Simulationsumgebung. So können Dritte die Architektur sehr einfach anwenden und erweitern. Zur ursprünglichen Theorie nach DÖRNER bestehen nur marginale Unterschiede [Bac07, S. 223ff.].

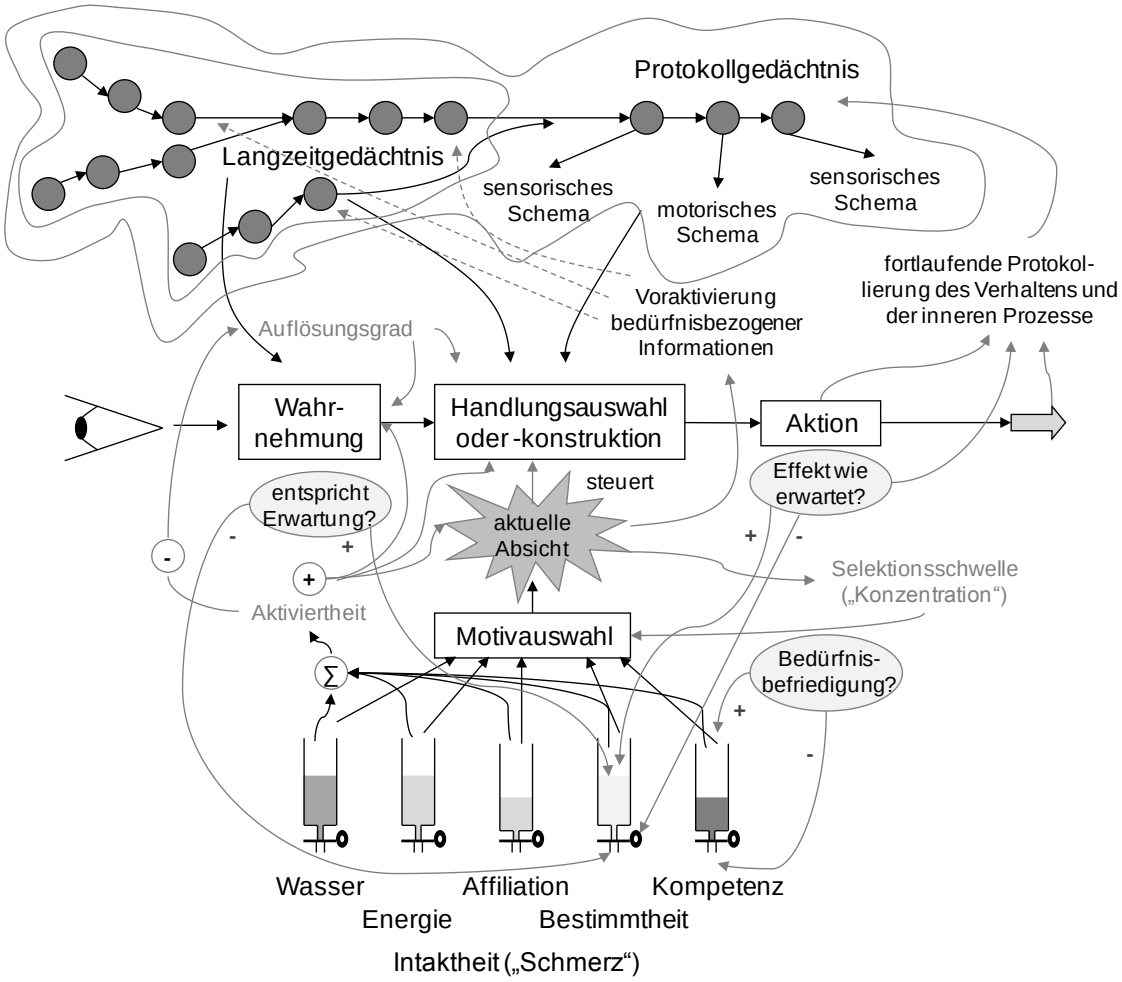


Bild 3-4: Eine Übersicht über die Struktur von Ψ nach [DSD01, S. 8]

PECS (**P**hysical conditions, **E**motional states, **C**ognitive capabilities, **S**ocial status) ist eine kognitive Architektur, die dem PSI-Ansatz stark ähnelt. PECS unterscheidet zwei Ebenen der Verhaltenssteuerung. Die reaktive Ebene realisiert eine vorprogrammierte oder erlernte Verhaltenssteuerung, wohingegen die deliberative Ebene ein konstruktives sowie reflektierendes Verhalten ermöglichen soll [Sch00, S. 24ff.]. PECS wurde als agentenbasiertes Simulationsmodell realisiert [Urb04].

Bewertung: Die erläuterten Architekturen wurden in erster Linie zur Erklärung psychischer Prozesse entworfen. Die Architektur von PSI ist aus psychologischer Sicht hervorragend fundiert und geht sogar über NEWELLS Forderung nach einer einheitlichen Theorie der Kognition hinaus, indem Aspekte der Motivation und Emotion integriert werden. DÖRNER betont, dass PSI gerade die Phänomene des alltäglichen Lebensablaufs des Menschen für die Psychologie offen legt. PSI ist daher nur bedingt auf technische Systeme übertragbar und beschränkt sich auf eine reine subsymbolische Realisierung. PECS fokussiert noch mehr das soziale Umfeld und strukturiert die Verhaltenssteuerung ähnlich dem Dreischichtenmodell nach STRUBE. Der Kern, wie z.B. die Motiv- und Handlungsselektion, entspricht in weiten Teilen der PSI-Theorie.

3.2.2.4 Operator-Controller-Modul

Im SFB 614 wurde mit dem Operator-Controller-Modul (OCM) eine Architektur für die Informationsverarbeitung selbstoptimierender Systeme entwickelt. Die Arbeiten basieren auf dem Konzept von NAUMANN, der die erste Version des OCM für vernetzte mechatronische Systeme definierte (Bild 3-5) [Nau00, S. 27ff.], [ADG+, S. 13ff.].

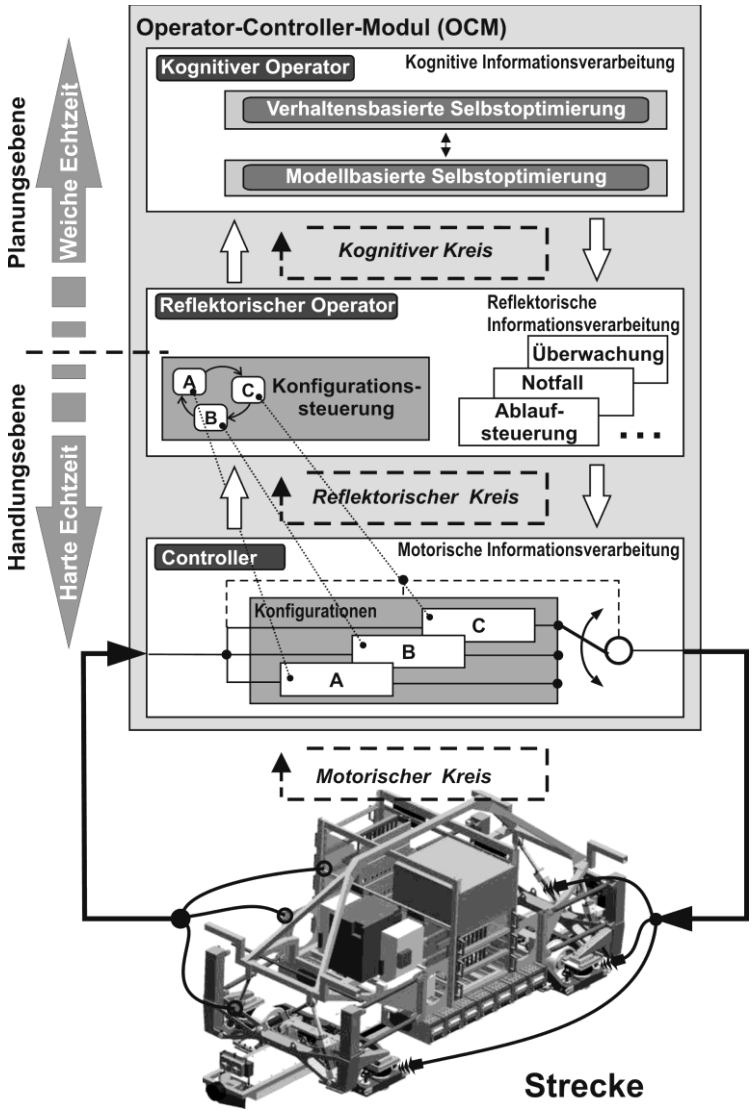


Bild 3-5: Struktur des Operator-Controller-Modul (OCM) [ADG+09, S. 14]

Der **Controller (CO)** bildet die regelungstechnische sowie unterste Ebene des OCM. Seine Aufgabe ist es, das dynamische Verhalten des Grundsystems in gewünschter Weise zu beeinflussen. Über den *motorischen Kreis* stellt er eine direkte Verbindung zwischen dem Grundsystem und der regelnden Informationsverarbeitung her. Die Schnittstellen dieser Verbindung sind wie im mechatronischen Regelkreis die Sensorik und Aktorik. Der CO arbeitet unter harten Echtzeitbedingungen.

Der **reflektorische Operator (RO)** ist über den *reflektorischen Kreis* mit dem Controller verbunden. Er überwacht und steuert diesen. Über Parameter und Strukturänderun-

gen modifiziert er den CO, greift aber nicht direkt auf die Aktorik des Systems ein. Für Strukturänderungen besitzt der RO eine Konfigurationssteuerung, die zwischen Konfigurationen im CO umschalten kann. Diese sind eine Kombination aus Reglern und Schaltelementen, die über Signalflüsse miteinander verbunden sind. Ferner übernimmt der RO Funktionen wie Ablaufsteuerung, Überwachungs- und Notfallprozesse. Aufgrund der Verknüpfung zum CO arbeitet die Informationsverarbeitung in diesem Bereich des RO ebenfalls unter harten Echtzeitbedingungen. Gleichzeitig agiert der RO als Schnittstelle für Informationen und Messwerte zwischen CO und kognitivem Operator.

Aufgabe des **kognitiven Operators (KO)** ist die Umsetzung der kognitiven Informationsverarbeitung zur Realisierung der Selbstoptimierung (vgl. Kap. 2.2.3). Auf Basis unterschiedlicher Verfahren kann zuvor gewonnenes Wissen zur Verbesserung des Systemverhaltens genutzt werden. Zum Einsatz kommen dabei Planungs- und Lernverfahren, modellorientierte Optimierungsverfahren sowie wissensbasierte Systeme[37]. Aufgrund der zeitlichen Entkoppelung der Verfahren zum Verhalten des realen Systems arbeitet die Informationsverarbeitung in diesem Bereich unter weichen Echtzeitbedingungen.

Die beteiligten Domänen zur Umsetzung einer derart komplexen Informationsverarbeitung sind die Regelungstechnik, die Softwaretechnik, die höhere Mathematik sowie Techniken der künstlichen Intelligenz. Für den CO und für den RO hat HESTERMEYER jeweils einen charakteristischen Aufbau beschrieben. Diese bestehen aus vordefinierten Systemelementen sowie zeit- und ereignisdiskreten Signalflüssen [Hes06, S. 37ff.].

Bewertung: Die Grundstruktur des OCM entspricht im Wesentlichen dem Grundaufbau des Dreischichtenmodells der Verhaltenssteuerung nach STRUBE und eignet sich daher prinzipiell zur Integration kognitiver Funktionen in technischen Systemen. Sie berücksichtigt, dass neben einer kognitiven Informationsverarbeitung noch weitere Ebenen vorhanden sein müssen. Der CO, RO und KO interagieren dafür stark, werden aber nur nach Echtzeitbedingungen strukturiert. Eine systematische Zuordnung von Funktionen und Verfahren fehlt. Zudem fehlt ein Konzept zur frühzeitigen Spezifikation der beteiligten Systemelemente und der Informationsflüsse, insbesondere für den KO.

3.2.2.5 Architektur des CoTeSys

RODRIGUEZ et al. haben im Zuge des Exzellenzclusters CoTeSys ein Konzept einer kognitiven Architektur für technische Systeme entwickelt (vgl. Kap. 3.1.1). Sie soll eine Rahmenstruktur bzw. Integrationsplattform für verschiedene Module sein, die im Exzellenzcluster erarbeitet werden. Mit Modulen sind Komponenten zur Umsetzung der Perzeption, Kognition und Handlungsausführung im Sinne einer kognitiven Informati-

[37] Der Begriff „wissensbasiertes System“ wird durch die Softwaretechnik geprägt. Durch die Trennung der Wissensbasis und der Wissensverarbeitung ermöglichen derartige Systeme sowohl die Abbildung als auch die Nutzbarkeit vorhandenen Wissens [BK06].

onsverarbeitung gemeint. Die kognitive Informationsverarbeitung wird von CoTeSys als eine sog. P-C-A-Schleife (Perception-Cognition-Action) definiert. Da die notwendigen Module in der Regel in unterschiedlichen Arbeitsgruppen und teilweise unabhängig voneinander entstehen, ist es das Ziel der Architektur, diese Module gemeinsam und ohne großen Aufwand in bestmöglicher Performance zu betreiben. Hierfür werden aber auch feste Bestandteile definiert. Bild 3-6 zeigt das Konzept der Architektur, deren Module durch konkrete Umsetzungen ausgeprägt werden können (z.B. ein 3D Trackingsystem als ein Perception Module) [RFK08].

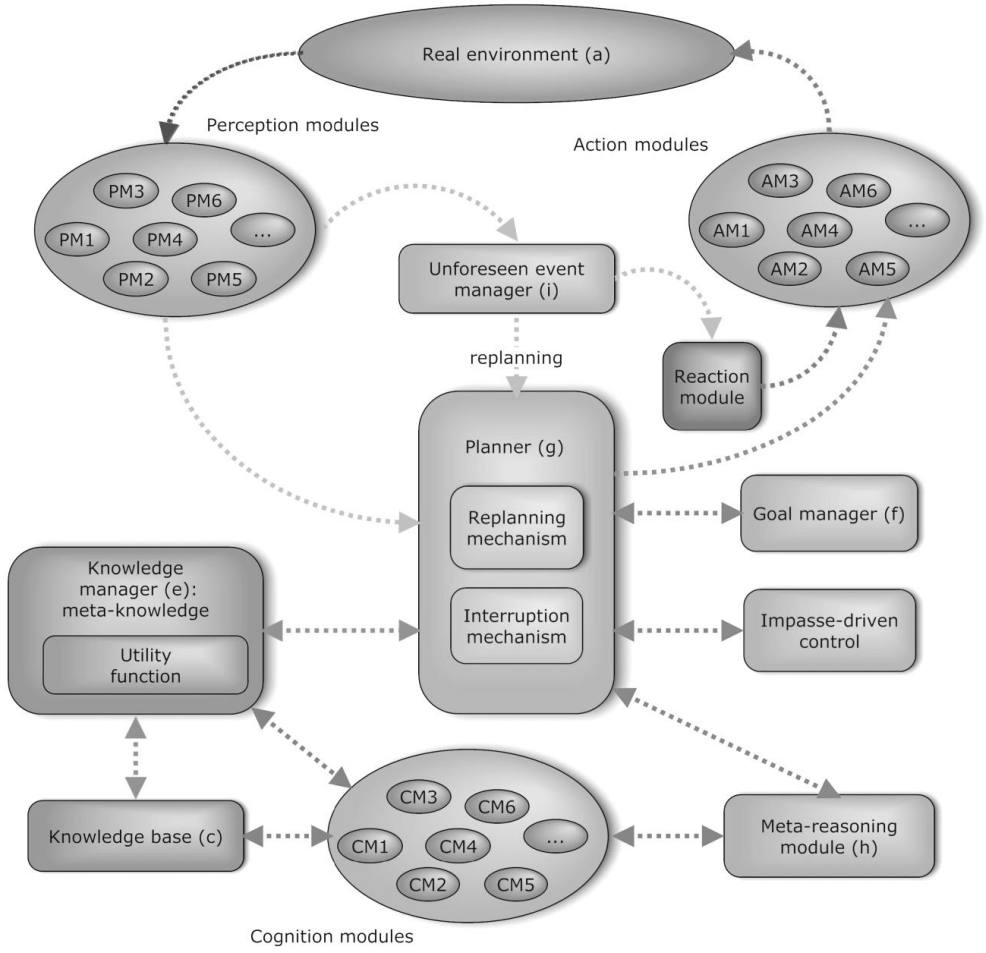


Bild 3-6: Konzept der CoTeSys-Architektur [RFK08]

Zwischen den einzelnen Modulen und den festen Bestandteilen der Architektur sind als Pfeile die zentralen Informationsflüsse eingezeichnet. Die von den **Perception Modules** aus dem realen Umfeld aufgenommenen Daten werden an den Planner oder bei unvorhergesehenen Ereignissen an den **Unforeseen Event Manager** weitergeleitet. Dieser aktiviert zunächst einen *Interruption Mechanism*, der die Wichtigkeit des Ereignisses bewertet und entscheidet, ob der aktuelle P-C-A-Prozess unterbrochen werden muss. Wird der Zwischenfall hingegen als systemkritisch eingeschätzt, aktiviert der Unforeseen Event Manager das **Reaction Module** ohne vorher mit der kognitiven Einheit zu kommunizieren. Dies garantiert eine möglichst schnelle Reaktion.

Die kognitive Einheit wird im Wesentlichen durch den **Planner** repräsentiert. Dieser koordiniert alle Pläne zur Erfüllung der Systemziele. Hierfür kann er Pläne korrigieren (*Replanning Mechanism*). Die zu erfüllenden Zielen speichert das System im **Goal Manager**. Es werden globale Ziele, wie z.B. „Sicherheit“ und lokale Ziele, wie z.B. „greife dieses Objekt“ unterschieden. Globale Ziele sind in der Regel höher priorisiert.

Der Planner steht in direkter Verbindung mit dem **Knowledge Manager**. Dieser entscheidet mittels einer geeigneten *Utility Function*, welches Wissen in der Knowledge Base abgelegt wird. Ein zentraler Punkt ist, dass sog. *Meta-Knowledge*, also Wissen über Wissen, extrahiert wird, um die Wissensmenge nicht unnötig aufzublähen. Die **Knowledge Base** ist die globale Wissensbasis des Systems, die allen Komponenten zugänglich ist. Um unterschiedliche Module integrieren zu können, sollen alle möglichen Arten von Wissensformaten gespeichert werden können.

Da nicht alle Prozesse die volle Rechenleistung benötigen und unterschiedliche Prozesse auch verschiedene Prioritäten haben, verwaltet das **Meta-Reasoning Module** die Prozessorleistung. Diese ist von entscheidender Bedeutung, wenn verschiedene, spezialisierte kognitive Module eine P-C-A-Schleife ausführen.

Die **Impasse-Driven Control** wird bei einem Systemstillstand aktiviert. Dieser kann bspw. durch einen Mangel an Informationen hervorgerufen werden. In einem solchen werden alle extern zugeschalteten Module kurzzeitig deaktiviert und die Architektur bestimmt eigenständig die nächste Systemhandlung.

Bewertung: Primäres Ziel der CoTeSys-Architektur ist die Integration mehrerer sowie verschiedener kognitiver Module zur Durchführung der P-C-A-Schleife. Die drei Ebenen der kognitiven Informationsverarbeitung nach STRUBE sind nur in Ansätzen realisiert. Eine direkte, reaktive Kopplung zwischen Sensorik und Aktorik ist nur implizit vorhanden, wobei die regelungstechnische Umsetzung überhaupt nicht adressiert wird. Ferner fehlt eine konkrete Spezifikation der Informationsflüsse inkl. einer Unterscheidung zwischen Signalen, Daten, Informationen und Wissen. Diese werden stellenweise in der Architektur gleichgesetzt. Ungeklärt bleibt, wie konkrete kognitive Module aussehen und ob diese eine einheitliche Struktur haben.

3.2.3 Sprachen zur Beschreibung der Informationsverarbeitung

3.2.3.1 UML – Unified Modeling Language

UML (Unified Modeling Language) ist eine semiformale, grafische Sprache, die im Rahmen der Softwareentwicklung angewendet wird. Sie wird von der Object Management Group (OMG) entwickelt. Die Version 1.4.2 wurde in der ISO/IEC 19501 standardisiert [ISO19501]. Ihr Einsatz reicht von der Modellierung bis hin zur Analyse von Softwareprogrammen. Mit ihr ist es möglich, Strukturen, Architekturen, Verhalten von Systemen sowie Interaktionen zu weiteren Systemen durch Diagramme abzubilden

[For07], [Oes05]. Wesentliche Bestandteile dieser Diagramme sind Objekte und Klassen. Sie beinhalten spezifische Attribute und Methoden und bilden so die Grundlage dieser Modellierungssprache (Bild 3-7).

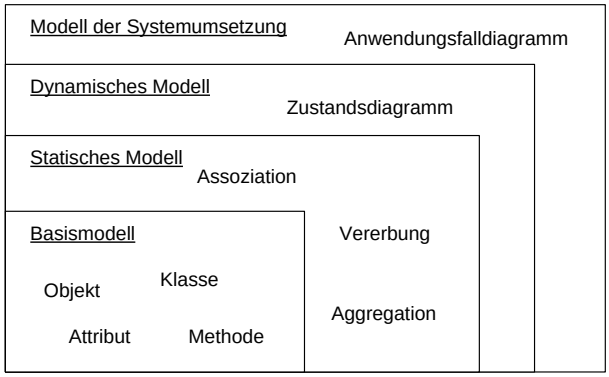


Bild 3-7: Gliederung des objektorientierten Modellverständnisses nach [For07, S. 21]

Ein **Objekt** ist im Allgemeinen eine grafische Repräsentation eines Gegenstandes des Interesses, insbesondere einer Beobachtung, Untersuchung oder Messung. Objekte können Modellierungen realer Sachverhalte, Dinge oder Begriffe sein (vgl. Bild 3-8: Modellierung einer Person im Kontext einer Universität). Sie besitzen bestimmte Eigenschaften und reagieren mit einem vorgegebenen Verhalten auf definierte Anfragen. Außerdem besitzt jedes Objekt eine Identität, die es von allen anderen Objekten unterscheidet. Eine **Klasse** beschreibt eine Sammlung von Objekten mit gleichen Eigenschaften (Attributen), gemeinsamer Funktionalität (Methoden), gemeinsamen Beziehungen zu anderen Objekten und gemeinsamer Semantik. **Attribute** beschreiben die Eigenschaften eines Objektes bzw. einer Klasse. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch unterschiedliche Attributwerte. Eine **Methode** ist ein Algorithmus. Dieser ist jedem Objekt zugeordnet und kann von diesem abgearbeitet werden. Das Verhalten eines Objektes wird durch eine bestimmte Menge von Methoden ausgedrückt.

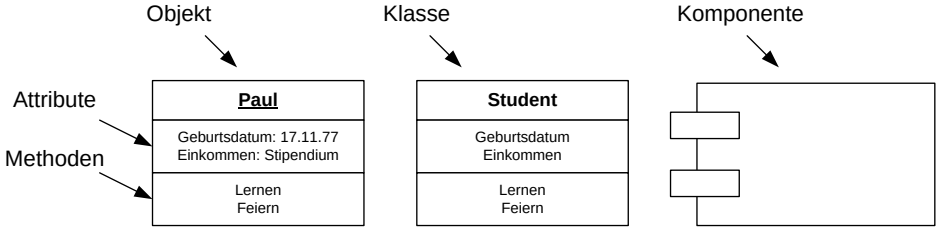


Bild 3-8: Darstellung eines Objektes (links), einer Klasse (mittig) und einer Komponente in UML-Notation in Anlehnung an [For07, S. 20]

Da Softwareprogramme im Allgemeinen sehr komplex sind, bietet es sich an, ausführbare und austauschbare Einheiten zu definieren. Diese Einheiten werden als **Komponenten** bezeichnet und verfügen über festdefinierte Schnittstellen und einer eigenen Identität. Intern besteht eine Komponente in der Regel aus einer Menge von Klassen, die das Verhalten realisieren.

Bewertung: Die Anwendung von UML ist fester Bestandteil der objekt- bzw. komponentenbasierten Softwareentwicklung. Mit dieser Modellierungssprache ist es möglich, alle notwendigen Aspekte eines Softwareprogramms abzubilden. Neben der Repräsentation sämtlicher Modelle existieren feste Definitionen aller Schnittstellen und Verbindungen. Sofern die Entwicklung der Informationsverarbeitung kognitiver technischer Systeme objektorientiert erfolgt, eignet sich diese Sprache zur Abbildung der Vorgänge auf assoziativer und kognitiver Ebene. Die reaktive Ebene der Regelungstechnik wird nicht unterstützt. Zudem kann die UML trotz ihres Anspruches nicht als domänenübergreifende Sprache in der Systementwicklung angesehen werden.

3.2.3.2 SysML – Systems Modeling Language

Während die UML für die Softwareentwicklung definiert wurde, adressiert die SysML das Systems Engineering und somit die ganzheitliche sowie disziplinübergreifende Modellierung technischer Systeme. Sie ist eine semiformale, grafische Sprache zur Modellierung, Analyse und Verifikation von Systemen, die in der aktuellen Version 1.2 auf UML 2.3 basiert. Hierzu wurden einige Anpassungen und Erweiterungen zur UML vorgenommen; so werden z.B. Klassen als Blöcke bezeichnet. Da sich Aufbau und Konzepte sehr ähnlich sind, können vorhandene UML-Werkezuge angepasst werden. Um den Umfang der Sprache nicht unnötig aufzublähen, wurden irrelevante Bestandteile der UML gestrichen. Die Sprache wurde von der OMG im April 2006 als Standard anerkannt und in der Version 1.0 Anfang 2007 offiziell veröffentlicht [Wei06, S. 157ff.].

Mittels verschiedener Diagramme ermöglicht SysML die Beschreibung der Aspekte Struktur, Anforderungen und Verhalten. Bild 3-9 zeigt in Form eines Blockdefinitionsdiagramms (bdd) die unterschiedlichen Diagrammtypen, von denen einige unverändert aus der UML übernommen und andere erweitert sowie teilweise umbenannt wurden. Das Anforderungsdiagramm sowie das Zusicherungsdiagramm sind neu hinzugekommen.

Blöcke beschreiben die **Struktur** und relevante Strukturkonfigurationen sowie Eigenschaften des zu entwickelnden Systems. Es kann sich um ein informationsverarbeitendes oder physikalisches Element handeln. Der Begriff Einheit ist auf ein physikalisches Element beschränkt. *Blockdefinitionsdiagramme* beschreiben die Beziehung zwischen verschiedenen Blöcken, ihre Assoziationen, Generalisierungen sowie Abhängigkeiten. *Interne Blockdiagramme* legen die Beziehung zwischen den Bestandteilen eines Blocks durch Ports, Konnektoren und Flüsse fest. Hierbei können mit *Zusicherungsdiagrammen* die Beziehungen zwischen Eigenschaften verschiedener Blöcke definiert werden. So können parametrische Beziehungen (z.B. physikalische Gesetze) modelliert werden.

Anforderungsdiagramme ermöglichen die Beschreibung der funktionalen und der nicht funktionalen **Anforderungen**. Letztere kann die UML nicht leisten. Zudem können die bestehenden Beziehungen zwischen den Anforderungen spezifiziert werden. Es werden Ableitungs-, Enthält-, Erfüllungs-, Kopie-, Prüf-, Verfeinerungs- und Verfolgungsbe-

ziehungen unterschieden. Die Modellierung erfolgt zwar vorrangig grafisch, eine Tabellennotation zur in der Praxis üblichen textuellen Darstellung ist aber vorhanden.

Das **Verhalten** kann in der SysML mit vier unterschiedlichen Diagrammtypen modelliert werden. Mit Hilfe von *Anwendungsfalldiagrammen* lassen sich die Interaktionen aller Benutzer oder externer Geräte mit dem zu entwickelnden System beschreiben. *Zustandsdiagramme* stellen mögliche Zustände des Systems und Zustandsübergänge dar. In *Sequenzdiagrammen* werden Szenarien beschrieben und wie sich in diesen die Systembestandteile im Zusammenspiel verhalten sollen. Während diese drei Verhaltensdiagramme nahezu unverändert aus der UML entnommen sind, wurde das *Aktivitätsdiagramm* erweitert. In diesem werden die Systemabläufe inkl. Ein- und Ausgabedaten beschrieben. Des Weiteren ist eine Dekomposition der Aktivitäten möglich, um eine Art Funktionshierarchie abzuleiten.

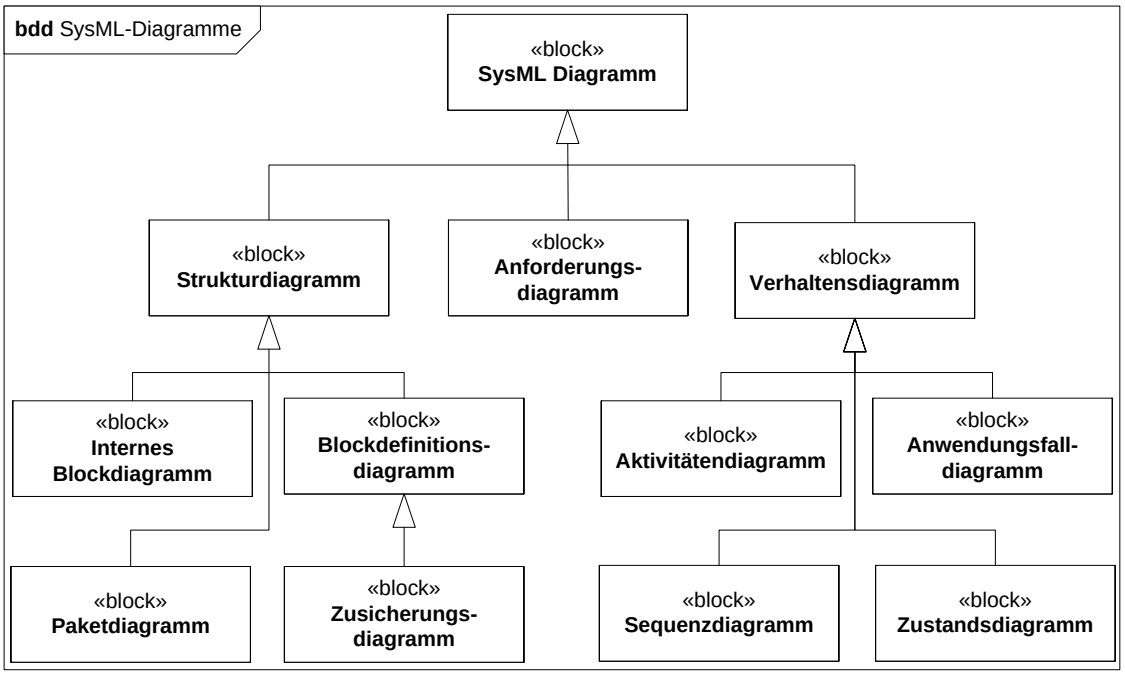


Bild 3-9: Diagramme der SysML nach [Wei06, S. 160]

Bewertung: SysML kann den kompletten Entwurf eines technischen Systems abbilden. Sie beschränkt sich dabei auf drei Aspekte. Der enorme Umfang an Diagrammen und Konstrukten, deren Verwendung zum Teil nur grob vordefiniert ist, lässt nicht immer eine eindeutige oder gar intuitive Verwendung zu. Hinzu kommt, dass die Vielzahl der Konstrukte teilweise beliebig einsetzbar ist. Die Einarbeitung ist daher mit einem hohen Aufwand verbunden. Die Spezifikation fortgeschrittener mechatronischer Systeme und deren kognitive Informationsverarbeitung, die zu einer Verhaltensänderung führt, werden nicht direkt adressiert. Im Hinblick auf den im Fokus stehenden Systementwurf ist eine funktionale Dekomposition nur mit Mehraufwand möglich. Die so abgeleitete Funktionshierarchie ist aber primär nicht zur Lösungsfindung gedacht.

3.2.3.3 Spezifikationstechnik des SFB 614

Im Rahmen des SFB 614 entstand eine Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme. Sie basiert auf den Arbeiten von FRANK, GAUSEMEIER und KALLMEYER und ermöglicht eine intuitive und domänenübergreifende Beschreibung fortgeschrittener mechatronischer Systeme [Fra06], [GEK01], [Kal98]. Es handelt sich um eine semiformale Spezifikationstechnik, die sich in die acht Aspekte Umfeld, Anwendungsszenarien, Anforderungen, Funktionen, Wirkstruktur, Verhalten, Gestalt und Zielsystem gliedert (vgl. Bild 3-10). Die rechnerinterne Abbildung dieser Aspekte ergibt jeweils ein Partialmodell. Diese und die partialmodellübergreifenden Beziehungen resultieren in einer konsistenten und ganzheitlichen Beschreibung der Prinziplösung. Die Aspekt, respektive die Partialmodelle, können in einer bestimmten Reihenfolge erarbeitet werden. Dabei sind etliche Iterationen durchzuführen, um gerade die partialmodellübergreifenden Beziehungen zu beschreiben. Im Folgenden werden die wesentlichen Inhalte der acht Partialmodelle dargestellt [ADG+09], [GFD+08b].

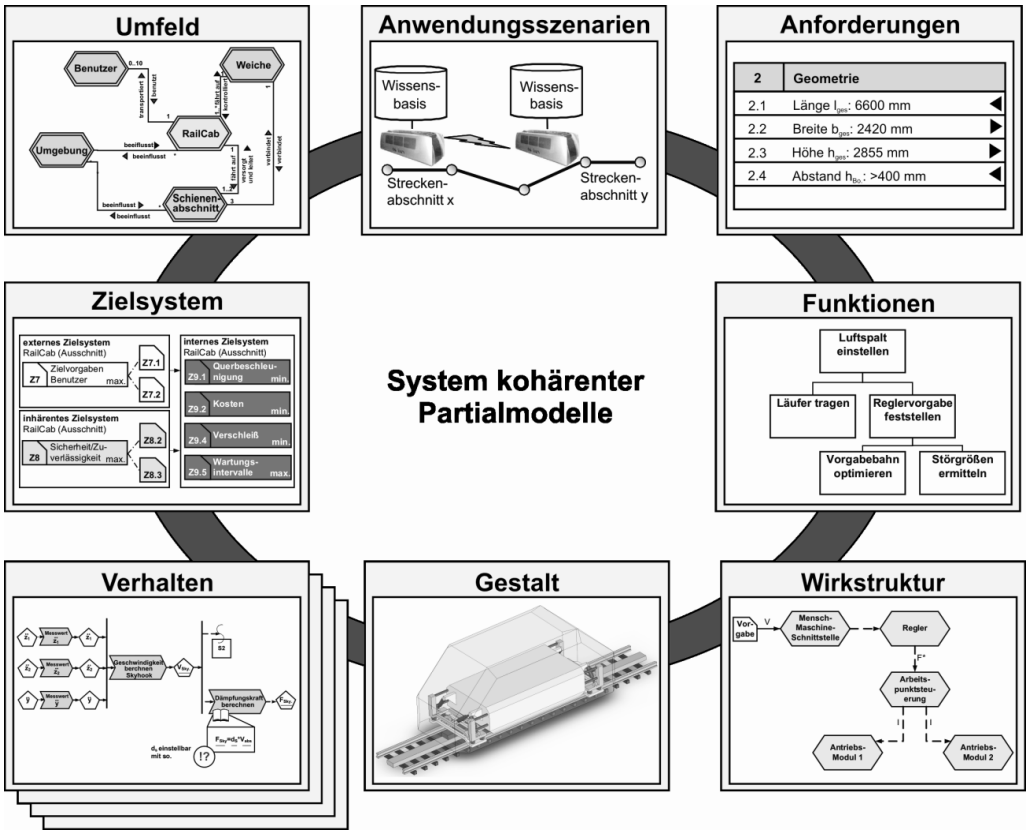


Bild 3-10: Partialmodelle zur domänenübergreifenden Beschreibung der Prinziplösung selbstoptimierender Systeme nach [Fra06, S. 80]

In einer Black-Box-Darstellung des zu entwickelnden Systems wird in einem ersten Schritt dessen **Umfeld** und Wechselwirkungen mit diesem beschrieben. Ziel ist es, die relevanten Einflussbereiche (z.B. übergeordnete Systeme) und Einflüsse bzw. Störgrößen (z.B. Wärmestrahlung) auf das System aufzuzeigen. Zusätzlich werden Wechsel-

wirkungen einzelner Einflüsse identifiziert. Eine konsistente Menge von gemeinsam auftretenden Einflüssen ergibt eine Situation, für die das System auszulegen ist.

Anwendungsszenarien beschreiben in welcher Art und Weise sich das System in einem Zustand oder einer bestimmten Situation verhalten soll. Zusätzlich enthalten sie Angaben, durch welche Ereignisse Zustandsübergänge stattfinden sollen. Anwendungsszenarien werden zunächst prosaisch beschrieben und im Laufe der Konzipierung vor allem durch die Partialmodelle Funktionen, Wirkstruktur und Verhalten konkretisiert.

Sämtliche **Anforderungen** an das zu entwickelnde System werden in einer Anforderungsliste zusammengetragen. Diese bildet die Grundlage der späteren Entwicklung und ist fortlaufend zu pflegen. Die Anforderungen werden dabei prosaisch beschrieben und ggf. durch Attribute und deren Ausprägungen konkretisiert.

Die Darstellung der **Funktionen** erfolgt hierarchisch mit der Gesamtfunktion am Kopf und den Teilfunktionen darunter. Das Ergebnis ist eine lösungsneutrale Abbildung der grundsätzlichen Funktionalität des zu entwickelnden Systems.

Die **Wirkstruktur** ist das zentrale Modell der Spezifikationstechnik des SFB 614. Mit ihr werden die Systemelemente und deren Beziehungen eines Systems repräsentiert. Sie bildet daher die gesamte Struktur inkl. aller vorausgedachten Systemkonfigurationen eines selbstoptimierenden Systems feingegliedert ab. Im Laufe der Entwicklung wird sie zum Dreh- und Angelpunkt vieler partialmodellübergreifender Beziehungen.

Das **Verhalten** unterteilt sich in drei Partialmodelle, die jeweils eine unterschiedliche Art von Verhalten beschreiben. *Verhalten – Zustände* bilden dabei sämtliche vorausgedachten und zu berücksichtigenden Systemzustände und Zustandsübergänge ab. Die dabei auftretenden Ablaufprozesse werden hingegen als *Verhalten – Aktivitäten* modelliert. Mit ihnen können auch die für die Selbstoptimierung typischen Anpassungsprozesse spezifiziert werden (vgl. Kap. 2.2.3). Eine *Verhalten – Sequenz* bildet die Wechselwirkungen zwischen mehreren Systemelementen ab. In einer chronologischen Reihenfolge werden gewünschte Systeminteraktionen dargestellt.

Die **Gestalt** wird in der Regel mit einem 3D CAD-Systemen erstellt. Ein erster Wurf ist bereits während der Konzipierung zu erarbeiten, um erste Angaben über Wirkflächen, Wirkorte, Hüllflächen und Stützstrukturen zu definieren.

Die Repräsentation der internen, externen und inhärenten Ziele eines Systems sowie deren Verknüpfungen erfolgt im **Zielsystem**. Entsprechend des Selbstoptimierungsproblems kann ein Zielvektor, eine Zielhierarchie oder ein Zielgraph genutzt werden (vgl. Kap. 2.2.3). Die Beeinflussung der Ziele untereinander können in einer Einflussmatrix dargestellt werden.

Im Rahmen des Verbundprojekts VireS[38] entstand der **Mechatronic Modeller**. Er ist ein IT-Werkzeug zur effektiven und durchgängigen Modellierung der Prinziplösung mechatronischer Systeme auf Basis der Spezifikationssprache des SFB 614 [GDK10].

Bewertung: Die Spezifikationstechnik des SFB 614 bildet mit ihren Aspekten und Konstrukten eine sehr gute Basis zur Spezifikation der kognitiven Informationsverarbeitung. Durch die allgemeinverständliche Beschreibung eignet sich diese Technik bei der Integration verschiedener Fachdisziplinen. Durch eine geeignete Darstellung besteht zudem die Möglichkeit, für die frühe Phase des Entwurfs notwendiges Wissen in explizites Wissen zu überführen. Der Mechatronic Modeller unterstützt die rechnerinterne Abbildung. Offen ist, mit welchen Aspekten und in welcher Weise die kognitive Informationsverarbeitung zu beschreiben ist.

3.2.3.4 Modelica®

Modelica® ist eine objektorientierte Sprache, die im Rahmen des ESPRIT-Projekts „Simulation in Europe Basic Research Working Group“ entstand. Sie wird durch die Modelica Association weiterentwickelt und gefördert. Diese bietet die Sprache selbst und Modelica-Standardbibliothek zur freien Verfügung an. Der aktuelle Sprachstandard lautet 3.2. Es wird angestrebt, Modellierungssprachen und -ansätze wie VHDL-AMS[39] und ObjectMath[40] zu vereinheitlichen. Vorwiegend wird die Entwicklungsumgebung Dymola von Dassault Systems eingesetzt. Der Austausch von Modelldaten und die Wiederverwendung von (Teil-)Modellen werden unterstützt [Mod10].

Modelica® eignet sich in erster Linie zur Modellierung mechatronischer Systeme mit elektrischen, mechanischen, hydraulischen, pneumatischen, thermischen und regelungstechnischen Komponenten. Ziel ist die Modellierung und anschließende Simulation großer, komplexer und heterogener physikalischer Systeme. Physikalische Modelle werden mit der Sprache aufgestellt und mit Hilfe eines Modelica-Übersetzers in ein mathematisches Modell überführt. Die Simulation erfolgt in einer entsprechenden Simulationsumgebung. Der Benutzer beschreibt das System entweder mit Objektdiagrammen oder direkt auf Gleichungsebene. Mit den Objektdiagrammen werden die Komponenten des Systems durch Symbole dargestellt, durch Beziehungen miteinander verbunden und durch Parameter konkretisiert. Das Symbol repräsentiert dabei das ma-

---

[38] VireS – Virtuelle Synchronisation von Produkt und Produktionssystem, gefördert vom Bundesministerium für Bildung und Forschung (BMBF).

[39] VHDL-AMS (VHSIC Hardware Description Language – Analog and Mixed Signal Extensions) ist eine Hardware-Beschreibungssprache zur formalen und textbasierten Spezifikation digitaler und analoger Schaltungen [IEEE1076.1].

[40] ObjectMath ist eine objektorientierte Erweiterung der Computer-Algebra-Sprache „Mathematica“. Sie ermöglicht die Strukturierung mathematischer Modelle durch Gruppierung mathematischer Funktionen und Gleichungen [FHV92].

thematische Modell der Komponente, das aus den physikalischen Gesetzen resultiert. [EM98], [EMO99], [Mod10].

Bewertung: Modelica® ist in erster Linie zur Beschreibung physikalischer Systeme gedacht. Daher eignet sie sich gut zur Darstellung der Domänen Maschinenbau, Elektrotechnik und Regelungstechnik sowie zur Auslegung der physikalischen Eigenschaften des zu entwickelnden Systems. Die softwareinterne Umsetzung der Symbole erfolgt auf Basis mathematischer Gleichungen und verfolgt zudem das Ziel, das erstellte System auf sein reales Verhalten hin zu simulieren. Die Spezifikation der Informationsverarbeitung beschränkt sich auf die Regelungs- und Steuerungstechnik. Die Softwaretechnik und die Techniken der künstlichen Intelligenz werden nicht unterstützt. Hinzu kommt, dass Aspekte der Kognitionswissenschaft nicht berücksichtigt werden können.

3.3 Wiederverwendung von Lösungswissen

Vor dem Hintergrund der Entwicklung der Informationsverarbeitung fortgeschrittener mechatronischer Systeme und den sich daraus ergebenden Anforderungen, sollte einmal erfolgreich eingesetztes Lösungswissen für die Wiederverwendung aufbereitet werden. Dazu werden zunächst Musteransätze verschiedener Fachdisziplinen und anschließend zwei grundlegende IT-Konzepte für den Umgang mit Lösungswissen diskutiert.

3.3.1 Muster zur Beschreibung wiederkehrender Lösungen

3.3.1.1 Wirkprinzipien der Konstruktionslehre

Nach PAHL/BEITZ werden Wirkprinzipien zur Konstruktion technischer Systeme in der Phase des Konzipierens eingesetzt, die sie wie folgt definieren:

> *„Konzipieren ist der Teil des Konstruierens, der nach Klären der Aufgabenstellung durch Abstrahieren auf die wesentlichen Probleme, Aufstellen von Funktionsstrukturen und durch Suche nach geeigneten Wirkprinzipien und deren Kombination in einer Wirkstruktur die prinzipielle Lösung festlegt.“ [PBF+07, S. 195].*

Das Konzipieren nach PAHL/BEITZ entspricht somit weitestgehend dem Handlungsfeld des Systementwurfs wie er im Fokus dieser Arbeit steht (vgl. Kap. 2.4.3). KOLLER nennt als Basis eines Wirkprinzips physikalische, chemische oder biologische Effekte [Kol98, S. 121]. Nach PAHL/BEITZ beruht ein Wirkprinzip auf einem physikalischen Effekt:

> *„Nur die Gemeinsamkeiten von physikalischem Effekt sowie geometrischen und stofflichen Merkmalen (Wirkgeometrie, Wirkbewegung und Werkstoff) lässt das Prinzip der Lösung sichtbar werden. Dieser Zusammenhang wird als Wirkprinzip bezeichnet […]. Das Wirkprinzip*

stellt den Lösungsgedanken für eine Funktion auf erster konkreter Stufe dar“ [PBF+07, S. 54].

Daraus folgt, dass Wirkprinzipien in Abhängigkeit der zu erfüllenden Funktion und dem zugrundeliegenden Effekt sowie der Wirkgeometrie, der Wirkbewegung und dem Werkstoff beschrieben werden (Bild 3-11).

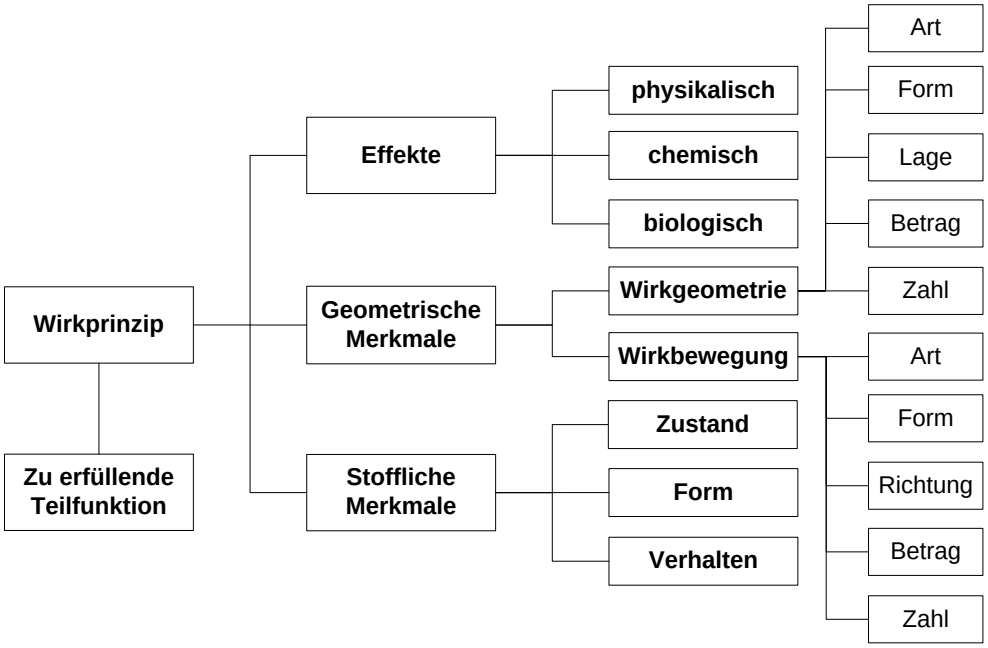


Bild 3-11: Strukturierung von Wirkprinzipien anhand ordnender Gesichtspunkte in Anlehnung an [PBF+07, S. 148f.]

Um eine systematische Suche und Anwendung von Wirkprinzipien zu ermöglichen, werden diese tabellarisch in Katalogen dokumentiert, die sich hinsichtlich ihres Inhaltes, der Darstellungsform und des angebotenen Konkretisierungsgrades unterscheiden können (vgl. Bild A-1 und Bild A-2). So existieren auch Kataloge und Handbücher für physikalische Effekte oder bereits konkrete Lösungselemente (wie z.B. Kauf- oder Normteile) [PBF+07], [Rot00], [VDI2727].

Bewertung: Wirkprinzipien sind in der Konstruktionslehre des Maschinenbaus fest etabliert. Sie haben einen direkten Bezug zu einer zu erfüllenden Funktion und einem zugrundeliegendem Effekt. Ein wesentlicher Punkt ist, dass im Vorfeld eine funktionale Beschreibung des zu entwickelnden Systems erfolgt. Eine Übertragung auf weitere Fachdisziplinen, die an der Entwicklung fortgeschrittener mechatronischer Systeme beteiligt sind, existiert jedoch nicht. Die Spezifikation erfolgt in tabellarischer Form mit Skizze und textueller Beschreibung. Es gibt keinen Ansatz, Wirkprinzipien mit einer Modellierungssprache zu beschreiben.

3.3.1.2 Muster in der Softwaretechnik

Es existieren verschiedene Ansätze, Muster zur Softwareentwicklung einzusetzen. Der bekannteste sind die **Entwurfsmuster** (engl. design patterns) der „Gang of Four“[41]. Es handelt sich dabei um Lösungsschablonen für sich wiederholende Entwurfsprobleme in der Softwareentwicklung. Der Aufbau der Muster basiert dabei auf dem von ALEXANDER vorgestellten Konzept. Die eher generische Beschreibung von Mustern wurde dabei an die Anforderungen der Softwareentwicklung angepasst. Die ausgearbeiteten Muster enthalten dabei bis auf Beispielcode keine fertig codierten Lösungen, sie beschreiben lediglich den Lösungsweg, also wie Softwarecode idealtypisch zu entwerfen ist:

> *„Die Entwurfsmuster […] sind Beschreibungen zusammenarbeitender Objekte und Klassen, die maßgeschneidert sind, um ein allgemeines Entwurfsproblem in einem bestimmten Kontext zu lösen.“*
> *[GHJ+04, S. 4].*

Die Muster lassen sich bestimmten Klassen zuordnen. Hierbei wird zwischen erzeugenden Mustern, strukturellen Mustern sowie Verhaltensmustern unterschieden. Ein Beispiel eines Verhaltensmusters ist der *Beobachter*. Dieses Muster beschreibt z.B. einen Mechanismus, der es erlaubt, alle von einem Objekt abhängigen Objekte zu informieren, wenn sich sein Zustand geändert hat. Analog zur UML-Notation wird ein Zustand durch die Attribute festgelegt (vgl. Bild 3-7). Als Beispiel für ein strukturelles Muster ist das *Kompositum*. Mit diesem Muster werden baumartige Aggregationen hergestellt, die sowohl als einzelne Objekte als auch als Zusammensetzung von Objekten in gleicher Weise genutzt werden können [GHJ+04].

Um die Anwendung von Mustern im Rahmen der Softwareentwicklung sicherzustellen, ist ein einheitliches Format notwendig. Die GoF gliedert die Entwurfsmuster dabei in untergeordnete Abschnitte. Sie verfolgt damit das Ziel, die Struktur so zu vereinheitlichen, dass die Muster miteinander vergleichbar sind. Die Darstellung erfolgt textuell sowie grafisch in UML-Notation und umfasst folgende Aspekte [GHJ+04, S. 8ff.]:

- **Mustername und Klassifizierung:** Der Mustername ist so zu wählen, dass er präzise den wesentlichen Inhalt des Musters vermittelt.
- **Zweck:** Der Zweckabschnitt besteht aus einer kurzen Darstellung, welche Ziele das Muster verfolgt.
- **Auch bekannt als:** Dieser Abschnitt beinhaltet weitere analoge Namen für das Muster, sofern sie existieren.

---

[41] E. GAMMA, R. HELM, R. JOHNSON und J. VLISSIDES verhalfen im Jahre 1994 mit dem Buch „Design Patterns – Elements of Reusable Object-Oriented Software“ den Entwurfsmustern zum Durchbruch in der Softwareentwicklung [GHJ+94]. Sie sind seitdem unter dem Spitznamen „Gang of Four“ (GoF) bekannt.

- **Motivation:** Dieser Abschnitt besteht aus einem Szenario, das darstellt, wie die Klassen- und Objektstrukturen des Musters das Problem lösen.
- **Anwendbarkeit:** Der Anwendbarkeitsabschnitt beschreibt, in welchen Situationen das Entwurfsmuster eingesetzt werden kann.
- **Struktur:** Diese Kategorie ist zweigeteilt. Strukturdiagramme sind eine grafische Repräsentation der Klassen im Muster. Interaktionsdiagramme hingegen veranschaulichen die Abfolge von Operationsaufrufen zwischen Objekten.
- **Teilnehmer:** Der Teilnehmerabschnitt beinhaltet alle beteiligten Klassen und Objekte sowie ihre Zuständigkeiten.
- **Interaktionen:** Der Interaktionsabschnitt beschreibt, wie die Teilnehmer zur Erfüllung der gemeinsamen Aufgabe zusammenarbeiten.
- **Konsequenzen:** Dieser Abschnitt diskutiert, welche Vor- und Nachteile sich durch die Anwendung des Musters ergeben. Zusätzlich wird dargestellt, wie Aspekte der Systemstruktur unabhängig voneinander variiert werden können.
- **Implementierung:** Der Implementierungsabschnitt präsentiert sprachspezifische Aspekte und Implementierungsmöglichkeiten.
- **Beispielcode:** Der Beispielcode diskutiert Codefragmente, die veranschaulichen sollen, wie das Muster implementiert werden kann.
- **Bekannte Verwendung:** Dieser Teil enthält Musteranwendungen in echten Systemen.
- **Verwandte Muster:** Dieser letzte Abschnitt einer Musterbeschreibung setzt das Muster in Bezug zu anderen Entwurfsmustern, diskutiert die relevanten Unterschiede und erläutert, mit welchen Mustern das Muster zusammen verwendet werden kann.

Ein weiterer Ansatz sind die im Rahmen des SFB 614 entwickelten **Echtzeitkoordinationsmuster**. Sie definieren die Kommunikation und Vernetzung mehrerer mechatronischer Systeme auf abstrakter Ebene. Sie werden mit der MechatronicUML-Notation beschrieben, deren zugrunde liegendes Architekturkonzept von der OCM-Struktur abgeleitet ist (vgl. Kap. 3.2.2.4). Die MechatronicUML ist eine Sprache u.a. zur Modellierung des Echtzeitverhaltens des reflektorischen Operators und zur formalen Verifikation sicherheitsrelevanter Aspekte. Sie nutzt im Wesentlichen formalisierte Komponenten- und Zustandsdiagramme der UML (vgl. Kap. 3.2.3.1). Einzelne Komponenten besitzen fest definierte Schnittstellen durch die weitere Komponenten angebunden sind. Sog. *Real-Time Statecharts* spezifizieren das (zeitliche) Verhalten dieser Schnittstellen (sog. Ports), die nicht nur diskrete, sondern (quasi-)kontinuierliche Signale senden. Sie sind so aufgebaut, dass sie im Sinne der Mustertheorie im Rahmen der Softwareentwicklung auf verschiedene Komponenten übertragen werden können. Bild 3-12 zeigt das Echt-

zeitkoordinationsmuster „Distancecoordination“ zur Beschreibung einer Konvoibildung mehrerer intelligenter Schienenfahrzeuge, sog. RailCabs (vgl. Kap.2.2.3) [GHH+08], [ADG+09], [SEH+10].

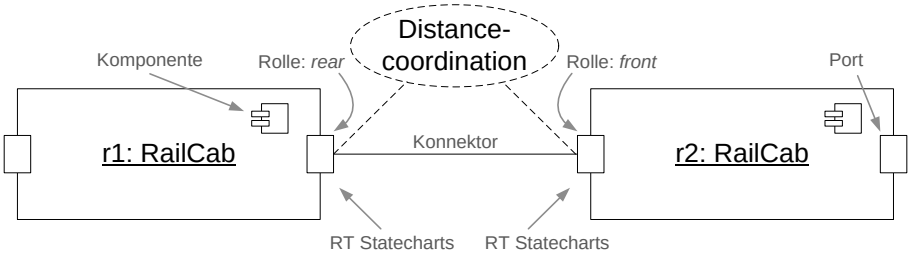


Bild 3-12: Struktur des Echtzeitkoordinationsmusters „Distancecoordination“ in Anlehnung an [GHH+08]

Die Struktur eines Echzeitkoordinationsmuster wird mit dem Komponentendiagramm der MechatronicUML spezifiziert. Sie umfasst drei Bestandteile: die beiden Teilsysteme (Rolle r1 und r2), die Kommunikationsverbindung (Konnektor) und Sicherheitseigenschaften. Ein Echtzeitkoordinationsmuster umfasst neben dem Namen und der Struktur noch die Aspekte Zweck/Ziele, Bedingungen/Voraussetzungen, Verhalten und Verifikation [May08]. Das Verhalten und die Verifikation der Rollen sowie des Konnektors werden durch Real-Time Statecharts beschrieben. Erfolgt eine Einbettung kontinuierlichen Systemverhaltens, z.B. durch Regler, werden diese zu hybriden Rekonfigurationscharts erweitert. So können hierarchische Rekonfigurationen von Regelstrukturen zur Laufzeit modelliert werden, indem Komponenten mit Synchronisationsstatecharts ausgetauscht werden. Wechselt ein RailCab von dem Zustand *no convoy* in den Zustand *convoy rear*, wird zusätzlich ein Abstandsregler vor den Geschwindigkeitsregler geschaltet (Bild 3-13) [GHH+08], [SEH+10].

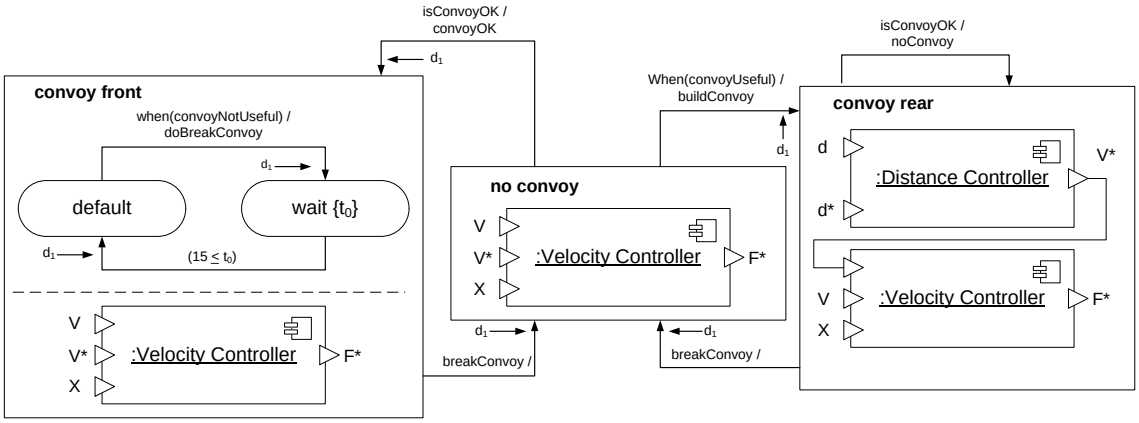


Bild 3-13: Hybrider Rekonfigurationschart für das Echtzeitkoordinationsmuster „Abstandskoordination“ nach [GHH+08]

Bewertung: Die Entwurfsmuster der GoF erfüllen die Definition nach ALEXANDER und sind insbesondere für den objektorientierten Softwareentwurf geeignet. Sie beschreiben in erster Linie, wie sich Software idealtypisch entwerfen lässt und sind somit für die

Dokumentation von Entwurfsobjekten der kognitiven Informationsverarbeitung fortgeschrittener mechatronischer Systeme ungeeignet. Die Echtzeitkoordinationsmuster beziehen sich direkt auf das OCM-Konzept und berücksichtigen folglich eine kognitive Struktur für die Informationsverarbeitung. Sie liefern diesbezüglich erste Muster, wie Umschaltvorgänge des Controllers gesteuert werden können. Sie werden allerdings nicht zum Systementwurf eingesetzt werden, sondern finden im domänenspezifisch Softwareentwurf ihren Einsatz. Ferner adressieren sie vorrangig Echtzeit- und Sicherheitsaspekte.

3.3.1.3 Musteransätze aus der Regelungstechnik

Für die Domäne Regelungstechnik existieren in der Literatur nur Ansätze für Lösungsmuster entsprechend der Musterdefinition nach ALEXANDER. Eine etablierte Herangehensweise in der Regelungstechnik, erfolgreich eingesetztes Wissen zu abstrahieren und auf neue Problemstellungen zu übertragen, liefert bspw. **FÖLLINGER**. Um das Vorgehen und sämtliche Zusammenhänge im Rahmen der Entwicklung grafisch zu modellieren, werden sog. Blockschaltbilder eingesetzt. Es ist die Basis für die weitere Ausarbeitung einer Regelung bzw. eines Regelkreises (Bild 3-14).

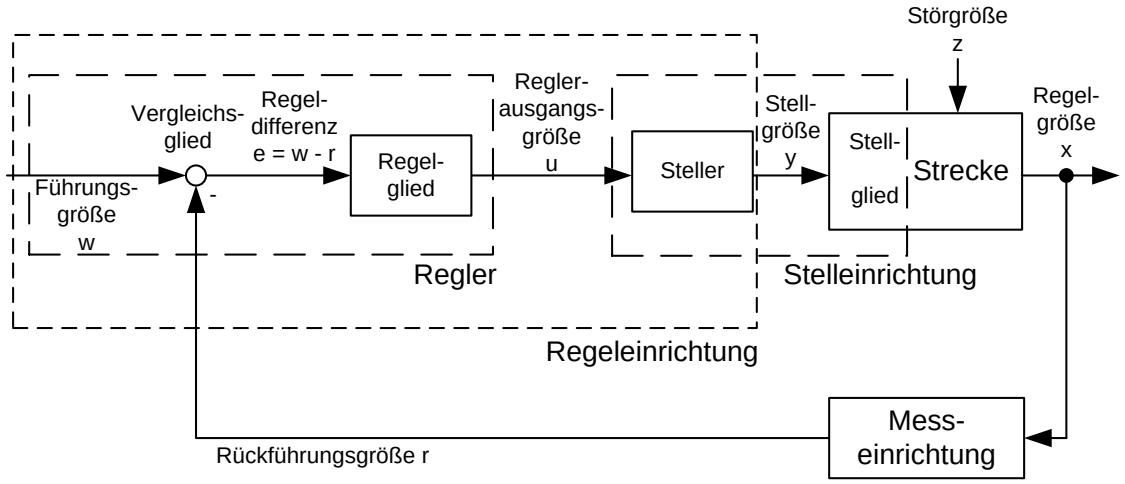


Bild 3-14: Abstraktes Blockschaltbild einer Regelung [Föl08, S. 3]

Kästchen stellen allgemeine Funktionselemente und die durchgezogenen Linien die Funktionsgrößen dar. Diese verbinden die Funktionselemente. Somit werden die wesentlichen Komponenten, wie Regelglied oder Stellglied, als auch sämtliche Bezeichnungen der relevanten Ein- und Ausgangsgrößen abgebildet. Die Ausgangsgröße der sog. Strecke, bzw. des dynamischen Systems, ist die Regelgröße x. Diese wird fortlaufend von Sensoren erfasst und an die Messeinrichtung weitergeleitet. Analoge Messgrößen wie Temperatur oder Druck werden dabei in elektrische Größen umgewandelt, die als Rückführungsgröße r bezeichnet werden. Diese Rückführungsgröße wird mit einer von außen vorgegebenen Führungsgröße w verglichen. Innerhalb des Vergleichsgliedes erfolgt die Ermittlung der Regeldifferenz e. Hauptaufgabe des darauf folgenden Regel-

gliedes ist die Korrektur des dynamischen Verhaltens des Systems. Ohne das Regelglied besteht die Gefahr, dass die Regelung zu instabil, zu ungenau oder zu träge arbeitet. Hat der Vergleich der Führungsgröße w mit der Rückführungsgröße r ergeben, dass eine Differenz vorliegt, muss die Regelgröße an den Sollverlauf der Führungsgröße angeglichen werden. Hervorgerufen wird diese Differenz durch die auf die Strecke einwirkende Störgröße z oder durch eine Vorgabe eines anderen Sollwerts. Die anschließende Anpassung der Werte wird durch die Stelleinrichtung realisiert, die sich in Steller und Stellglied unterteilt [DIN19226], [Föl08].

Bei der Entwicklung einer Regelung geht es daher um Umwandlungen und Anpassungen von skalaren Größen. Diese erfolgen durch die sog. Übertragungsglieder (z.B. das Vergleichsglied), die z.B. durch elektrische Schaltungen realisiert werden. Komplexe anwendungsspezifische Regelungen setzen sich dabei stets aus einer Reihe elementarer Übertragungsglieder im Sinne von wiederverwendbaren Lösungen zusammen. Aus der vorgenommenen Anpassung der Eingangsgröße im Vergleich zur Ausgangsgröße resultieren sog. Übertragungsfunktionen, die durch mathematische Formeln festgehalten sind. Die anschließende Simulation einer entwickelten Regelung erfolgt durch Softwaretools und basiert auf mathematischen Modellen. Diese Modelle gehen dabei auf die Kombination der Übertragungsglieder mit den zugehörigen Übertragungsfunktionen zurück [Föl08, S. 46].

GAUSEMEIER et al. greifen diesen Ansatz auf und zeigen eine erste Zusammenstellung für Muster der Regelungstechnik beispielhaft auf (vgl. Bild A-4). Für diese werden eine zu erfüllende Teilfunktion, eine Blockschaltdarstellung und konkrete Lösungselemente definiert. Allerdings werden die Muster selbst nicht weiter spezifiziert [GHK+06].

LOW beschäftigte sich im Rahmen des SFB 614 mit der Spezifikation der Regelungstechnik in der Prinziplösung selbstoptimierender Systeme und der Überführung dieser in die domänenspezifische Konkretisierung auf Basis von Blockschaltbildern. Die Arbeit ist ein weiterer Ansatz zur Ableitung von Lösungsmustern. LOW nutzte die Spezifikationstechnik des SFB 614 und orientierte sich an dem Operator-Controller-Modul. Der Fokus seiner Arbeit liegt auf dem Controller (CO). Hierfür wurden folgende Partialmodelle identifiziert [Low09, S. 60]:

- **Umfeld**: Auf den CO einwirkende Einflüsse unterteilt nach erforderlichen, neutralen und störenden Einflüssen.
- **Anwendungsszenario**: Charakterisierung der Regelaufgabe und grobe Beschreibung der Lösung.
- **Anforderungen**: Eine tabellarische Auflistung der Anforderungen an den CO.
- **Zielsystem**: Auflistung der Ziele, die die Regelung erreichen soll.
- **Funktionen**: Hierarchische Strukturierung der zu erfüllenden Regelfunktionen.

- **Wirkstruktur**: Grundstruktur der Regelung abgebildet durch Systemelemente und Flussbeziehungen.
- **Verhalten**: Darstellung des Regelverhaltens in Aktivitäten und Zustände.

Low definiert für den Controller mehrere Wirkstrukturen hinsichtlich verschiedener Regelungen und Regelstrategien. Ausgearbeitete Modelle für die anderen Partialmodelle fehlen, so dass keine Lösungsmuster direkt ableitbar sind. Auch für das Operator-Controller-Modul stellt er eine triviale Wirkstruktur auf (Bild 3-15).

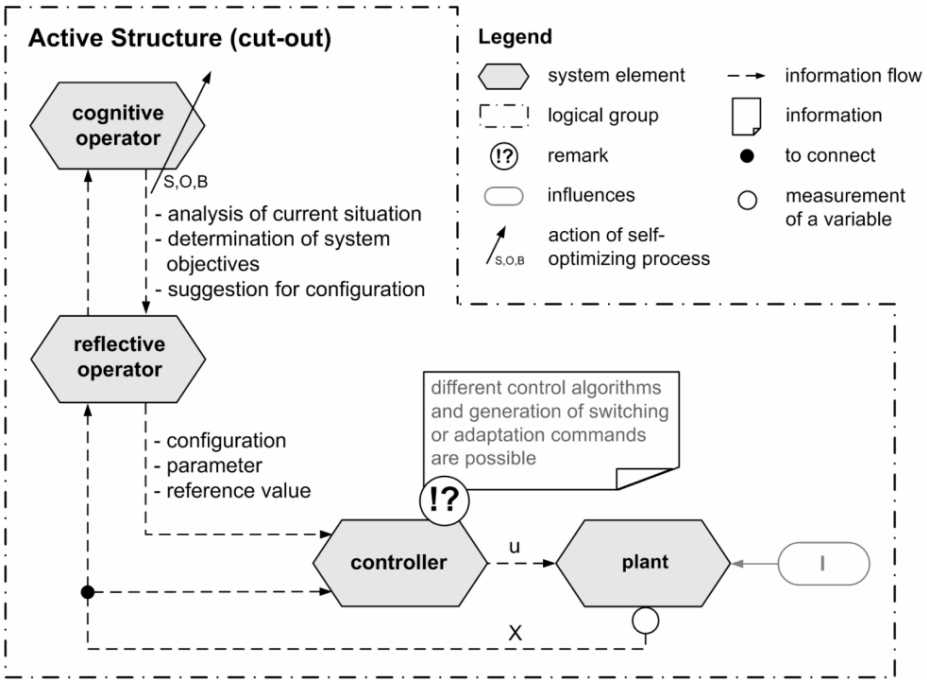


Bild 3-15: Wirkstruktur des Operator-Controller-Moduls [Low09, S. 74]

Bewertung: Allen untersuchten Ansätzen fehlt ein klares Spezifikationsschema für Lösungsmuster der Regelungstechnik. FÖLLINGER liefert mit der Darstellung einer Regelstrecke als Blockschaltbild und der Definition wesentlicher Übertragungsglieder erste Ansatzpunkte zur Ableitung von Lösungsmustern. Das Verhalten der Glieder spiegelt den Verlauf der Sprungantwort wieder, welche stets auf mathematischen Gesetzmäßigkeiten beruht. Der Ansatz nach GAUSEMEIER et al. ist nur für wenige Beispiele und nicht detailliert ausgeführt. LOW bezieht sich auf die gesamte Informationsverarbeitung, indem er sich an der Struktur des OCM orientiert. Er definiert Partialmodelle zur vollständigen Abbildung einer Regelung. Zur Ableitung von Lösungsmustern ist der Detaillierungsgrad noch unzureichend und es wurden bislang nur einige wenige musterähnliche Wirkstrukturen definiert. Zudem wird lediglich der Hinweis gegeben, dass unterschiedliche Regelalgorithmen sowie Schaltbefehle möglich sind. Erste Angaben, welche Verfahren umzusetzen sind, fehlen. Die triviale Wirkstruktur des OCM muss für eine Spezifikation der kognitiven Informationsverarbeitung noch konkretisiert werden.

3.3.1.4 Wirkmuster zur Selbstoptimierung nach SCHMIDT

Im Rahmen des SFB 614 hat SCHMIDT eine Spezifikation von Wirkmustern zur Selbstoptimierung erstellt, um eine Wiederverwendung von Lösungen in den frühen Phasen des Entwurfs selbstoptimierender Systeme zu ermöglichen (vgl. Kap. 2.2.3). Er erkannte, dass selbstoptimierende Systeme zur Umsetzung des Selbstoptimierungsprozesses Funktionen erfüllen müssen, die über konventionelle Funktionen wie „Kraft übertragen“ oder „Energie leiten“ wesentlich hinausgehen. Nach SCHMIDT existieren folgende vier Funktionen, auf denen das Verhalten eines selbstoptimierenden Systems basiert. Sie werden als Funktionen zur Selbstoptimierung bezeichnet [Sch06b, S. 75]:

- **reflektieren:** aus vergangenem Verhalten lernen
- **exploitieren:** bestehendes Wissen nutzen
- **interagieren:** mit anderen Systemen kooperieren
- **explorieren:** zukünftige System- und Umfeldzustände erkunden

Ausgehend von diesen Funktionen definiert SCHMIDT die vier Basiswirkmuster zur Selbstoptimierung Reflexion, Exploitation, Interaktion und Exploration. Durch eine Verfeinerung oder Kombination können aus diesen weitere Wirkmuster zur Selbstoptimierung abgeleitet werden. Die Beschreibung der Wirkmuster zur Selbstoptimierung erfolgt teilweise mit der Spezifikationstechnik des SFB 614, an die sich auch das Spezifikationsschema orientiert (Bild 3-16).

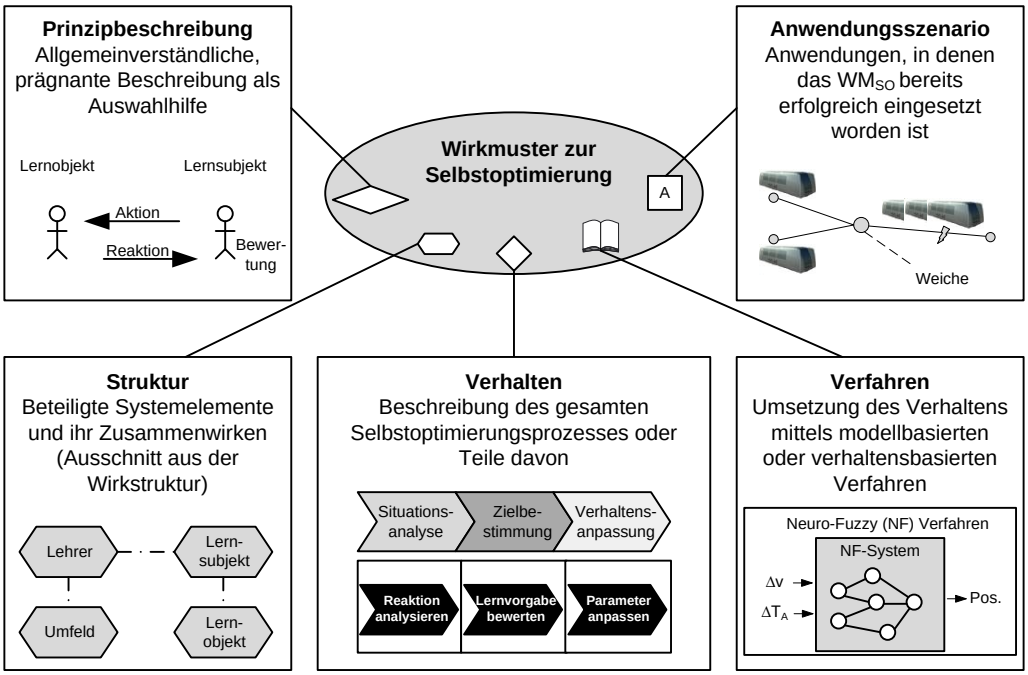


Bild 3-16: Partialmodelle des Spezifikationsschemas für Wirkmuster zur Selbstoptimierung [Sch06b, S. 78]

Das Partialmodell **Anwendungsszenario** kann aus einem oder mehreren Anwendungsszenarien bestehen. Diese beschreiben, wie das Wirkmuster eingesetzt werden kann und sind somit ein wichtiges Kriterium für dessen Auswahl. Ein Anwendungsszenario wird durch eine Skizze und eine textuelle Beschreibung spezifiziert. Zudem werden die im Hinblick auf die Selbstoptimierung wesentlichen Punkte prosaisch festgehalten.

Einen intuitiven Zugang zu dem Wirkmuster soll die **Prinzipbeschreibung** ermöglichen. Auch diese besteht aus einer grafischen sowie textuellen Darstellung des Prinzips und einer Komponente Selbstoptimierung. Letztere fasst die Teilnehmer und deren Beteiligung am Selbstoptimierungsprozess zusammen.

In der **Struktur** werden mittels einer Strukturskizze die beteiligten Systemelemente und deren Beziehung zueinander dargestellt. Sie soll einen Ausschnitt der Wirkstruktur des zu entwickelnden Systems abbilden. Die einzelnen Systemelemente werden den drei Phasen des Selbstoptimierungsprozesses zugeordnet.

Das **Verhalten** beschreibt den Selbstoptimierungsprozess oder Teile davon. Die entsprechende Verhaltensskizze kann mit Aktivitäten der beteiligten Systemelemente spezifiziert werden, die wiederum dem Selbstoptimierungsprozess zugeordnet werden. Grundsätzlich können aber auch andere Techniken wie Petri-Netze oder Kollaborationsdiagramme eingesetzt werden.

Im Partialmodell **Verfahren** werden Methoden, Algorithmen und Erfahrungswissen zur Umsetzung eines Teils oder des gesamten Selbstoptimierungsprozesses aufgelistet. Entsprechend einem Anwendungsszenario werden geeignete Verfahren in einer Verfahrensskizze beschrieben. Eine Konkretisierung erfolgt in den späteren Entwurfsphasen.

SCHMIDT hat auch ein **Wissensmanagementsystem** für die Wirkmuster entworfen. Es besteht aus einer Verwaltungskomponente, einer Zugriffskomponente und einer Wissensbasis. Bei der Erfassung der Wirkmuster im System werden erste Kriterien annotiert, um eine spätere Suche zu ermöglichen.

Bewertung: Das Spezifikationsschema der Wirkmuster ist intuitiv verständlich. Ferner wird mit der Spezifikationstechnik des SFB 614 eine konkrete Modellierungssprache eingesetzt. Allerdings wird diese nicht immer konsequent in den einzelnen Partialmodellen genutzt und in Teilen abgewandelt. Das resultiert in einer rein logischen und mitunter prosaischen und wenig ingenieurwissenschaftlichen Beschreibung. Zwar identifiziert SCHMIDT basale Funktionen zur Selbstoptimierung, eine durchgängige Beschreibung der Informationsverarbeitung ist mit diesen aber nicht möglich. Ein Partialmodell für die Funktionen fehlt gänzlich. Das System zum Management der Wirkmuster ist gut durchdacht, der Aspekt der domänenübergreifenden Zusammenarbeit wird nicht adressiert. Ferner fehlt eine Anbindung an ein Modellierungswerkzeug.

3.3.1.5 Lösungsmuster nach SUHM

SUHM bezeichnet elementare Einheiten zur Entwicklung technischer Systeme als Lösungsmuster. Dieses ist wie folgt definiert:

> *„Ein Lösungsmuster ist eine anwendungsneutrale Beschreibung einer Lösung, die an bestimmte Problemstellungen anpaßbar ist. Lösungsmuster unterstützen durch die Beschreibung einer parametrisierten Lösung im Zusammenhang mit der Lösungsvoraussetzung und der Lösungsumgebung den Problemlösungsprozess innerhalb der Konstruktion unmittelbar. Lösungsmuster repräsentieren damit Konstruktionswissen explizit und deklarativ in elementaren Einheiten.“ [Suh93, S. 6f.].*

Dabei finden diese Lösungsmuster ihren Einsatz im Systementwurf[42]. Der Aufbau eines Lösungsmusters unterteilt sich in die Aspekte Voraussetzungen, Lösung sowie Umgebung. Bild 3-17 zeigt ein Lösungsmuster eines Radiallagers beispielhaft.

Lösungsmuster Radiallager

Radiallagerung — Bezeichnung

DIN 615, DIN 617,... — Quelle

Symbol

Voraussetzungen	Funktionsoperation	(Kraft) leiten
	Funktionsgröße	Radialkraft, Axialkraft
	Wartung	wartungsfrei
Lösung	Relativbewegung	wälzen
	Tragzahl	0,3 … 5000 kN
	Drehzahl	< 10000 min^{-1}
Umgebung	sitzt auf	Welle
	sitzt in	Lagerblock, Gehäuse
	Sicherung innen	keine, Sprengring, Mutter

Bild 3-17: Beispiel eines Lösungsmusters für den Systementwurf nach [Suh93, S 79]

Voraussetzungen: Hier werden die Soll-Eigenschaften eines technischen Objekts dokumentiert. Diese beziehen sich auf Begriffe und Merkmale aus vorangegangenen Entwicklungsphasen, die als Voraussetzungen für das vorliegende Lösungsmuster gelten. In der Regel handelt es sich um Angaben bezgl. Anforderungen und Funktionen.

[42] SUHM bezeichnet den Systementwurf als Prinzipmodellierungsphase, in der ausgehend von der Funktionsstruktur des zu entwickelnden Systems die Prinzipstruktur festgelegt wird. Sie entspricht somit dem Systementwurf fortgeschrittener mechatronischer Systeme (vgl. Kap. 2.4.3) [Suh93, S. 76].

Lösung: Dieser Aspekt beschreibt die Ist-Eigenschaften des Lösungsmusters, die die Voraussetzungen erfüllen. Diese liegen in der Regel in parametrisierter Form vor. Es können aber auch komplexe Ausprägungen definiert werden, die sich wiederum aus einer komplexen (Teil-)Lösung zusammensetzen.

Umgebung: Die Lösung muss unter Berücksichtigung der Umgebung die Voraussetzungen erfüllen, in die sie eingesetzt wird. Daher werden in diesem Aspekt weitere Randbedingungen und Anforderungen definiert.

SUHM strukturiert die Lösungsmuster im Sinne einer Komplexitätsbeherrschung. Dazu wird zwischen elementaren und komplexen Lösungsmustern unterschieden. Komplexe Lösungsmuster lassen sich grundsätzlich in elementare Lösungsmuster zergliedern. Aufgrund der damit einhergehenden hohen Granularität, lässt sich jede Konstruktionsaufgabe durch Kombination von elementaren Lösungsmustern lösen. Der Zusammenhang zwischen komplexen und elementaren Lösungsmustern ist anhand eines Lagersitzes auf einer Welle in Bild 3-18 dargestellt [Suh93].

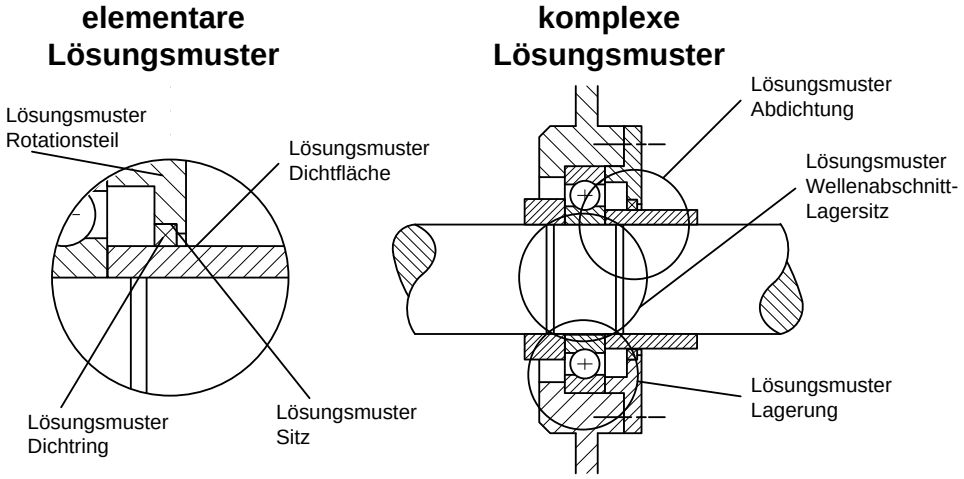


Bild 3-18: Zusammenhang zwischen komplexen und elementaren Lösungsmustern [Suh93, S. 81]

Aufbauend auf den Arbeiten von SUHM verfolgten GRABOWSKI und LOSSACK mit der Universal Design Theory (UDT) das Ziel, Lösungsmuster unterschiedlicher Fachdisziplinen zusammenzuführen. Dafür ist es notwendig, die komplexen Probleme aus den unterschiedlichen Domänen zu abstrahieren. Die abstrahierte Beschreibung soll anschließend eine Übertragung auf weitere Fachbereiche ermöglichen. Durch die Ausarbeitung domänenübergreifender Basiselemente lassen sich Probleme, für die es bisher keine Lösung gab, durch Lösungsansätze anderer Disziplinen lösen [GL00], [Los06].

Bewertung: Die Lösungsmuster nach SUHM bilden die Grundlage für die Dokumentation von technischen Teillösungen. Allerdings eignen sie sich in erster Linie zur Beschreibung rein maschinenbaulicher Problemstellungen. Da die Inhalte jedoch textuell in tabellarischer Form aufgelistet werden, ist eine Übertragung auf weitere Fachdisziplinen denkbar. Die Beschreibungstiefe der Lösung ist hinsichtlich fortgeschrittener me-

chatronischer Systeme zu gering. Weder strukturelle noch verhaltensspezifische Aspekte einer Lösung können abgebildet werden.

3.3.1.6 Engineering Design Pattern nach SALUSTRI

SALUSTRI bezieht sich bei seiner Interpretation des Begriffes Muster auf die Definition von ALEXANDER. Er versteht unter einem Muster eine Art Sprache, mit der eine kontextgebundene Lösung eines Problems, bzw. einer Klasse von Problemen, beschrieben werden kann. Er unterscheidet vier integrale Bestandteile einer Musterbeschreibung, die im Wesentlichen den Kategorien nach ALEXANDER entsprechen (vgl. Kap. 2.4.3) [Sal01], [Sal05]. Im Fokus von SALUSTRIS Arbeit stehen Muster für die Produktentwicklung. Sie sollen helfen, Wissen zu bewahren und wiederabrufbar zu machen. Ferner sollen sie dazu beitragen, erfolgreich eingesetztes Lösungswissen zu verbreiten. Insbesondere im Hinblick auf die zunehmende Beteiligung verschiedener Disziplinen wie der Regelungstechnik, der Mechanik und der Softwaretechnik sei eine einheitliche Spezifikation von großem Vorteil [Sal01, S7f.].

SALUSTRI definiert neun Charakteristika für eine erfolgreiche Akzeptanz und Anwendung von Mustern. Die wesentlichen sind ein einheitlicher Aufbau, eine leichtverständliche sowie domänenübergreifende Darstellung und das Aufzeigen von Lösungsalternativen. Daraus ergibt sich eine basale Struktur für Engineering Design Pattern, die als *therefore-but* Musterstruktur bezeichnet wird und wie folgt definiert ist [Sal05]:

- **Muster-Name (Descriptive Pattern Name):** Diese Kategorie beinhaltet den Namen, den Autoren sowie das Bearbeitungsdatum des Musters.
- **Problem:** Im Rahmen der Problembeschreibung ist darauf zu achten, präzise und detailliert den Kern des Problems in einem Satz zu schildern. Darüber hinaus ist auf den Kontext des Problems einzugehen. Dieser Abschnitt des Musters enthält zudem eine Auflistung der wesentlichen Anforderungen an die zu erarbeitende Lösung.
- **Deshalb (Therefore):** In einem Absatz wird zunächst in Form einer kurzen Anleitung die Art des Lösungsweges beschrieben. Es folgt eine detaillierte Unterteilung sowie generische Ausarbeitung der Problemlösung. Zusätzlich ist im Rahmen dieser Kategorie auf andere geeignete Muster, Lösungen oder Methoden hinzuweisen.
- **Aber (But):** In diesem Abschnitt erfolgt eine Beschreibung der Konsequenzen bei der Anwendung des betrachteten Musters. Hierzu ist es notwendig, explizite Anwendungsbeispiele aufzuzeigen. Idealerweise werden drei Beispiele hinterlegt, die jeweils unterschiedliche Konsequenzen beinhalten.
- **Siehe auch (See Also):** Diese Kategorie des Musters ist optional. Hier können andere Muster genannt werden, die für den Anwender interessant sein könnten. Dabei müssen diese Muster nicht zwangsläufig in der gleichen Form dokumentiert sein.

SALUSTRI beschreibt in dieser Form drei Beispielmuster, von denen nur eines im Kontext der Entwicklung technischer Systeme relevant ist (vgl. Anhang A1.2). Von entscheidender Bedeutung ist, dass die Muster nach SALUSTRI eine Art beschreibende Anleitung für den Wissenstransfer zwischen einem Individuum und einer Gruppe sind. Um dieses Wissen bestmöglich verfügbar zu machen, schlägt SALUSTRI die Nutzung des Internets vor und bezieht sich dabei auf das Wiki-Konzept (vgl. Kap. 3.3.2.2) [Sal05].

Bewertung: Der vorgestellte Ansatz ist in weiten Teilen sehr abstrakt aufbereitet. Er kommt nicht über eine generische Sichtweise auf die Wiederverwendung von Lösungen in der Produktentwicklung hinaus. Die Kategorisierung entspricht zwar der Definition nach ALEXANDER, jedoch ist die Beschreibungstiefe gering. Der Ansatz adressiert zwar eine domänenunabhängige Beschreibung, einen konkreten Einsatz vor allem hinsichtlich der Spezifikation der Informationsverarbeitung ist nicht zu erkennen. Ferner bleibt ungeklärt, wie eine einheitliche Form von Lösungsmuster definiert sein muss, um Wissen verschiedener Disziplinen zusammenzuführen und abzurufen. Der Einsatz eines Wiki-Systems für die IT-Unterstützung ist zwar interessant, SALUSTRI bietet aber auch hier kein konkretes Konzept zur Umsetzung.

3.3.1.7 Musteransatz nach DEIGENDESCH

DEIGENDESCH definiert ein Muster im Kontext der Produktentwicklung wie folgt:

> *„Ein Muster ist die Beschreibung der invarianten Merkmale einer Vielzahl von Lösungen zu ähnlichen Problemstellungen in einer definierten Situation. Charakteristisch für ein Muster ist eine feste Struktur inhaltlicher Elemente und deren Verknüpfungen zu über- und untergeordneten Mustern.“ [Dei09, S. 129]*.

Er identifiziert die drei Kernkomponenten Situation, Problem und Lösung, die jedes Muster besitzen muss. Den Aufbau eines Musters für die Produktentwicklung lehnt er ferner an die SPALTEN-Methode[43] nach ALBERS an [ADT09], [Dei09, S. 129f.]:

- **Name (Alias):** Die Namensgebung ist notwendig, um den Kern des Musters eindeutig und treffend zu beschreiben.
- **Situation:** Die Situation definiert den Kontext und somit die Gültigkeit des Musters. Die Beschreibung beinhaltet alle relevanten Informationen der Situationsanalyse und stellt zusätzlich die Verbindung zu übergeordneten Mustern her.

[43] Das Akronym SPALTEN steht für **S**ituationsanalyse, **P**roblemeingrenzung, **A**lternative Lösungen, **L**ösungsauswahl, **T**ragweitenanalyse, **E**ntscheiden und Umsetzen sowie **N**acharbeiten und Lernen. Es beschreibt eine systematische Problemlösung im Kontext der Produktentwicklung [ABM+05].

- **Problem:** Im Rahmen der Problembeschreibung wird das zugrunde liegende Problem feingegliedert dargestellt. Dies erfolgt sowohl textuell als auch mittels unterstützender Grafiken.
- **Lösung:** Diese bezieht sich nicht auf einen bestimmten Anwendungsfall, sondern liegt in abstrakter Form vor. Sie enthält sowohl den Kern zur Lösung des Problems als auch Hinweise zur Musteranwendung. Ferner wird Bezug genommen zwischen der abstrakten Lösung und dem gegebenen Problem. Die Darstellung erfolgt erneut sowohl textuell als auch grafisch.
- **Tragweite:** Hier werden Chancen und Risiken beschrieben, die sich aus der Anwendung des Musters ergeben können.
- **Konsequenz:** Dieser Teil beinhaltet das Ergebnis der Umsetzung.
- **Beispiele:** Im Gegensatz zur abstrakten Lösungsbeschreibung wird an dieser Stelle auf Anwendungsfälle eingegangen. Erfolgreiche Beispiele als auch Umsetzungen, die nicht zum gewünschten Ergebnis geführt haben, werden dargestellt.
- **Verwandte Muster:** Es besteht die Möglichkeit, dass ein Problem durch unterschiedliche Muster gelöst werden kann. Diese sind in dieser Kategorie aufgelistet.
- **Quellen:** Sämtliche verwendeten Quellen sind zu dokumentieren.
- **Metadaten:** Schlagworte, Kategorien sowie ergänzende Informationen, wie Signifikanz, Ersteller, Bearbeiter oder Änderungsdatum werden hier aufgeführt.

DEIGENDESCH hat die Spezifikation im Rahmen des SFB 499[44] auf Lösungen im Bereich der Mikrotechnik angewandt (vgl. Bild A-5). Bild 3-19 zeigt, wie die dokumentierten Muster erzeugt und angewendet werden. Zunächst identifiziert und dokumentiert ein Experte ein Muster mit den drei Kernkomponenten Situation, Problem und Lösung. Das dafür notwendige Wissen besitzt dieser z.B. aus zurückliegenden Produktentstehungsprozessen. Die Musterrepräsentation enthält die Wissensbasis, in der sämtliche Muster gespeichert sind. Über Schnittstellen haben weitere Experten oder Benutzer die Möglichkeit, auf die abgelegten Muster zuzugreifen. Sie können die Muster anschließend sowohl auf ihre Problemstellung anwenden als auch validieren und optimieren und sie anschließend wieder in der Wissensbasis ablegen. Die Musterrepräsentation wurde in das Informationssystem **MyBoK**[45] integriert, so dass dessen Funktionalität, insbesondere zur Suche, genutzt werden kann.

---

[44] Der Sonderforschungsbereich 499 „Mikrourformen“ wird von der Deutschen Forschungsgemeinschaft (DFG) gefördert und ist noch bis 2011 bewilligt.

[45] Das Micro Book of Knowledge (MyBoK) ist ein projektinternes Wiki-System des SFB 499 (vgl. Kap. 3.3.2.2).

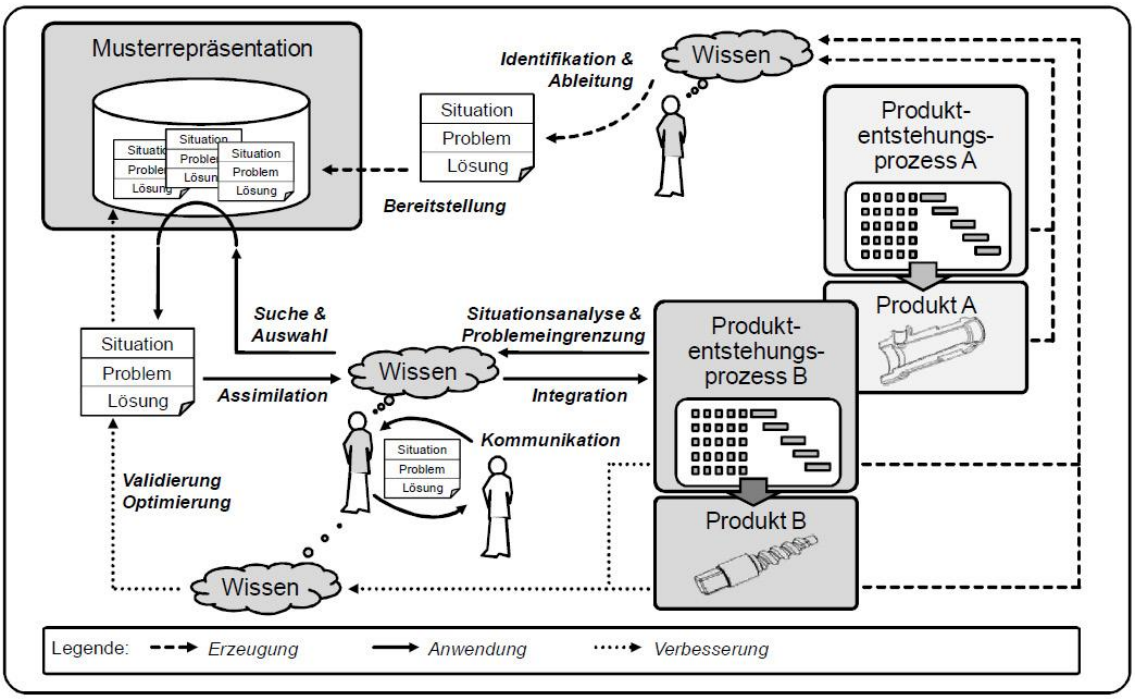


Bild 3-19: Aktivitätenmodell des Musteransatzes [Dei09, S. 133]

Bewertung: Der vorgestellte Musteransatz nutzt eine textuelle Beschreibung zur Darstellung der wesentlichen Inhalte. Die Kernkomponenten und Form entsprechen weitestgehend der Definition nach ALEXANDER. Der Ansatz wurde für die Mikrotechnik umgesetzt, weitere Disziplinen wurden bislang nicht adressiert. Trotz des generischen Ansatzes wird eine domänenübergreifende Spezifikation nicht angestrebt. Der Ansatz wird durch die Orientierung an einer in der Produktentwicklung etablierten Vorgehensweise dem Entwurf technischer Systeme gerecht. Kritisch ist der hohe Grad an prosaischer Beschreibung. Eine Modellierungssprache zur Beschreibung der Muster ist nicht vorgesehen. Konkrete Kriterien zur problemspezifischen Auswahl werden in den Mustern nicht explizit definiert.

3.3.2 IT-Konzepte für den Umgang mit Lösungswissen

3.3.2.1 Expertensysteme

Bei Expertensystemen handelt es sich um wissensbasierte Systeme, die das Wissen von Experten bestimmter Fachgebiete softwaretechnisch repräsentieren und zur Lösungsfindung komplexer Problemstellungen eingesetzt werden. Sie sind dem Forschungsbereich der künstlichen Intelligenz zuzuordnen, den sie mit Beginn ihrer kommerziellen Einführung in den 1980er Jahren stark prägten. Ein wesentlicher Aspekt dieses Konzepts liegt auf der Formalisierung und Dokumentation des Expertenwissens in einer Wissensbasis. Expertenwissen bezeichnet jenes Wissen, das durch langjährige Erfahrung entstand und in der Regel personengebunden ist. Durch eine Externalisierung entsteht die Möglich-

keit, dieses Wissen in geeigneter Form abzubilden und es somit Dritten zur Verfügung zu stellen. Wesentliche Komponente ist eine Problemlösungsstrategie, die das Wissen aus der Wissensbasis für logische Schlussfolgerungen nutzt. Ziel ist die Reproduktion der menschlichen Problemlösungskompetenz. Expertensysteme weisen daher in ihrer Struktur eine klare Trennung zwischen Wissensbasis und Problemlösungskomponente auf (Bild 3-20) [SNC87], [Kel00], [BK06].

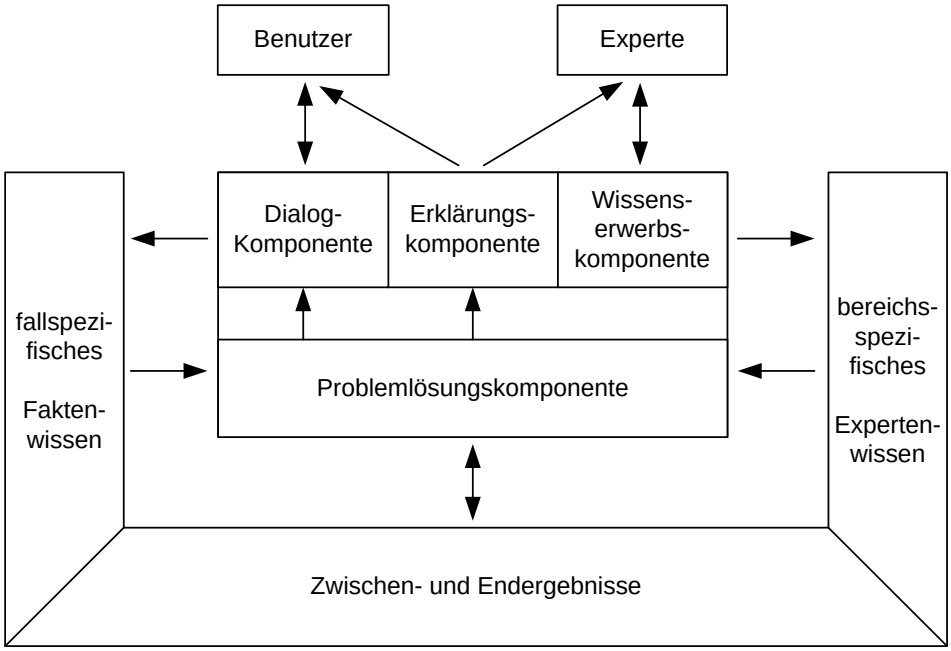


Bild 3-20: Architektur eines Expertensystems [Pup86, S. 2]

Die Architektur eines Expertensystems besteht aus der anwendungsspezifischen Wissensbasis und einem bereichsunabhängigen Steuersystem. Zusätzlich existieren Schnittstellen zu den sog. Experten (Wissenslieferanten) und den Benutzern (Wissensanwendern) Die Wissensbasis setzt sich aus insgesamt drei Kategorien zusammen [Pup86]:

- **Fallspezifisches Faktenwissen** wird durch den Benutzer während einer Anfrage eingegeben.
- **Bereichsspezifisches Expertenwissen** wird von einem Experten hinzugefügt.
- **Zwischen- und Endergebnisse** werden durch das System hergeleitet.

Das Steuersystem setzt sich hingegen aus folgenden vier Komponenten zusammen [Kel00, S. 66f.]:

- Die **Dialogkomponente** managet den Informationsaustausch zwischen dem Benutzer und dem Expertensystem.
- Die **Wissenserwerbskomponente** empfängt und überführt das Wissen in die Wissensbasis. Der Experte kann dieses zu einem späteren Zeitpunkt noch verändern.

- Die **Problemlösungskomponente** sucht nach einer Lösung für das durch den Benutzer eingegebene Problem. Sie kann während des Bearbeitungsvorganges weitere Fakten von dem Benutzer anfordern und hat zudem Zugriff auf die Wissensbasis.
- **Erklärungskomponente:** Mit Hilfe dieser Komponente wird bspw. durch Plausibilitätskontrollen oder Fehleranalysen versucht, dem Benutzer das interne Vorgehen des Expertensystems verständlich zu vermitteln.

Eine wesentliche Schwierigkeit bei der Umsetzung eines Expertensystems ist die Externalisierung des Expertenwissens. Dieses Wissen muss in derartigen Systemen in geeigneter Form gespeichert und abgebildet werden. Zusätzlich ist eine Vorgehensweise zu erarbeiten, wie dieses Wissen auf die jeweiligen Probleme der Benutzer zielführend angewendet werden kann [Kel00, S. 67].

Bewertung: Das Grundkonzept, personengebundenes Wissen zu externalisieren, spiegelt den Grundgedanken der Mustertheorie wider. Auf welche Art und Weise dieses umgesetzt wird, ist je nach Anwendungsfall unterschiedlich. Daher ist dieser Aspekt in Bezug auf die Lösungsmuster neu zu entwickeln. Eine Wissensbasis für die Bewahrung des Lösungswissens muss ein integraler Bestandteil eines IT-Konzepts zum Umgang mit Lösungswissen sein. Inferenzmechanismen zur Generierung neuer Lösungen in Form von Mustern durch das IT-System bieten zwar eine interessante Funktionalität, sind aber keine Anforderung an die Arbeit des Autors. Der Fokus liegt auf dem gezielten Zugriff auf das Lösungswissen, das sich in der Wissensbasis befindet.

3.3.2.2 Wiki-Systeme

1994 entstand das erste im Internet veröffentlichte Wiki-System, das WikiWikiWeb[46]. Es wurde von dem Softwareentwickler CUNNINGHAM entwickelt, um einen schnellen und kooperativen Zugriff auf Entwurfsmuster der Softwaretechnik zu ermöglichen (vgl. Kap. 3.3.1.2). Grundsätzlich ist ein Wiki-System ein vereinfachtes Content-Management-System (Inhaltsverwaltungssystem). Es handelt sich dabei um eine Software für Webseiten, die es dem Benutzer ermöglicht, z.B. textuelle oder grafische Inhalte sowohl zu recherchieren als auch hinzuzufügen. Ziel ist es, das Wissen einzelner Personen zu einem Konsens zusammenzutragen und eine kooperative Zusammenarbeit zu ermöglichen. Wiki-Systeme zeichnen sich durch eine einfache und intuitive Bedienbarkeit aus. Eine der bekanntesten Anwendungen ist die Online-Enzyklopädie Wikipedia [LC01], [EGH+08]. In der Regel sind die im Internet vorhandenen Wikis von jedem Benutzer öffentlich lesbar und veränderbar. Bei der Erstellung eines neuen Wikis besteht jedoch die Möglichkeit, nur bestimmen Benutzergruppen den Zugriff auf den Inhalt zu erlauben.

---

[46] Wiki heißt auf hawaiisch „schnell“. Auf Hawaii fahren sog. WikiWiki-Busse Touristen vom und zum Flughafen. CUNNINGHAM leitete daraus die Bezeichnung WikiWikiWeb ab [LC01].

Neben der Anwendung eines Wikis über das Internet lassen sich derartige Softwaretools auch in lokalen Netzwerken anwenden. Im Vordergrund steht dabei stets die Nutzung als gemeinschaftliches Arbeits- und Präsentationssystem. Eine Besonderheit dieser Software ist die Versionsverwaltung. Diese ermöglicht es, im Falle von Anwenderfehlern, eine frühere Version der fehlerbehafteten Webseiten wiederherzustellen. Eine weitere wesentliche Funktion der meisten Wikis ist die Verknüpfung unterschiedlicher Seiten durch Querverweise (Links). Durch diese Vernetzung hat der Benutzer die Möglichkeit, ohne großen Suchaufwand auf weitere Erklärungen einzelner Begriffe zuzugreifen. Doch noch weitere Funktionen einer sog. Social Software[47] können implementiert werden. Nach MCAFEE können insgesamt sechs Funktionen einer Social Software unterschieden werden [Mca06, S. 23ff.]:

- **Search:** Ein soziales Informationssystem muss eine strukturierte Suche nach dessen Inhalten, bspw. durch eine Volltextsuche, bieten.
- **Links:** Querverweise ermöglichen nicht nur den Zugriff auf ähnliche Inhalte, sondern können je nach Anzahl der Zugriffe die Suche konkretisieren.
- **Authoring:** Der Benutzer soll nicht nur Wissen abrufen, sondern auch als Wissenslieferant (Autor) agieren können.
- **Tags:** Mit kurzen Zusatzinformationen sollen Inhalte gekennzeichnet werden, um so zu einer unkomplizierten Kategorisierung und Auswertung zu gelangen.
- **Extensions:** Auswertealgorithmen sollen benutzerspezifische Profile erstellen, um für ihn weitere interessante und vielleicht unbekannte Inhalte zu identifizieren.
- **Signals:** Um den Benutzer nicht mit neuen Informationen zu erdrücken, sollten neue Inhalte mit einem Kernsatz und einem Link versehen werden.

Bewertung: Im Gegensatz zum Expertensystem liegt bei dem Wiki-Konzept der Schwerpunkt auf der Art und Weise des Zugriffs. Im Fokus steht das kollaborative Arbeiten an bestimmten Inhalten. Durch die Möglichkeit, Zugriffsrechte nur bestimmten Benutzergruppen zu erteilen, wird eine Übertragung auf die Anwendung von Lösungsmustern möglich. Wissenslieferanten könnten die Muster eingeben und ggf. verändern. Wissensanwender hätten anschließend die Möglichkeit, diese zu recherchieren, auf ihre Problemstellungen zu übertragen und wieder zurückzuführen. Die Funktionen einer Social Software umfassen interessante Konzepte für die domänenübergreifende Zusammenarbeit. Die Inhalte bei Wikis sind in der Regel textueller Natur. Eine Anbindung eines Werkzeugs zur Modellierung der Muster existiert nicht.

---

[47] Social Software und das Web 2.0 sind eng miteinander verbunden. Klare Definitionen bzw. Abgrenzungen existieren nicht. Entscheidendes Merkmal ist die hohe Relevanz des Beitrags des Nutzers. Wikis, Blogs, aber auch soziale Netzwerke sind Beispiele einer Social Software [KW08].

3.4 Handlungsbedarf

Ein Vergleich des Stands der Technik mit den Anforderungen aus Kapitel 2.6 führt zu folgender Bewertung, die Bild 3-21 zusammenfasst:

A1) Interdisziplinarität: Der Großteil der Ansätze berücksichtigt nur das Zusammenwirken technischer Disziplinen. Kognitionsrelevante Disziplinen fließen nur in die Strukturkonzepte für eine kognitive Informationsverarbeitung ein. Vorrangig die klassischen kognitiven Architekturen zeigen eine kognitionswissenschaftliche Fundierung.

A2) Systematische Vorgehensweise: Ansätze aus der Konstruktionslehre, wie die Funktionsbeschreibung in der Produktentwicklung, den Wirkprinzipien oder den Lösungsmustern nach SUHM bzw. DEIGENDESCH erfüllen diese Anforderung. Von den Beschreibungssprachen erfüllt die Spezifikationstechnik des SFB 614 die Anforderung.

A3) Orientierung am Systementwurf: Das Framework für kognitive Produkte ermöglicht die Zuordnung von Teillösungen zu kognitiven Funktionen. SysML sowie die Spezifikationstechnik des SFB 614 kommen als Beschreibungssprache in Frage. Etablierte Ansätze aus der Konstruktionslehre werden dieser Anforderung gerecht.

A4) Verständlichkeit: Die psychologieorientierten Architekturen benötigen eine Einarbeitung in die Grundlagen der Motivations- und Emotionspsychologie. Ansätze der Softwaretechnik könnten für den ingenieurgerechten Gebrauch aufbereitet werden.

A5) Ganzheitliche Spezifikation der Informationsverarbeitung: Keine der untersuchten Methoden zur Funktionsbeschreibung erfüllt diese Anforderung in vollem Umfang. Die untersuchten Strukturkonzepte entsprechen zwar teilweise der Strukturierung nach STRUBE, aber keiner der Ansätze beschreibt die kognitive und reaktive Ebene gleichwertig. Als Sprache eignen sich die Spezifikationstechnik des SFB 614 sowie die SysML.

A6) Technische Relevanz: Die Mehrzahl der untersuchten Ansätze können im Kontext der Entwicklung technischer Systeme genutzt werden. Die Beschreibung kognitiver Funktionen adressiert allerdings keiner der Ansätze explizit.

A7) Lösungswissen für den Entwurf der Informationsverarbeitung: Keiner der analysierten Musteransätze erfüllt diese Anforderung. Musteransätze aus der Softwaretechnik spezifizieren entweder den Softwareentwurf oder nur die Kommunikation.

A8) Auswahl und Kombination von Lösungswissen: Nur für die Wirkprinzipien gibt es Konzepte zur geeigneten Auswahl und Kombination zu Lösungsvarianten. Expertensysteme und auch Wiki-Systeme können zumindest zur Auswahl genutzt werden.

A9) Rechnerunterstützung durch eine Wissensbasis: Beide IT-Konzepte könnten zum Aufbau einer Wissensbasis eingesetzt werden. Die Mehrzahl der untersuchten Lösungsmusteransätze wird durch eine Datenbank unterstützt, die aber keine Funktionen für eine disziplinübergreifende Zusammenarbeit zur Verfügung stellen.

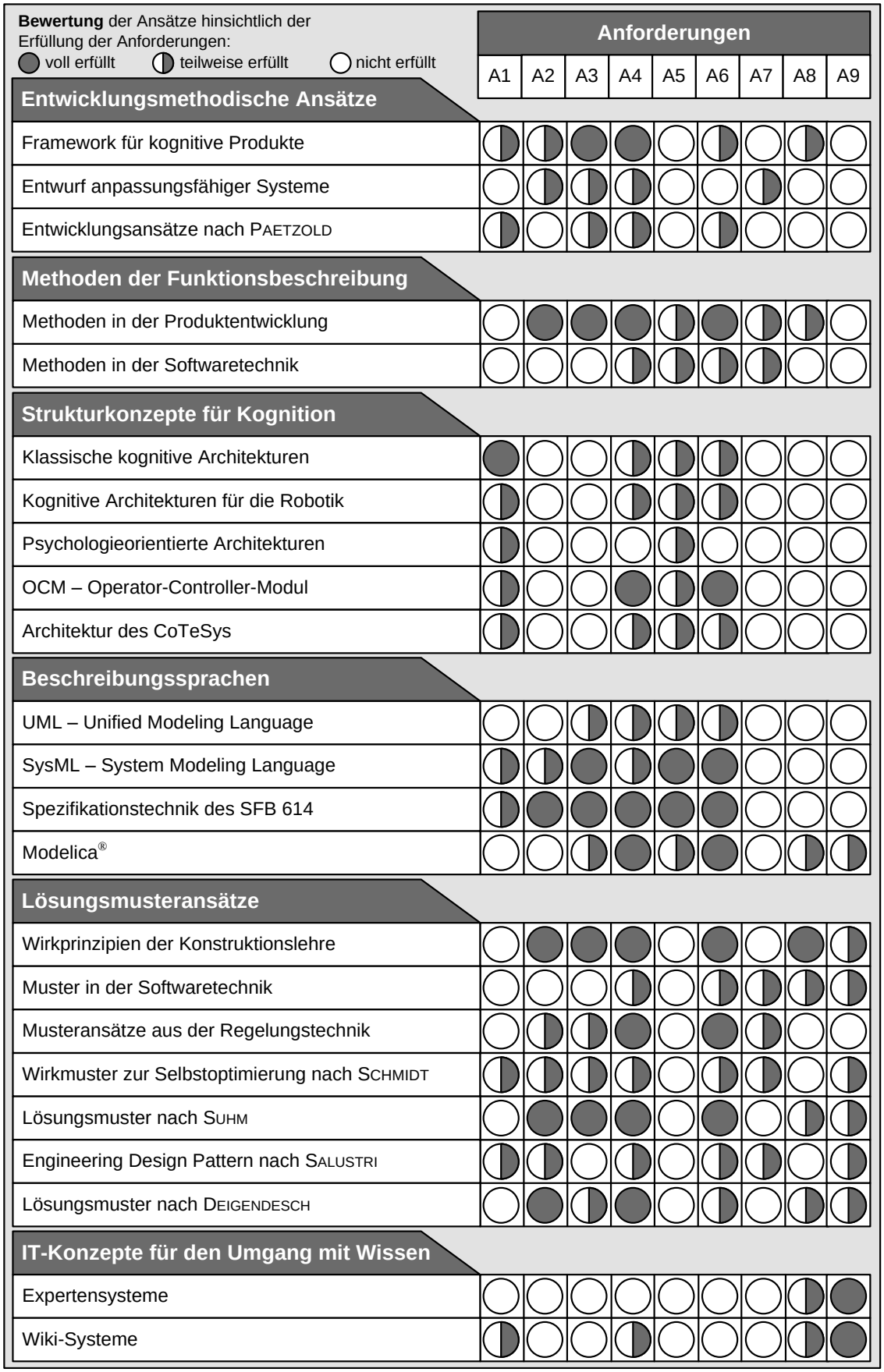


Bild 3-21: Bewertung des untersuchten Stands der Technik anhand der Anforderungen

Keiner der untersuchten Ansätze als auch keine triviale Kombination bestehender Ansätze erfüllt alle Anforderungen in vollem Umfang. Die entscheidende Schwachstelle ist die mangelhafte Verzahnung zwischen der etablierten Konstruktionsmethodik und modernen softwaretechnischen Ansätzen zur Realisierung einer kognitiven Informationsverarbeitung. Zudem fokussieren viele Ansätze nur Teilaspekte des Gesamtproblems. Es besteht dringender Handlungsbedarf für eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*.

4 Entwicklungssystematik zur Integration kognitiver Funktionen

Dieses Kapitel bildet den Kern der vorliegenden Arbeit. Es stellt eine *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* vor. Diese hat den Anspruch, dem aufgezeigten Handlungsbedarf aus der Problemanalyse und dem Stand der Technik gerecht zu werden. Kapitel 4.1 gibt einen Überblick über die Entwicklungssystematik und ihre Bestandteile. Deren Grundlage bildet eine gemeinsame Begriffswelt aus kognitionswissenschaftlicher und technischer Sicht. Diese wird in Kapitel 4.2 erläutert. Danach werden die Bestandteile der Entwicklungssystematik nacheinander beschrieben. Kapitel 4.3 stellt das Vorgehensmodell vor und ordnet dieses in den Entwicklungsprozess fortgeschrittener mechatronischer Systeme ein. Kapitel 4.4 fokussiert die Beschreibung kognitiver Funktionen sowie einer entsprechenden Informationsverarbeitung. In Kapitel 4.5 wird eine lösungsmusterorientierte Technik zur Wiederverwendung von Lösungswissen für den frühzeitigen Entwurf fortgeschrittener mechatronischer Systeme beschrieben. Abschließend präsentiert Kapitel 4.6 ein Konzept für den rechnergestützten Umgang mit diesen Lösungsmustern sowie eine prototypische Implementierung eines IT-Werkzeugs auf Basis des erarbeiteten Konzepts.

In diesem Kapitel wird die erarbeitete Entwicklungssystematik aus methodischer Sicht beschrieben. Vereinzelt werden einige Ansätze anhand der Demonstratoren des SFB 614 beispielhaft veranschaulicht (vgl. Kap. 2.2.3). Die durchgängige Anwendung der Entwicklungssystematik erfolgt im Anschluss in Kapitel 5.

4.1 Die Entwicklungssystematik im Überblick

Die *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* umfasst vier elementare Bestandteile (Bild 4-1):

- ein strukturiertes **Vorgehensmodell**, das die durchzuführenden Tätigkeiten definiert und deren Ergebnisse sowie notwendige Hilfsmittel, wie z.B. Methoden oder Werkzeuge, integriert,
- eine Technik zur **Systembeschreibung**, die eine dedizierte Sprache und abstrakte Entwurfsschablonen für eine intuitive und disziplinübergreifende Spezifikation der zu entwickelnden Informationsverarbeitung bereitstellt,
- wiederverwendbares **Lösungswissen** auf Basis eines Musteransatzes um relevante Aspekte einer Aufgabe und ihrer Lösung auch für Dritte bereitzustellen und
- ein Konzept zur **Werkzeugunterstützung** im Umgang mit dem Lösungswissen, das im Wesentlichen aus einer Wissensbasis sowie einem Zugriffsystem besteht und prototypisch implementiert ist.

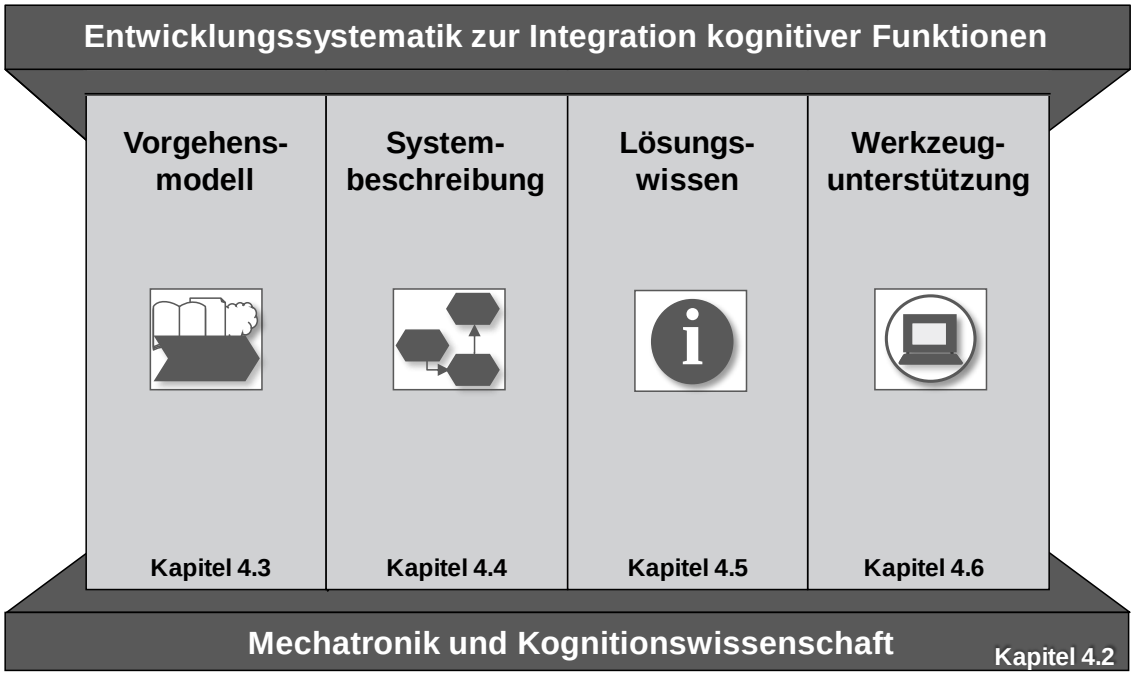


Bild 4-1: Aufbau der Entwicklungssystematik

Grundlegende Aspekte der Entwicklungssystematik sind die Klärung wichtiger kognitionswissenschaftlicher Begriffe im Kontext der Mechatronik sowie die Frage nach einer geeigneten Architektur für die kognitive Informationsverarbeitung und nach einer aussagestarken Beschreibungssprache. Angesichts der Analyse des Stands der Technik und der Bedeutung der Selbstoptimierung für fortgeschrittene mechatronische Systeme werden zu diesem Zweck das Operator-Controller-Modul (OCM) und die Spezifikationssprache des SFB 614 verwendet. Beide Ansätze werden hinsichtlich der Integration kognitiver Funktionen weiterentwickelt. Ziel ist die frühzeitige Spezifikation des OCM als abstrakte und nicht anwendungsspezifische Architektur der Informationsverarbeitung im Rahmen des Systementwurfs. Besonders die Prinziplösung eines selbstoptimierenden Systems im Sinne des SFB 614 als konkretes fortgeschrittenes mechatronisches System steht im Fokus der Entwicklungssystematik.

4.2 Begriffswelt der Mechatronik und Kognitionswissenschaft

Die Problemanalyse hat gezeigt, dass fortgeschrittene mechatronische Systeme ein Verhalten erreichen sollen, das bisher nur intelligenten Lebewesen zugeschrieben wurde. Derartige Systeme besitzen nicht nur ein reaktives und adaptives Verhalten, sondern können dieses auf Basis ihrer Ziele ändern. Die Ziele und insbesondere ihre Ausprägungen werden dabei nur teilweise durch den Entwickler vorausgedacht. Vielmehr werden diese zur Laufzeit erst bestimmt, um ein hinsichtlich der aktuellen Situation optimales Verhalten zu erzielen. Derartiges autonomes Verhalten geht einher mit einer Reihe von sog. Self-X-Eigenschaften, von denen die Selbstoptimierung das zentrale Wirkparadigma im Bezug auf die Mechatronik ist. Ein derart intelligentes Verhalten erfordert eine

technische Reaktivierung kognitiver Funktionen, wie sie die menschliche Informationsverarbeitung umsetzt. Intelligenz ergibt sich aus einer geeigneten Allokation dieser Funktionen zur Verhaltenssteuerung.

Die Entwicklung derartiger intelligenter mechatronischer Systeme bedingt eine stärkere Zusammenführung der Kognitionswissenschaft, die die Grundlagen kognitiver Systeme erforscht, und der Mechatronik, die die benötigten technischen Disziplinen vereint. Diese Verzahnung bildet die Basis für die zu erarbeitende Entwicklungssystematik und ihre Bestandteile (vgl. Bild 4-1). Gerade ein einheitliches Verständnis kognitionswissenschaftlicher Aspekte und deren Übertragung auf das mechatronische System sind unumgänglich.

In der Kognitionsforschung werden Begriffe wie „kognitive Funktionen“, „kognitiver Prozess“ oder „kognitive Leistung“ völlig uneinheitlich und in Teilen sogar synonym verwendet [Mül98, S. 7ff.]. Für eine methodische Vorgehensweise in der Entwicklung müssen aber die wesentlichen Begriffe eindeutig definiert werden – nur so kann unter allen Beteiligten ein einheitliches Verständnis des zu entwickelnden Systems erlangt werden. Hinzu kommt der unklare Informationsbegriff, der einer Konkretisierung hinsichtlich der Beschreibung derartiger Systeme bedarf. In diesem Zusammenhang sind die Begriffe „Daten“, „Informationen“ und „Wissen“ zu unterscheiden, die sich zumindest durch den Grad der Formalität und den Grad der Interpretation durch das verarbeitende System differenzieren lassen [Web10, S. 6]. So hält NORTH diese Begriffe in der sog. Wissenstreppe fest, in der auf der ersten Stufe aus Zeichen plus einer formalisierten und ordnenden Form **Daten** werden (vgl. Bild 4-2). Die Verarbeitung und Interpretation von Daten führt auf der zweiten Stufe zu **Informationen**, die sich insbesondere durch den Bedeutungsgewinn durch das verarbeitende System von Daten unterscheiden [ISO9000, S. 25]. In der Informationstechnik ist es daher nicht unüblich Daten als „unverarbeitete Informationen“ zu bezeichnen [Hey04-ol, S. 5]. Werden Informationen von dem System in Bezug zueinander gesetzt, entsteht auf der dritten Stufe **Wissen**. Bei der Wissensgewinnung spielt das bereits vorhandene Wissen eine entscheidende Rolle für die Vernetzung der Informationen.

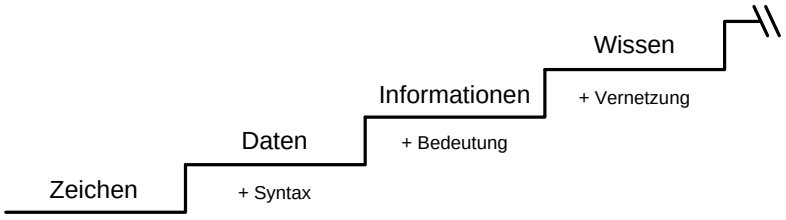


Bild 4-2: Gekürzte Wissenstreppe[48] nach [Nor05, S. 39]

[48] NORTH bezieht seine Wissenstreppe auf die wissensorientierte Unternehmensführung, so schließen sich noch die Stufen „Können“ „Handeln“, „Kompetenz“ und „Wettbewerbsfähigkeit“ an [Nor05, S. 39]. Ein ähnlicher Ansatz, der den Verlauf von Daten hin zu Wissen beschreibt, ist die sog. DIKW-Hierarchie (Data, Information, Knowledge, Wisdom) [Hey04-ol, S. 3].

Die Übertragung von Informationen und somit auch von Zeichen und Daten wird durch (physikalische) **Signale** ausgeführt. Während die Signalübertragung in technischen Systemen z.B. elektrisch, magnetisch oder optisch ablaufen kann, beruht diese innerhalb biologischer Systeme auf chemisch-elektrischen Vorgängen in und zwischen den Nervenzellen [Czi08, S. 13], [BE01, S. 37f.].

Vor diesem Hintergrund und der kognitionswissenschaftlichen Sicht, die Kognition als jede Art der Informationsverarbeitung durch eine zentrale Verarbeitungseinheit im System definiert (vgl. Kap. 2.3.1), können die zentralen kognitionsorientierten Begriffe in einem, für die Entwicklung mechatronischer Systeme relevanten Kontext gesetzt werden. Zunächst gilt, dass Kognition gleich **kognitive Informationsverarbeitung** ist. Sie überführt Informationen in systeminternes Wissen und setzt dieses wiederum zur Verhaltenssteuerung ein. Dieses Wissen kann aber nur entstehen, wenn zusätzlich weitere, einfachere Formen der Informationsverarbeitung vorhanden sind. Es müssen zunächst Zeichen detektiert, Daten daraus aufgebaut und aus diesen Informationen abgeleitet werden. Nur so kann ein System Wissen über sich und sein Umfeld schrittweise erlangen. Ferner ist die kognitive Informationsverarbeitung zeitintensiv, so dass ein Großteil der Informationsverarbeitung reaktiv und der Kognition untergelagert ablaufen muss.

Kognitive Funktionen sind grundsätzlich informationsverarbeitende Funktionen. Dabei werden aber nicht nur neue Informationen formalisiert, sondern sie sind die Basis dafür, dass die Informationen in Bezug zueinander gestellt und mit bestehenden Informationen verglichen werden. Dafür können in Anlehnung an die Funktionen des menschlichen Nervensystems drei übergeordnete kognitionsrelevante Funktionen der Informationsverarbeitung unterschieden werden [BE01, S. 34f.]:

- Die **Input-Funktion** sammelt die Informationen aus dem Systemumfeld und leitet diese an einen definierten Speicherort.
- Die **Verarbeitungsfunktion** analysiert Informationen und leitet eine entsprechende Entscheidung ab. Sie umfasst die Mehrzahl an kognitiven Funktionen wie das Denken, Erinnern oder Entscheiden.
- Die **Output-Funktion** leitet die Informationen bzgl. der getroffenen Entscheidungen an die jeweiligen Systemelemente, die diese in entsprechende Aktionen umsetzen können.

In der Entwicklung mechatronischer Systeme werden Funktionen prinzipiell als Substantiv-Verb-Kombination spezifiziert. Hier wird ersichtlich, dass das Funktionsverb zwar die zentrale Aufgabe einer Funktion beschreibt, aber erst die Wahl des Funktionssubstantivs eine kognitive Funktion ausreichend definiert – bspw. kann das Funktionsverb „erfassen“ eine nicht kognitive Funktion „Messdaten erfassen“ oder eine kognitive Funktion „Zusammenhänge erfassen“ bilden.

Kognitive Prozesse hingegen beschreiben den Ablauf einer Informationsverarbeitung, an der kognitive Funktionen beteiligt sind. Hier steht insbesondere deren einwandfreies Zusammenspiel im Fokus, da bereits der Wegfall einer Funktion zur einer erheblichen Störung des Prozesses führen kann. Die Grenzen zwischen beteiligten kognitiven und nicht kognitiven Funktionen sind dabei fließend, da ein kognitiver Prozess bereits mit der sensorischen Aufnahme beginnt und erst mit der aktorischen Ausgabe endet. Er ist aktiv in das reale Systemumfeld eingebunden. Das Resultat eines oder mehrerer kognitiver Prozesse kann als **kognitive Leistung** bezeichnet werden. Ist ein System in der Lage eine bestimmte kognitive Leistung gezielt zu erbringen, besitzt es die entsprechende **kognitive Fähigkeit**[49].

Für fortgeschrittene mechatronische Systeme, wie sie hier im Fokus stehen, ist z.B. die Selbstoptimierung eine wichtige kognitive Fähigkeit. Das System muss hierfür informationsverarbeitende Funktionen inkl. kognitiver Funktionen ausführen, die den (kognitiven) Selbstoptimierungsprozess bilden, bestehend aus Situationsanalyse, Zielbestimmung und Verhaltensanpassung. Die kognitive Leistung ist das umgesetzte verbesserte Systemverhalten.

4.3 Vorgehensmodell

Die *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* basiert im Kern auf einem Vorgehensmodell. Dieses bildet die methodische Grundlage der Entwicklungssystematik. Es strukturiert den Entwicklungsablauf und begleitet den Anwender während der Entwicklung, um so die systematische und letztlich erfolgreiche Integration kognitiver Funktionen in die Systemspezifikation sicherzustellen. Hierzu werden die Phasen des Vorgehensmodells beschrieben (Kap. 4.3.1). Im darauffolgenden Abschnitt wird dann das Vorgehensmodell – und somit die Entwicklungssystematik selbst – in den Entwicklungsprozess fortgeschrittener mechatronischer Systeme eingeordnet (Kap. 4.3.2).

4.3.1 Aufbau des Vorgehensmodells

Die Entwicklungssystematik zur Integration kognitiver Funktionen findet ihre Anwendung im Systementwurf in Sinne der VDI-Richtlinie 2206. Dieser ist der Kern der Konzipierungsaufgabe und findet in den frühen Phasen des Entwurfs statt (vgl. Kap. 2.4.3). In diesem Kapitel wird das Vorgehen der Entwicklungssystematik zunächst losgelöst von den übrigen Aufgaben in der Konzipierung fortgeschrittener mechatronischer Systeme vorgestellt. Das Vorgehen beschränkt sich nur auf die Schritte zur Integration kognitiver Funktionen und Spezifikation der entsprechenden Informationsverarbeitung.

[49] Ein System, das bspw. über die kognitive Fähigkeit des Lernens verfügt, führt die entsprechenden kognitiven Funktionen aus (vgl. Bild 2-12). Der Ablauf führt zu einem kognitiven Prozess und das Gelernte (z.B. eine Verhaltensweise) ist die kognitive Leistung.

Bild 4-3 zeigt das Vorgehen und die entsprechenden Schritte in Form eines Phasen-Meilenstein-Diagramms. Die Einordnung des Vorgehens in den übergeordneten Entwicklungsprozess erfolgt in Kapitel 4.3.2.

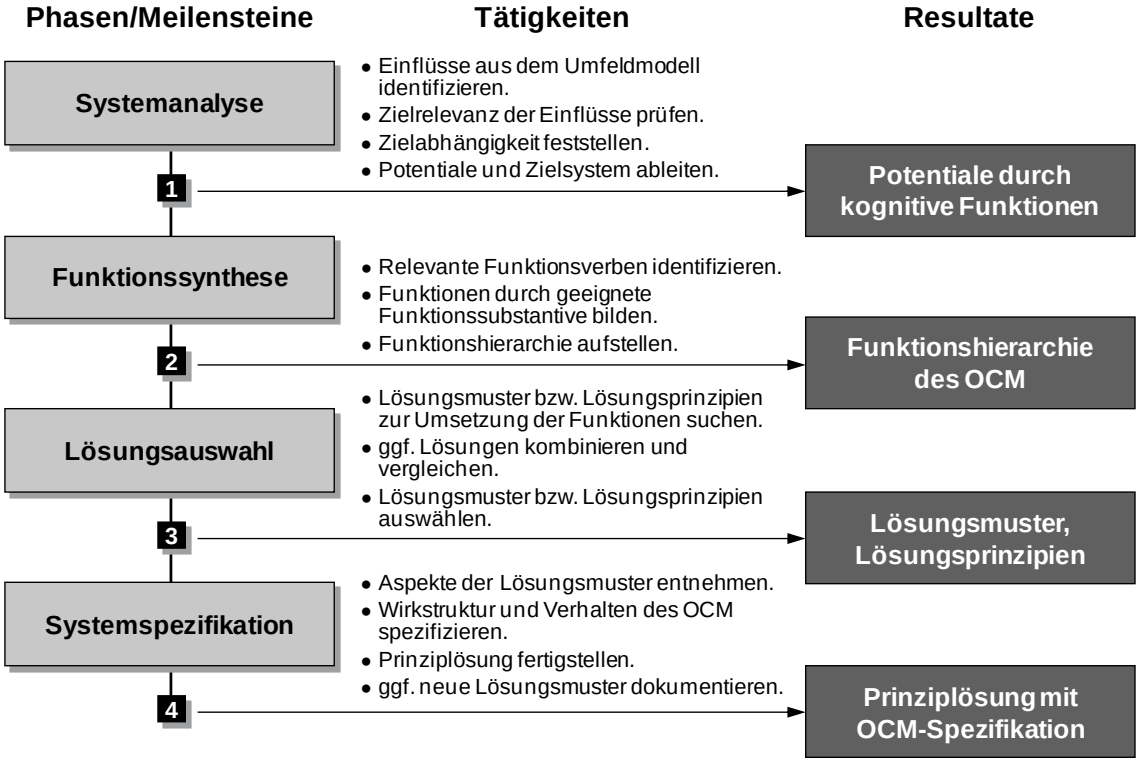


Bild 4-3: Vorgehensmodell zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme

Die Vorgehensweise bei der Anwendung der Entwicklungssystematik wird in einzelne Phasen und Meilensteine unterteilt. Das Vorgehensmodell gibt einen Überblick über die wesentlichen Tätigkeiten und zu erarbeitenden Resultate. Zudem wird die Reihenfolge der Durchführung der Phasen festgelegt. Dabei handelt es sich um eine sequentielle und idealtypische Darstellung, deren tatsächliche Anwendung aber Iterationen und auch Rücksprünge in allen Phasen zulässt.

Ausgangspunkt der Entwicklungssystematik ist der bisherige Stand der Entwicklung eines mechatronischen Systems, der noch keine Ausarbeitung der Informationsverarbeitung enthält. Dabei sollten eine Umfeld- und Anforderungsanalyse für das zu entwickelnde System erfolgt sein. Es ist aber auch möglich, dass zu diesem Zeitpunkt bereits Funktionen und Wirkstruktur des physikalischen Grundsystems beschrieben wurden (vgl. Kap. 3.2.3.3). Die Spezifikation der Informationsverarbeitung mit kognitiven Funktionen wird dann in vier Phasen erarbeitet.

Die erste Phase umfasst eine Analyse des zu entwickelnden Systems und der bisherigen Entwicklungsergebnisse aus der sich Nutzenpotentiale für den Einsatz kognitiver Funktionen ergeben können. Sind Potentiale vorhanden, werden in der zweiten Phase die zur Beschreibung des kognitiven Prozesses innerhalb des OCM notwendigen Funktionen

ermittelt und in einer hierarchischen Form spezifiziert. Die dritte Phase besteht aus der Suche und Auswahl geeigneter Lösungsmuster zur Erfüllung der vorher aufgestellten Funktionshierarchie. Zusätzlich können auch Lösungsprinzipien gesucht werden; im Falle der Informationsverarbeitung handelt es sich dabei um Verfahren (vgl. Kap. 4.5.1.1). In der letzten Phase werden mit Hilfe der Lösungsmuster und unter Berücksichtigung der bisherigen Systembeschreibung die Wirkstruktur und das Verhalten des OCM spezifiziert und in die Prinziplösung integriert. Mit der Bewertung und Freigabe der Prinziplösung endet die Konzipierung und somit die Anwendung der Entwicklungssystematik. Die OCM-Spezifikation und die ausgewählten Lösungsprinzipien bilden im weiteren Verlauf der Entwicklung die Basis für die softwaretechnische Implementierung der kognitiven Informationsverarbeitung.

Innerhalb der einzelnen Phasen werden Hilfsmittel zur Durchführung der Tätigkeiten angewendet. Das sind sowohl bestehende Hilfsmittel als auch solche, die im Rahmen dieser Arbeit neu entwickelt wurden. Bild 4-4 zeigt die neu entwickelten Hilfsmittel im Überblick, sowie in welcher Phase sie anzuwenden sind.

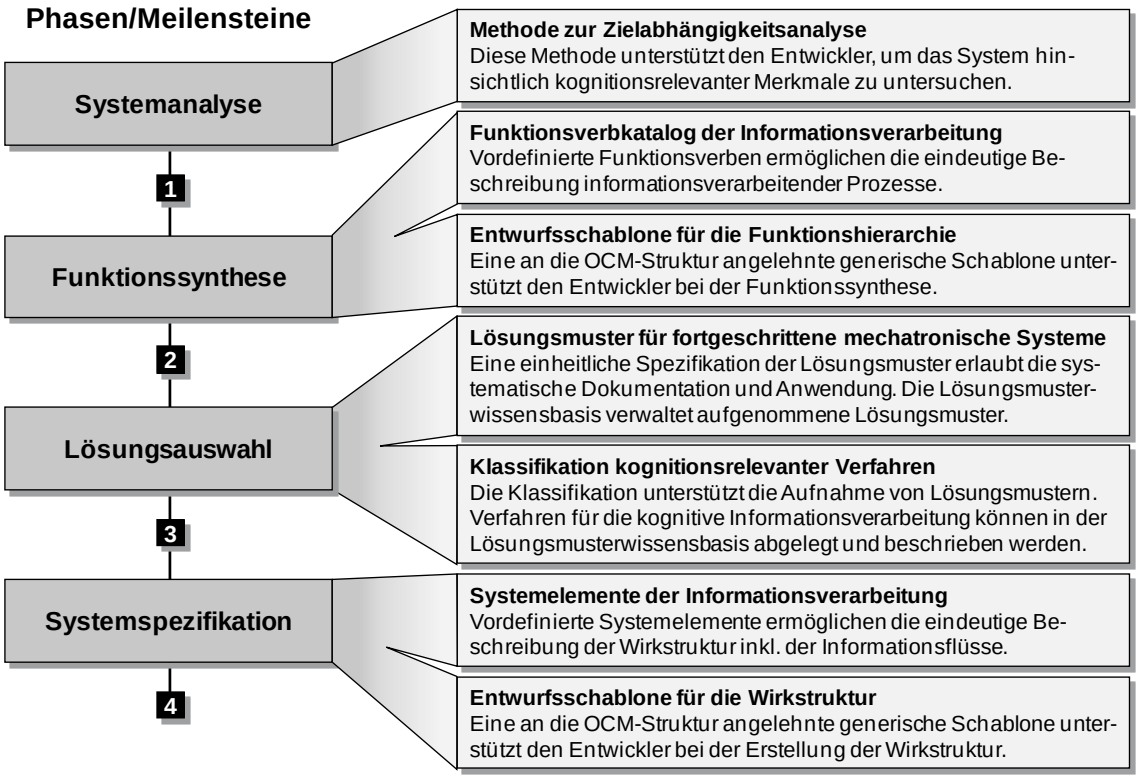


Bild 4-4: Einordnung der neu entwickelten Hilfsmittel in das Vorgehensmodell

In den nächsten vier Abschnitten werden die auszuführenden Tätigkeiten und zu erarbeitenden Resultate der einzelnen Phasen erläutert. Es werden Hilfsmittel zur Durchführung empfohlen und kurz erläutert. Da die Mehrzahl der neu entwickelten Hilfsmittel den anderen Bestandteilen der Entwicklungssystematik zugeordnet ist, wird auf diese an entsprechender Stelle verwiesen. Gleiches gilt für die im Stand der Technik bereits vorgestellte Methoden und Werkzeuge.

4.3.1.1 Phase 1: Systemanalyse

Ausgehend vom konkreten Entwicklungsauftrag des zu entwickelnden Systems und den bisher erarbeiteten Entwicklungsresultaten erfolgt eine systematische Analyse des Systems hinsichtlich der Relevanz einer fortgeschrittenen Regelung mit einer kognitiven Informationsverarbeitung. Ziel ist es, eine Aussage zu treffen, ob durch die Integration kognitiver Funktionen wesentliche Verbesserungen des Systemverhaltens zu erwarten sind. Diese Phase klärt folglich die Voraussetzungen für den Einsatz der Kognition und den entsprechenden Methoden. Sollten sich keine wesentlichen Nutzenpotentiale ergeben, besteht kein Bedarf für die Integration kognitiver Funktionen. Dennoch kann die Entwicklungssystematik zumindest zur Spezifikation der nicht kognitiven Ebenen des OCM eingesetzt werden.

Um Potentiale für den Einsatz kognitiver Funktionen zu erkennen, wird die *Methode zur Zielabhängigkeitsanalyse* eingesetzt. Diese wurde im Rahmen der Arbeit neu entwickelt und basiert auf der Methode zur Identifikation von Selbstoptimierungspotential nach GAUSEMEIER et al. [GZF+07], [ADG+09, S. 170ff.]. Mit deren Methode kann bei der Zusammenführung der Einzellösungen verschiedener Anwendungsszenarien die Notwendigkeit des Einsatzes des Wirkparadigmas der Selbstoptimierung festgestellt werden. Die Methode zur Zielabhängigkeitsanalyse verallgemeinert diesen Ansatz für fortgeschrittene mechatronische Systeme und setzt sich aus vier Phasen zusammen:

1) **Ermittlung zielrelevanter Einflussausprägungen**: In der ersten Phase sind, falls noch nicht geschehen, die Ziele des Systems zu identifizieren (z.B. geringer Energieverbrauch oder hohe Fahrsicherheit bei einem Fahrzeug). Ferner müssen mit einer Umfeldanalyse die Einflüsse ermittelt werden, die prinzipiell diese Ziele beeinflussen könnten; um beim Beispiel des Fahrzeugs zu bleiben, wären das bspw. Unebenheiten oder die Fahrermasse. Hierzu sollte ein Umfeldmodell mit der *Spezifikationstechnik des SFB 614* erstellt werden. Wichtig ist, dass mögliche Ausprägungen der Einflüsse bestimmt werden. Dabei genügt eine qualitative Unterscheidung; so kann die Fahrermasse groß oder klein sein.

2) **Abbildung der Zielauswirkung der Einflussausprägungen:** Im nächsten Schritt werden die Einflussausprägungen und die Ziele des Systems in Beziehung zueinander gestellt. Dabei interessiert, wie sich die Einflussausprägungen auf die Zielpriorität auswirken. Die Zielpriorität charakterisiert die Wichtigkeit eines Ziels unter dem gegeben Einfluss. Dementsprechend sollte eine Erhöhung der Zielpriorität mit einer Erhöhung der Zielgewichtung einhergehen. Hierfür eignet sich eine Zielprioritätsmatrix, in deren Zeilen die Einflüsse sowie ihre Ausprägungen eingetragen werden und in den Spalten die Ziele. In die Matrix kann dann die Zielprioritätsänderung aufgrund einer Einflussausprägung erfasst werden (starke Verringerung, Verringerung, keinen Auswirkung, Erhöhung und starke Erhöhung). Bild 4-5 zeigt einen Ausschnitt der Zielprioritätsmatrix des flexiblen Straßenfahrzeugs Chamäleon (vgl. Kap. 2.2.3).

Zielprioritätsmatrix
Fragestellung:
"Wie wirkt sich die Einflussausprägung i (Zeile) auf die Zielprorität der Systemziele j (Spalte) aus?"

Bewertungsmaßstab:
-- = starke Verringerung der Zielpriorität
- = Veringerung der Zielpriorität
0 = keine Veränderung der Zielpriorität
+ = Erhöhung der Zielpriorität
++ = starke Erhöhung der Zielpriorität

Einflüsse	Ausprägungen	Nr.	1 Geringer Energieverbrauch	2 Hohe Fahrsicherheit	3 Hoher Komfort	4 Geringer Reifenverschleiß	5 Erfüllung des Fahrerwunsches
		Systemziele					
Unebenheiten	vorhanden	1	0	+	+	0	0
	nicht vorhanden	2	0	0	0	0	0
Fahrermasse	groß	3	+	0	0	0	0
	klein	4	0	0	0	0	0
Lenkstrategie	übersteuern	5	0	+	0	0	0
	untersteuern	6	0	0	0	0	0
⋮							
Aktueller Energieverbrauch	hoch	23	+	0	0	0	0
	niedrig	24	0	0	0	0	0
negative Beschleunigung	hoch	25	0	0	0	-	0
	niedrig	26	0	0	0	0	0

Bild 4-5: Zielprioritätsmatrix des Chamäleons (Ausschnitt)

3) **Bildung relevanter Situationen:** In der dritten Phase werden aus der Zielprioritätsmatrix für den Betrieb nicht relevante Ziele und Einflussausprägungen gestrichen. Diese sind daran zu erkennen, dass eine Spalte (irrelevantes Ziel) oder eine Zeile (irrelevante Einflussausprägung) durchgehend neutral (also mit einer „0“) bewertet wurde. Danach müssen aus den übrigen Einflussausprägungen Situationen gebildet werden. Situationen sind konsistente Kombinationen der Einflüsse. Kombinationen von Einflussausprägungen, die in der Realität nicht vorkommen können, sind auszuschließen. So kann eine Situation nur eine Ausprägung eines Einflusses beinhalten. Für das Chamäleon ergeben sich dann bspw. 6912 Situationen[50].

4) **Auswertung der Situationsabhängigkeit der Ziele:** Im letzten Schritt werden für die jeweiligen Situationen die Zielprioritäten aus der Zielprioritätsmatrix zusammengezählt und eine situationsabhängige Priorität der einzelnen Ziele bestimmt. Daraus folgt der Grad der Zielabhängigkeit des Systems. So kann ein Ziel eine alleinige Priorität in einer bestimmten Situation besitzen oder sich diese mit einem oder mehreren anderen Zielen teilen. Werden die *Ziele über die Situationen hinweg unterschiedlich priorisiert*, ist das das erste Kriterium für die Entwicklung eines

[50] Aufgrund der vielen Kombinationsmöglichkeiten wurde in Visual Basic eine entsprechende automatisierte Situationsgenerierung und Auswertung implementiert.

fortgeschrittenen mechatronischen Systems, das zur Laufzeit in Abhängigkeit der jeweiligen Situation seine Ziele eigenständig bestimmt. Ferner ist der *Anteil der geteilten Ziele* von Bedeutung. Bei einem hohen Anteil geteilter Ziele deutet dies darauf hin, dass dieses Ziel in engem Bezug zu den anderen Zielen steht und seine Gewichtung nicht von vornerein und isoliert durch den Entwickler festgelegt werden sollte. Beim Chamäleon sind bspw. die Ziele „geringer Energieverbrauch“ und „Komfort“ sehr stark verknüpft und weisen einen Anteil von 52% bzw. 86% geteilter Zielprioritäten auf. Die Fahrsicherheit besitzt hingegen nur einen Anteil von 14%, was dafür spricht, dieses Ziel weniger flexibel und nicht durch das System autonom änderbar zu gestalten.

Nach der Analyse der Zielabhängigkeit wird, falls sich eine Zielabhängigkeit ergeben hat, das Zielsystem aufgestellt. Hierfür ist die *Spezifikationstechnik des SFB 614* anzuwenden. Im Zielsystem können nun die Beziehungen zwischen den Zielen und den entsprechenden Systemparametern festgehalten werden. Es ist die Grundlage für die spätere Implementierung des autonomen Zielverhaltens des Systems. Je nach Komplexität ergeben sich hieraus Potentiale für die Integration kognitiver Funktionen und entsprechender Verfahren zur Realisierung des Zielsystems. Zusammenfassend werden in der Phase Systemanalyse folgende **Hilfsmittel** eingesetzt:

- Methode zur Zielabhängigkeitsanalyse
- Spezifikationstechnik des SFB 614 und der Mechatronic Modeller als Werkzeugunterstützung (vgl. Kap. 3.2.3.3)

Das **Resultat** der Phase „Systemanalyse“ sind die *Potentiale durch die Integration kognitiver Funktionen* in das zu entwickelnde System. Dies umfasst den Grad der Zielabhängigkeit und das Zielsystem sowie ggf. eine erste informelle Beschreibung eines möglichen kognitiven Prozesses zur Umsetzung des Zielverhaltens.

4.3.1.2 Phase 2: Funktionssynthese

Da eine sofortige technische Lösung für den zu realisierenden kognitiven Prozess in der Regel nicht möglich ist, erfolgt eine erste abstrakte Beschreibung über Funktionen. Die funktionale Beschreibung ist entscheidend für die Problemklärung sowie für ein einheitliches Verständnis. Ihre Bedeutung nimmt noch zu, wenn Entwickler unterschiedlicher Disziplinen am Systementwurf beteiligt sind. Durch die abstrakte Definition der zu erzielenden kognitiven Funktionen, kann das gewünschte Systemverhalten disziplinübergreifend konkretisiert werden. Dazu müssen nicht nur die kognitive Informationsverarbeitung, sondern auch die untergeordneten Ebenen berücksichtigt werden. Dabei wird eine Funktionshierarche im Sinne der *Funktionsbeschreibung in der Produktentwicklung* aufgestellt, die an der Spitze eine komplexe Gesamtfunktion hat und dann in konkretere Teilfunktionen gegliedert wird. Ziel ist es, die Funktionshierarchie soweit zu

unterteilen, dass für die Funktionen auf der untersten Hierarchieebene eine Lösung gefunden werden kann.

Für die Definition der notwendigen kognitiven und nicht kognitiven Funktionen kann der hierfür neu entwickelte *Funktionsverbkatalog der Informationsverarbeitung* genutzt werden. Dieser definiert wesentliche Funktionsverben zur Beschreibung informationsverarbeitender Funktionen für fortgeschrittene mechatronische Systeme. Da deren Informationsverarbeitung im Rahmen dieser Arbeit durch den OCM umgesetzt wird, unterstützt eine entsprechend vorstrukturierte *Entwurfsschablone für die Funktionshierarchie* den Entwickler in diesem Schritt. Bei der Modellierung wird die *Spezifikationstechnik des SFB 614* eingesetzt. Die nachfolgende Aufzählung fasst die benötigten **Hilfsmittel** zusammen:

- Funktionsbeschreibung in der Produktentwicklung (vgl. Kap. 3.2.1.1)
- Funktionsverbkatalog der Informationsverarbeitung (vgl. Kap. 4.4.3)
- Entwurfsschablone für die Funktionshierarchie (vgl. Kap. 4.4.3)
- Spezifikationstechnik des SFB 614 und der Mechatronic Modeller als Werkzeugunterstützung (vgl. Kap. 3.2.3.3)

Das **Resultat** dieser Phase ist eine *Funktionshierarchie des OCM*, die einen oder mehrere kognitive Prozesse von der sensorischen Eingabe bis hin zur aktorischen Ansteuerung umfasst.

4.3.1.3 Phase 3: Lösungsauswahl

In der nächsten Phase sind für die Funktionen potentielle Lösungen zu ermitteln. Zu diesem Zweck müssen den Teilfunktionen auf der untersten Gliederungsebene der Funktionshierarchie Teillösungen zugeordnet werden, die diese und somit sukzessive auch die übergeordneten Funktionen bis hin zur Gesamtfunktion erfüllen. Im klassischen funktionsorientierten Entwurf wird hierfür ein *Morphologischer Kasten* nach ZWICKY eingesetzt [Zwi71]. Dieser ist ein Ordnungsschema, in dessen Zeilen die Teilfunktionen und in den Spalten mögliche Teillösungen eingetragen werden. Durch die Kombination der Teillösungen je Zeile werden Lösungsvarianten abgeleitet, die anschließend bewertet werden [Ehr07, S. 428ff.], [PBF+07, S. 136ff.].

Die Analyse des Stands der Technik legte offen, dass für informationsverarbeitende oder gar kognitive Funktionen bislang keine für die systematische Wiederverwendung dokumentierten Lösungen existieren. Daher wurde im Rahmen der Arbeit mit den *Lösungsmustern für fortgeschrittene mechatronische Systeme* eine Möglichkeit entwickelt, mit der entsprechende Lösungen für die eigene Wiederverwendung als auch für Dritte festgehalten werden können. Mit Hilfe von Experten wurden einige Lösungsmuster, insbesondere für den Entwurf der kognitiven Informationsverarbeitung, aufgenommen.

Diese sollten an dieser Stelle des Vorgehens zur Realisierung der Funktionshierarchie eingesetzt werden. Aufgrund der Komplexität der Lösungsmuster und des Umstands, dass die Lösungsmustersteller und die Lösungsmusternutzer aus verschiedenen Fachdisziplinen kommen, wurde eine Lösungsmusterwissensbasis konzipiert und prototypisch implementiert. Mit ihr können Lösungsmuster effektiv gesucht und ausgewählt werden. Da die Umsetzung der kognitiven Funktionen in der Regel auf Verfahren der künstlichen Intelligenz oder der höheren Mathematik beruhen, wurde eine *Klassifikation kognitionsrelevanter Verfahren* erstellt, die bei der direkten Suche nach einem Lösungsprinzip unterstützt, falls noch kein passendes Lösungsmuster vorhanden ist. Die in der Klassifikation enthaltenen Verfahren wurden in die Lösungsmusterwissensbasis aufgenommen und können im Hinblick auf eine Anwendung geprüft werden. Folgende Aufzählung umfasst die für diese Phase relevanten **Hilfsmittel**:

- Morphologischer Kasten
- Lösungsmuster für fortgeschrittene mechatronische Systeme und die Lösungsmusterwissensbasis als Werkzeugunterstützung (vgl. Kap. 4.5 und Kap. 4.6)
- Klassifikation kognitionsrelevanter Verfahren und die Lösungsmusterwissensbasis als Werkzeugunterstützung (vgl. Kap. 4.5 und Kap. 4.6)

Als **Resultat** der Lösungsauswahl liegen *geeignete Lösungsmuster* und/oder *Lösungsprinzipien* vor. Lösungsprinzipien für kognitive Funktionen sind grundsätzlich Verfahren der Informationsverarbeitung.

4.3.1.4 Phase 4: Systemspezifikation

In Anlehnung an die *Spezifikationstechnik des SFB 614* müssen für eine vollständige Systemspezifikation noch die Wirkstruktur und das Verhalten beschrieben werden. Hierzu sollte der *Mechatronic Modeller* genutzt werden, mit dem sämtliche Partialmodelle in Beziehung zueinander gesetzt werden können. Da nur noch Teile der Informationsverarbeitung spezifiziert werden, wird der Aspekt der physikalischen Gestalt zu diesem Zeitpunkt als vernachlässigbar angenommen. Selbstredend wird die Informationsverarbeitung später auf einer Hardware ausgeführt, welche jedoch erst während der tatsächlichen Implementierung ausgewählt wird. Zur Beschreibung der Wirkstruktur werden die entsprechenden Inhalte der Lösungsmuster konkretisiert. Dafür wurde eine *Entwurfsschablone für die Wirkstruktur* fortgeschrittener mechatronischer Systeme entwickelt. Sie orientiert sich an der OCM-Struktur und kann auch zur Dokumentation der Lösungsmuster eingesetzt werden. Ferner wurden ausgewählte *Systemelemente der Informationsverarbeitung* erarbeitet, die zur Spezifikation der Wirkstruktur eingesetzt werden. Für einzelne Systemelemente oder logische Gruppierungen von Systemelementen ist auch das Verhalten zu beschreiben. Hierzu ist wieder die *Spezifikationstechnik des SFB 614* anzuwenden, um Abläufe, Zustände und Zustandsübergänge des OCM zu

modellieren. Folgende Auflistung gibt einen Überblick über die zu verwendenden **Hilfsmittel**:

- Spezifikationstechnik des SFB 614 und der Mechatronic Modeller als Werkzeugunterstützung (vgl. Kap. 3.2.3.3)
- Entwurfsschablone für die Wirkstruktur (vgl. Kap. 4.4.2)
- Systemelemente der Informationsverarbeitung (vgl. Kap. 4.4.2)

Am Ende der Phase liegt als **Resultat** die *frühzeitige Spezifikation der Informationsverarbeitung in der OCM-Architektur* vor. Diese ist in die bestehende Spezifikation des Gesamtsystems zu integrieren. Für den Fall, dass eine bisher unbekannte (Teil-)Lösung entwickelt wurde, sollte diese abschließend in Form der Lösungsmusterspezifikation dokumentiert werden. Dabei muss die letztendliche Realisierung in den weiteren Entwurfsphasen berücksichtigt werden.

4.3.2 Einordnung in den Entwicklungsprozess

Die *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* wird in der Praxis nicht unabhängig von dem restlichen Entwicklungsgeschehen eingesetzt werden. Vielmehr ist sie begleitend anzuwenden. Daher muss ihr Vorgehen in den Entwicklungsprozess fortgeschrittener mechatronischer Systeme integriert sein. Von vornherein wurde die Entwicklungssystematik für die frühen Phasen des Entwurfs, also für die Konzipierung ausgelegt. Ein entsprechender Entwicklungsprozess für fortgeschrittene mechatronische Systeme wurde bereits in Kapitel 2.4.2 vorgestellt [GFD+08b, S. 97], [ADG+09, S. 169]. Hierfür kann das *Vorgehensmodell für die Konzipierung selbstoptimierender Systeme* in zwei Punkten vereinfacht dargestellt werden (vgl. Bild 2-16). Zum einen spielt im Rahmen dieser Arbeit die Produktmodularisierung eine untergeordnete Rolle, daher wird die Entwicklungssystematik ausschließlich in die Phase „Konzipierung auf Systemebene“ eingeordnet. Gleichwohl kann die Entwicklungssystematik auch zur Konzipierung einzelner Module angewendet werden. Unterschiede in der Anwendung existieren nicht.

Zum anderen spielt im Rahmen des Vorgehens der Entwicklungssystematik die Entwicklung mit Anwendungsszenarien, wie sie GAUSEMEIER et al. vorschlagen, eine untergeordnete Rolle. Aus rein methodischer Sicht kann diese in der Darstellung vernachlässigt werden. Bild 4-6 zeigt den entsprechenden vereinfachten Entwicklungsprozess fortgeschrittener mechatronischer Systeme und die Einordnung der vier Phasen des Vorgehensmodells der Entwicklungssystematik.

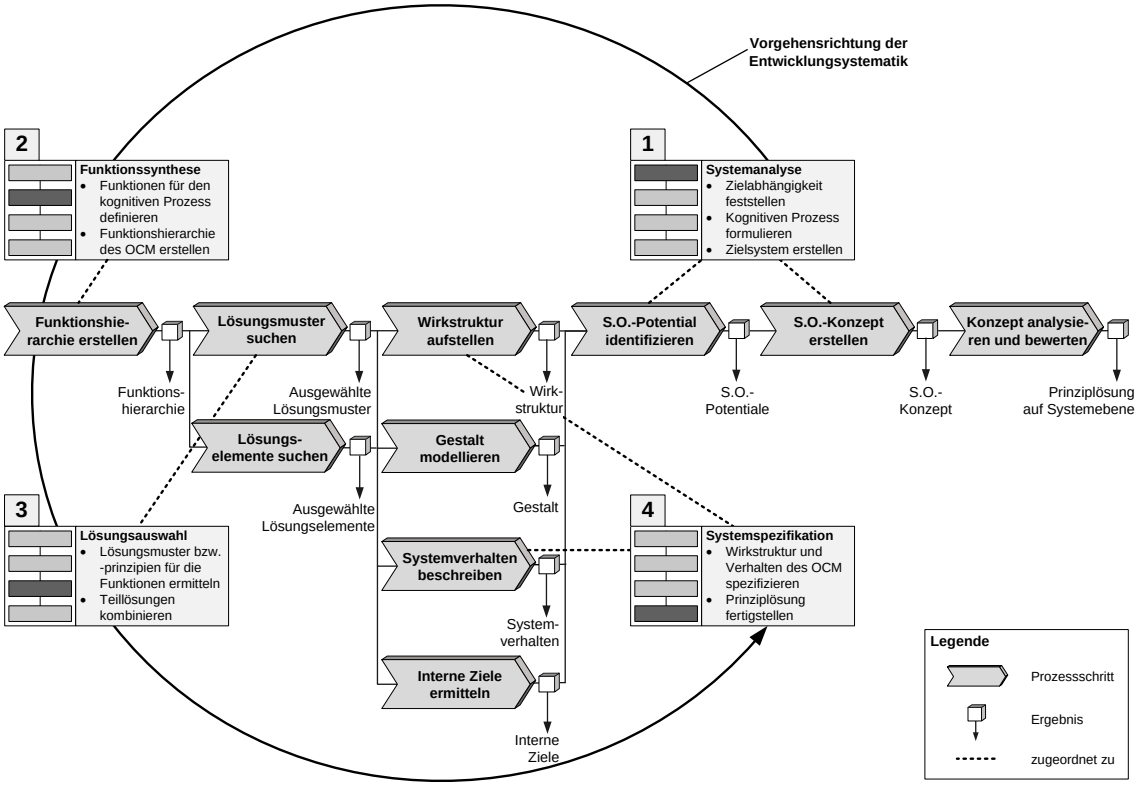


Bild 4-6: Vereinfachte Darstellung der Konzipierung auf Systemebene und Einordnung des Vorgehensmodells der Entwicklungssystematik

Systemanalyse: Die bisherigen Entwicklungsergebnisse erlauben ein detailliertes Verständnis der zu realisierenden Systemabläufe. Auf deren Basis werden Potentiale identifiziert, das zu entwickelnde System mit kognitiven Funktionen zu verbessern.

Funktionssynthese: In der zweiten Phase erfolgt ein iterativer Rücksprung im Entwicklungsprozess, indem die bestehende funktionale Systembeschreibung um die der Informationsverarbeitung des OCM modifiziert wird.

Lösungsauswahl: Für die neuen Funktionen der Funktionshierarchie werden auf Basis der Anforderungen und des Systemanalyse Lösungen gesucht und ausgewählt.

Systemspezifikation: Mit der letzten Phase ist der Rücksprung zu Ende, so dass nach der Spezifikation der Wirkstruktur und des Systemverhaltens die Spezifikation des OCM in die Prinziplösung integriert werden kann.

Die Prinziplösung ist das Ergebnis der Konzipierung und Ausgangspunkt für die weitere domänenspezifische Ausarbeitung und Integration aller Entwicklungsergebnisse. Die Spezifikation des OCM wird insbesondere in den Domänen Regelungs- und Softwaretechnik implementiert und umgesetzt (vgl. Bild 2-15).

4.4 Systembeschreibung

Für die frühzeitige Spezifikation der Informationsverarbeitung wird die Spezifikationstechnik des SFB 614 eingesetzt. Ferner wird das Operator-Controller-Modul (OCM) als Architektur der Informationsverarbeitung beschrieben, das in den späteren Entwurfsphasen implementiert wird. In diesem Kapitel wird zunächst die Spezifikation fortgeschrittener mechatronischer Systemen erläutert (Kap. 4.4.1). Aus kognitionswissenschaftlicher Sicht ist zu klären, wie die Wirkstruktur der Informationsverarbeitung zu spezifizieren ist (Kap. 4.4.2). Im Anschluss steht die funktionale Beschreibung des OCM auf Basis allgemeiner Funktionsverben der Informationsverarbeitung im Fokus (Kap. 4.4.3). Sowohl für die strukturelle als auch für die funktionale Spezifikation des OCM wird jeweils eine Entwurfsschablone erarbeitet und vorgestellt.

4.4.1 Spezifikation fortgeschrittener mechatronischer Systeme

In der Analyse des Stands der Technik wurde die Spezifikationstechnik des SFB 614 als geeignete Technik zur domänenübergreifenden Beschreibung der Prinziplösung fortgeschrittener mechatronischer Systeme identifiziert (vgl. Kap. 3.2.3.3). Diese unterscheidet mehrere Partialmodelle, die jeweils einen bestimmten Aspekt des zu entwickelnden Systems fokussieren. Zusammengenommen ergeben die Partialmodelle ein vollständiges Modell des Systems. Hierzu stehen die Partialmodelle in enger Beziehung zueinander. Ferner sind bestimmte Partialmodelle derart von anderen abhängig, dass ihre Erstellung erst nach einem ersten Entwurf des oder der anderen erfolgen kann. Für die Integration von kognitiven Funktionen in fortgeschrittene mechatronische Systeme sind nicht alle Partialmodelle gleichwichtig. So kann das Partialmodell *Gestalt* im Rahmen der Entwicklungssystematik vernachlässigt werden. Auch das Partialmodell *Anwendungsszenarien* spielt eine untergeordnete Rolle, fällt aber nicht ganz weg. Die zentralen Partialmodelle sind *Funktionen* und *Wirkstruktur*, die mit dem Großteil der anderen Partialmodelle in Beziehung stehen. Bild 4-7 zeigt, vereinfacht dargestellt, das Zusammenwirken der Partialmodelle und deren Verwendung zur Systembeschreibung als Bestandteil der Entwicklungssystematik zur Integration kognitiver Funktionen.

Die Partialmodelle **Umfeld**, **Anwendungsszenarien** und **Anforderungen** liegen in der Regel zu Beginn der Anwendung der Entwicklungssystematik bereits vor. Das Umfeldmodell ist notwendig, um relevante Einflüsse und deren Ausprägungen zu ermitteln. Grundsätzlich können Widersprüche zwischen unterschiedlichen Anwendungsszenarien und deren Lösungskonfigurationen zu Potentialen für die Integration kognitiver Funktionen führen [ADG+09, S. 170]. Zudem werden in der Lösungsmusterspezifikation Systemelemente, die nicht direkt zur Lösungsumsetzung beitragen, als Umfeldelemente gekennzeichnet (vgl. Kap. 4.5.1). Es werden nur Anforderungen betrachtet, die sich auf die Ausführung von informationsverarbeitenden Prozessen beziehen. Diese unterstützen bei der Auswahl geeigneter Lösungsmuster oder Lösungsprinzipien, indem sie mit deren Merkmalen verglichen werden können. Die einzuhaltenden Anforderungen sind im

Rahmen der Systembeschreibung mit den entsprechenden Funktionen und Systemelemente zu verknüpfen.

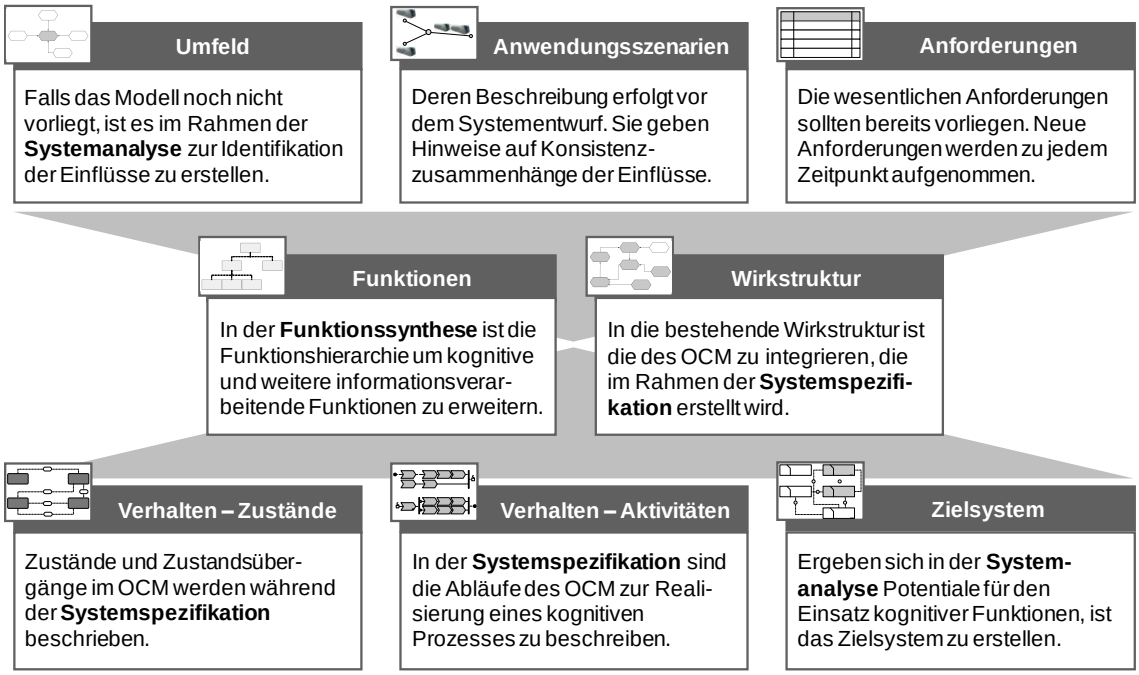


Bild 4-7: Verwendung der Spezifikationstechnik des SFB 614 im Rahmen der Entwicklungssystematik zur Integration kognitiver Funktionen

Kern der Systembeschreibung zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme sind die Partialmodelle **Funktionen** und **Wirkstruktur**. Die einzelnen Funktionen bestehen aus einem Funktionssubstantiv und Funktionsverb. Diese Funktionen werden in hierarchischer Form aufgestellt, beginnend mit der komplexesten Funktion, die dann immer weiter untergliedert werden kann. Im Mittelpunkt stehen die Funktionen, die das OCM ausführen soll. Sie werden durch Systemelemente realisiert, die in der Wirkstruktur dargestellt werden. Die Wirkstruktur beschreibt zudem das Zusammenwirken der Systemelemente. Während die Funktionsbeschreibung weitestgehend lösungsneutral erfolgt, ergibt sich durch die Wahl der Systemelemente und deren Verknüpfungen eine erste Konkretisierung in Richtung Lösung. Aufgrund ihrer Bedeutung im Rahmen der Entwicklungssystematik werden beide Partialmodelle in den Kapiteln 4.4.2 und 4.4.3 detailliert behandelt.

Die vollständige Wirkungsweise des OCM wird erst durch die Spezifikation seines Verhaltens klar. Mit dem Partialmodell **Verhalten – Aktivitäten** werden die Systemabläufe innerhalb des OCM beschrieben. Prinzipiell beschreiben Aktivitäten die Ausführung der Funktionen. Die Verknüpfung mehrerer Aktivitäten ergibt sog. Aktivitätsstränge. Aktivitätsstränge können den Systemelementen oder Gruppen von Systemelementen direkt zugewiesen werden. Im Fokus steht dabei die Spezifikation der Abläufe des kognitiven Operators. Das Partialmodell **Verhalten – Zustände** umfasst die möglichen Zustände, die ein System oder Teile des Systems einnehmen können. Sie können über-

geordnete Betriebsmodi des Systems festlegen, was im Hinblick auf die Informationsverarbeitung von hoher Bedeutung ist. Grundsätzlich kann aber jedem Systemelement ein Zustandsmodell zugewiesen werden – im einfachsten Fall „an“ oder „aus“ bzw. „nicht defekt“ oder „defekt“.

Das Partialmodell **Zielsystem** beschreibt die Optimierungsziele des Systems (vgl. Kap. 2.2.3) und in welchen Abhängigkeiten diese zueinander stehen. Es handelt sich dabei um das initiale Zielsystem, das später dem zu entwickelnden System implementiert wird. Die Bewertung der Ziele kann das System zur Laufzeit noch ändern. Das Zielsystem charakterisiert folglich die Systemintentionalität, die durch kognitive Funktionen erzielt wird. Grundsätzlich können die drei Zielbeziehungsarten *positiv*, *negativ* und *neutral* unterschieden werden. Positiv heißt, dass sich die Ziele gegenseitig fördern, so dass die Erfüllung des einen Ziels zwangsläufig mit der Erfüllung eines anderen Ziels oder zumindest einer wesentlichen Zielannäherung einhergeht. Stehen Ziele in Konflikt wird deren Beziehung als negativ gekennzeichnet. Voneinander unabhängige Ziele oder Ziele, deren Beziehung nicht eindeutig definiert werden kann, werden als neutral bezeichnet. Ihre Beziehung muss nicht explizit im Zielsystem festgehalten werden.

Im Folgenden werden die beiden zentralen Partialmodelle zur Systemspezifikation hinsichtlich der Beschreibung der informationsverarbeitenden Abläufe systematisch angepasst. Die durchgängige Anwendung der Spezifikationstechnik erfolgt im Rahmen der Validierung der Entwicklungssystematik in Kapitel 5.

4.4.2 Wirkstruktur der Informationsverarbeitung

Vor dem Hintergrund der Problemanalyse und den untersuchten Strukturkonzepten (vgl. Kap. 3.2.2) wird ersichtlich, dass die Systemarchitektur fortgeschrittener mechatronischer Systeme mehrere Ebenen der Informationsverarbeitung unterscheiden muss. Grund hierfür ist, dass die kognitive Informationsverarbeitung nicht die reaktive ersetzen darf, sondern diese erweitern muss. Grundlegende Funktionen, die insbesondere die Systemsicherheit und den Systemerhalt gewähren, benötigen eine unflexible, schnelle und störungsfreie Informationsverarbeitung. Diese Informationsverarbeitungsebene sollte nur gezielt manipuliert werden.

Das Dreischichtenmodell der Verhaltenssteuerung nach STRUBE wird diesem basalem Anspruch gerecht. Es unterteilt sich in die Bereiche *nicht kognitive Regulierung*, *assoziative Regulierung* und *kognitive Regulierung* (vgl. Bild 2-13). Diese drei Ebenen repräsentieren die Vorgänge, die sich im Gehirn eines Lebewesens, wie bspw. dem Menschen, abspielen. Nach STRUBE ist das Dreischichtenmodell, zumindest aus kognitionswissenschaftlicher Sicht, auch die grundlegende Struktur technischer kognitiver Systeme. Im Rahmen der vorliegenden Arbeit ist sie Ausgangspunkt für eine strukturelle Beschreibung einer Informationsverarbeitung zur Integration kognitiver Funktionen.

Bild 4-8 zeigt das Dreischichtenmodell zur Verhaltenssteuerung in der Wirkstrukturnotation der zuvor beschriebenen Spezifikationstechnik des SFB 614. Zudem sind informationsverarbeitende (inkl. kognitiver) Funktionen, wie sie in der Kognitionswissenschaft beschrieben werden, den Systemelementen der Wirkstruktur zugeordnet.

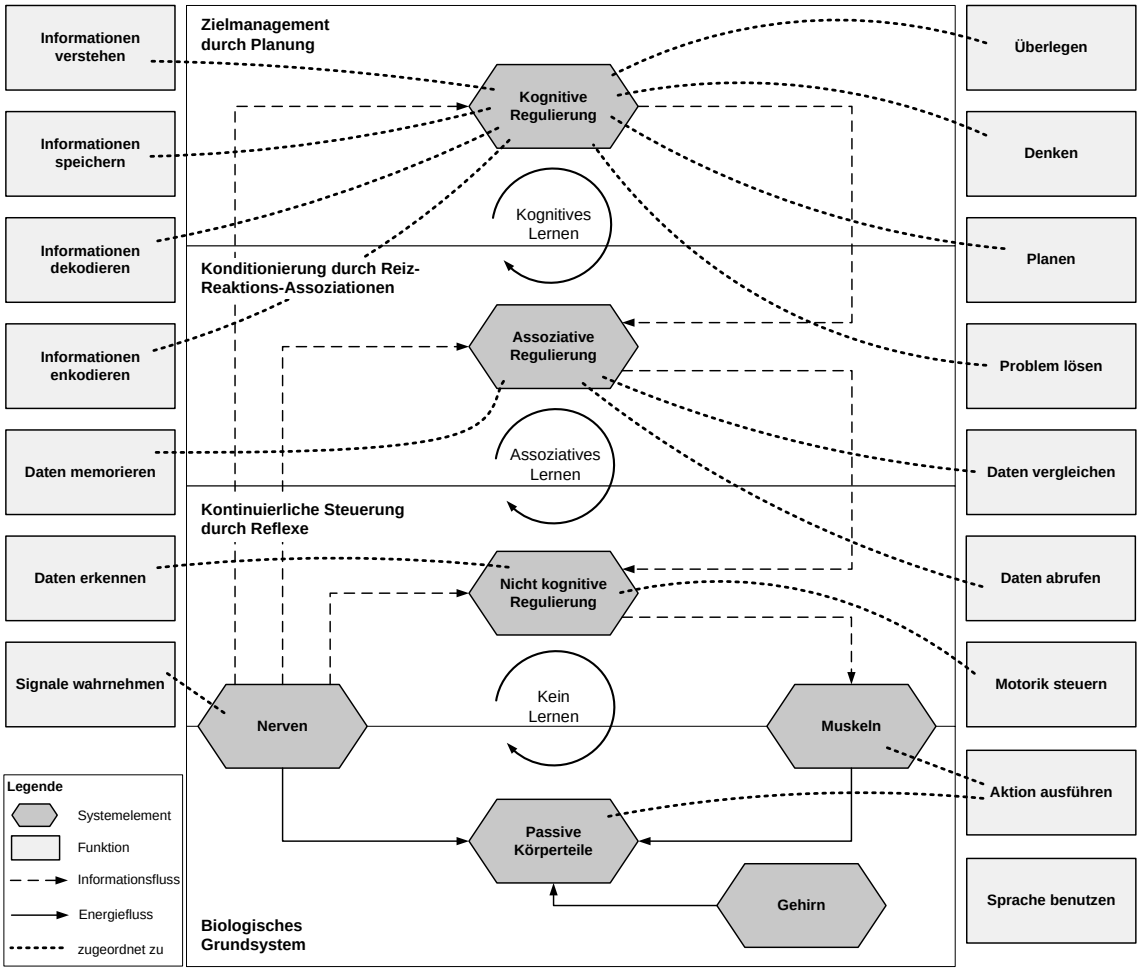


Bild 4-8: Wirkstruktur der Informationsverarbeitung aus kognitionswissenschaftlicher Sicht in Anlehnung an das Dreischichtenmodell nach STRUBE

Um eine Übertragung auf die Technik zu ermöglichen, werden unterhalb der nicht kognitiven Regulierung ausführende physische Systemelemente dargestellt. Diese Darstellung ist eine starke Vereinfachung der realen Anatomie eines Lebewesens, das in der Realität eine enorme Komplexität aufweist. So gliedert sich der betrachtete Körper als **biologisches Grundsystem** im Wesentlichen in die Systemelemente *passive Körperteile* sowie *Gehirn*. Das Gehirn wird als Organ explizit aufgeführt, da es das Zentrum der Verhaltenssteuerung und somit die „Hardware“ der Informationsverarbeitung ist. Sämtliche Energieflüsse der betrachteten Elemente symbolisieren dabei deren physische Verbindung.

Die grundlegenden Schnittstellen zwischen der Informationsverarbeitung und dem Körper eines biologischen Grundsystems sind *Muskeln* und *Nerven*. Letztere sind zwar auch für die Weiterleitung von Signalen sowie das Innervieren der Muskulatur verantwort-

lich, ihre wesentliche Funktion im Hinblick auf die Kognition ist aber das *Wahrnehmen von Signalen*. Die Muskeln selbst *führen* im Zusammenspiel mit den entsprechenden passiven Körperteilen *Aktionen aus*.

Die unterste Ebene des Dreischichtenmodells steht für die starre Kopplung zwischen Nerven und Muskeln. Es handelt sich dabei um eine **kontinuierliche Steuerung durch Reflexe**. Die nicht kognitive Regulierung muss hierfür *Daten erkennen* und die *Motorik steuern*. Ab dieser Stelle nimmt die in Kapitel 4.2 eingeführte Unterscheidung von Daten und Informationen eine zentrale Rolle ein, da das Funktionssubstantiv die Komplexität einer Funktion maßgeblich bestimmt[51]. Auf dieser Ebene findet kein Lernen statt; die motorische Steuerung basiert auf fest implementierten Prozessen.

Auf der mittleren Ebene erfolgt eine **Konditionierung durch Reiz-Reaktions-Assoziationen**. Diese beruht auf der klassischen und der operanten Konditionierung (vgl. Kap. 2.3.2). Die assoziative Regulierung muss hierfür zunächst *Daten memorieren* können. Zudem müssen *Daten abgerufen* und *verglichen* werden.

Die oberste Ebene ist das **Zielmanagement durch Planung**, deren zentrales Element die kognitive Regulierung ist. Diese erfüllt die komplexeste Form des Lernens, das auf den klassischen geistigen kognitiven Funktionen wie *Planen*, *Denken*, *Problem lösen* und *Überlegen* beruht. Hierzu müssen *Informationen enkodiert*, *gespeichert*, wieder *dekodiert* und *verstanden* werden. Die Enkodierung nimmt dabei eine zentrale Rolle ein, da sie wesentlich für die Informationsgewinnung verantwortlich ist.

Die Funktion *Sprache benutzen* kann in der Wirkstruktur nicht einer bestimmten Ebene zugeordnet werden. Es handelt sich um eine Funktionalität, die sich aus einer Vielzahl verschiedener Funktionen zusammensetzt und in diesem Sinne eher als kognitive Leistung bzw. Fähigkeit zu sehen ist. Bei der Übertragung auf technische Systeme muss diese Funktion im Sinne einer Kommunikationsfunktionalität berücksichtig werden.

Die Bild 4-8 dargestellte kognitionsorientierte Wirkstruktur verdeutlicht, wie grundlegende informationsverarbeitende Funktionen und kognitive Funktionen in ihrer Wirkungsweise zusammenhängen. Jede einzelne Ebene reguliert dabei auf eine eigene Art und Weise das Verhalten des Menschen. Hinsichtlich einer Übertragung auf technische Systeme sind allerdings sowohl die Systemelemente als auch die Funktionen teilweise noch zu abstrakt. Daher wird im Folgenden die kognitionswissenschaftliche Sicht auf die technikorientierte OCM-Architektur übertragen. Das Ergebnis der Integration der kognitionswissenschaftlichen Sicht in die OCM-Struktur ist in Bild 4-9 abgebildet. Es handelt sich um eine Wirkstruktur, die mit spezifischen Systemelementen und Flussausprägungen konkretisiert werden kann. Aufgrund ihres generischen Charakters ist sie

[51] So beschreibt die Funktion *Daten erkennen* eine einfache informationsverarbeitende Funktion und die Funktion *Informationen erkennen* eine kognitive Funktion. Zur besseren Unterscheidung wird diese in Bild 4-8 als *Informationen verstehen* spezifiziert.

zugleich eine **Entwurfsschablone für die Wirkstruktur** fortgeschrittener mechatronischer Systeme, die in der Phase der Systemspezifikation einzusetzen ist.

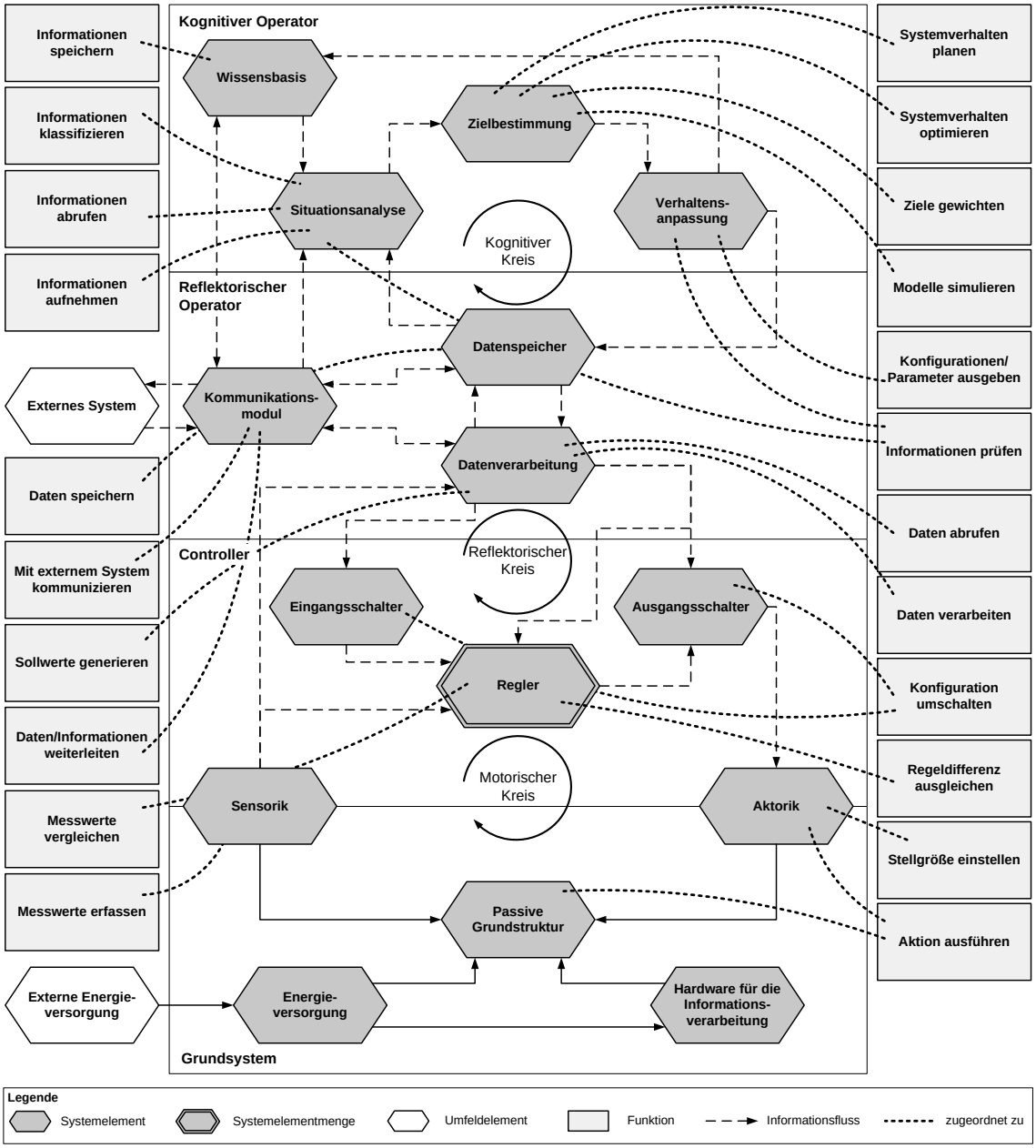


Bild 4-9: Generische Wirkstruktur fortgeschrittener mechatronischer Systeme

Die OCM-Architektur unterscheidet drei Ebenen für die Informationsverarbeitung fortgeschrittener mechatronischer Systeme. Die physikalischen Systemelemente befinden sich im **Grundsystem (GS)** unterhalb der Informationsverarbeitung. Dieses lässt sich in die Systemelemente *passive Grundstruktur*, *Hardware für die Informationsverarbeitung* sowie *Energieversorgung* unterteilen. Die entsprechenden Analogien aus der kognitionswissenschaftlichen Wirkstruktur sind die *passiven Körperteile* sowie das *Gehirn* (vgl. Bild 4-8). Die Energieversorgung als explizites Element wird erst beim Aufbau einer grundsätzlichen Struktur für technische Systeme relevant, daher wurde diese bei

Lebewesen vernachlässigt. Die Energieflüsse (durchgezogene Pfeile), die von der Energieversorgung abgehen, symbolisieren die Versorgungsenergie. Alle übrigen Energieflüsse des Grundsystems stehen für die Befestigung der einzelnen Elemente an der passiven Grundstruktur. Die Schnittstellen zwischen dem Grundsystem zu der Informationsverarbeitung bilden die Sensorik und die Aktorik. Sie sind die technische Umsetzung der in Bild 4-8 dargestellten Nerven und Muskeln und erfüllen daher die Funktionen *Messwerte erfassen* und *Aktion ausführen*.

Der **Controller (CO)**, das Pendant zur untersten Ebene des Dreischichtenmodells, realisiert die nicht kognitive Regulierung. Wesentliche Aufgabe des *Reglers* ist die Verbesserung des Systemverhaltens durch Ausregelung von Störgrößen, die auf das Grundsystem einwirken. Hierzu bestimmt er per *Vergleich* die *Regeldifferenz* zwischen vorgegebenem Sollwert (Führungsgröße) und der gemessenen Regelgröße (Rückführgröße) und *gleicht* diese *aus*. Aus regelungstechnischer Sicht ist die Aktorik integraler Bestandteil der Stelleinrichtung und erfüllt somit die Funktion *Stellgröße einstellen*, um auf das Grundsystem entsprechend der Regeldifferenz einzuwirken.

Der CO realisiert eine starre Kopplung zwischen der Sensorik und der Aktorik, deren Wirkungsablauf als *motorischer Kreis* bezeichnet wird. Dieser kann durch Einwirken der höheren Ebenen manipuliert werden. Dies erfordert teilweise aber auch weitere strukturelle Voraussetzungen im CO. Fortgeschrittene mechatronische Systeme besitzen im Normalfall eine Regelreinrichtung mit mehr als nur einem Regler. Durch einen *Eingangsschalter* und einen *Ausgangsschalter* können unterschiedliche Reglerkonfigurationen für bestimmte Referenzen *umgeschaltet* werden. Diese müssen aber nicht zwangsläufig spezifiziert werden, da sie keine ausführenden Systemelemente sind [Hes06, S. 37].

Im **reflektorischen Operator (RO)** kann die assoziative Regulierung, wie das klassische oder operante Konditionieren, durch entsprechende Algorithmen umgesetzt werden. Der RO kann so als lernende, dem CO übergelagerte Steuerung realisiert werden. Ein Beispiel ist die Integration einer Adaptionseinrichtung (vgl. Kap. 2.2.2).

Durch den RO verläuft jedoch eine weitere Trennlinie. Es handelt sich dabei um die Unterteilung der Verarbeitungsgeschwindigkeit in weiche und harte Echtzeit (vgl. Kap. 3.2.2.2). Die Untersuchungen bestehender Systeme haben gezeigt, dass assoziative Funktionen in weicher Echtzeit ablaufen und in der Regel zur Planung neuer Systemziele eingesetzt werden. Daher wurde diese Funktionalität ausschließlich dem kognitiven Operator zugeordnet. Unter Berücksichtigung des Dreischichtenmodells sind dies jedoch Funktionen, die sich oberhalb der Trennlinie im Bereich der weichen Echtzeit des RO befinden. Grundsätzlich läuft der Großteil der Berechnungen im RO unter harten Echtzeitbedingungen ab – im Gegensatz zum CO jedoch nicht zeitdiskret, sondern ereignisdiskret. Das Systemelement *Datenverarbeitung* repräsentiert diese Berechnungen, die sich in den generischen Funkionen *Daten abrufen*, *verarbeiten* und *weiterleiten* unterteilen. Hinzu kommt die konkrete Funktion *Konfigurationen* zur Reglerumschaltung

an Controller *auszugeben*. Auf diese Weise steuert der RO die untere Ebene auf Basis der Informationen vom kognitiven Operator. So ist seine primäre Aufgabe situationsgerechte und unter Umständen neue *Sollwerte* für die Regelung zu *generieren*. Dieser Ablauf wird als *reflektorischer Kreis* bezeichnet, in dem auch die sensorischen Eingaben von unten durchgereicht werden.

Im Hinblick auf den reflektorischen Kreis ist die Datenverarbeitung im Verlauf der OCM-Spezifikation weiter zu konkretisieren und ggf. in mehrere Systemelemente aufzuteilen, die verschiedene Funktionen erfüllen (z.B. Konfigurationssteuerung, Überwachung oder Korrekturelement). Ferner ist der externe Zugriff auf die Informationsverarbeitung fortgeschrittener mechatronischer Systeme primär im RO zu sehen. Über ein *Kommunikationsmodul* findet die *Kommunikation mit externen Systemen* statt, um wichtige Daten oder Informationen auszutauschen. Ferner ist das Kommunikationsmodul – zumindest in der generischen Form – auch eine zentrale interne Schnittstelle zur *Weiterleitung* von *Daten* bzw. *Informationen*. Der *Datenspeicher* hinterlegt Daten für eine weitere Verarbeitung. Es handelt sich dabei um eines der wenigen nicht aktiven Systemelemente in der generischen Wirkstruktur.

Die kognitive Regulierung wird im **kognitiven Operator (KO)** realisiert. Die Verarbeitung der anfallenden Rechenaufgaben erfolgt grundsätzlich in weicher Echtzeit, wobei zwischen Offline- und Online-Berechnungen unterschieden wird. Auf dieser Ebene des OCM findet der Kern der kognitiven Informationsverarbeitung und entsprechender Prozesse statt. Im Fokus fortgeschrittener mechatronischer Systeme steht die Realisierung des Selbstoptimierungsprozesses durch modellorientierte oder verhaltensorientierte Verfahren. Die letztgenannten Verfahren werden dabei durch problemspezifische Softwarealgorithmen realisiert. Die modellorientierte Optimierung ist hingegen nur möglich, wenn das Verhalten des gesamten Systems (bzw. der relevanten Systemteile) durch mathematische Gleichungen ausgedrückt werden kann (vgl. Kap. 4.5.3).

Die beteiligten Systemelemente zur Umsetzung dieser Verfahren sind in Anlehnung an den Selbstoptimierungsprozess die *Situationsanalyse*, die *Zielbestimmung* und die *Verhaltensanpassung*. Die benötigten *Informationen* zur Optimierung werden dabei durch die Situationsanalyse *aufgenommen*, *geprüft* und *klassifiziert*. Diese Informationen können die aktuelle Strategie des gesamten Systems sein, auf dessen Basis die Berechnungen unter harter Echtzeit erfolgen. Außerdem sind aktuelle, aufbereitete Sensordaten und Daten von externen Systemen erforderlich. Die kurzfristige Aufnahme von Informationen ist für ein autonomes und intelligentes Systemverhalten nicht ausreichend. Daher *speichert* eine *Wissensbasis* die wichtigsten *Informationen* und vernetzt diese im Betrieb zu internem Wissen. Diese *Informationen* werden ebenfalls von der Situationsanalyse *abgerufen*.

Das Ergebnis der Verarbeitung durch die Situationsanalyse geht in die Zielbestimmung. In dieser befindet sich das systeminterne Zielsystem, dessen *Ziele* zur Laufzeit *gewichtet werden*. Hierzu *plant* die Zielbestimmung das zu erwartende *Systemverhalten* in be-

stimmten Situationen. Die Gewichtung kann dann auf Basis dieser Pläne erfolgen. Ein anderer Weg, der alternativ oder auch in Kombination möglich ist, ist die Zielgewichtung über die *Simulation* eines mathematischen *Modells* zu berechnen und so das *Systemverhalten zu optimieren*.

Die Ergebnisse der Zielbestimmung werden an die Verhaltensanpassung weitergegeben. Dieses Systemelement greift dabei nicht direkt auf die Regelung bzw. das Grundsystem mit der dazugehörigen Aktorik zu. Vielmehr stellt es die geplante und optimierte Systemstrategie im RO ein. Diese Kopplung zwischen RO und KO wird *kognitiver Kreis* genannt, der maßgeblich durch die *Ausgabe* der entsprechenden *Systemparameter* (z.B. Bewertungsgrößen oder Optimierungswerte) oder relevanter *Konfigurationen* charakterisiert ist. In der Regel *prüft* die Verhaltensanpassung die Ausgabe, ehe diese an den Datenspeicher im RO weitergeleitet wird.

Die Wirkstruktur beschreibt das Zusammenspiel aller beteiligten Systemelemente. Sie ist daher der Kern der OCM-Spezifikation. Die einzelnen Systemelemente der Wirkstruktur erfüllen hierfür eine oder mehrere Funktionen und beschreiben eine erste Lösung. Die Systemelemente nehmen somit in der OCM-Spezifikation die Rolle der Funktionserfüller und Lösungsträger ein. Die Festlegung der Systemelemente ist daher von entscheidender Bedeutung.

Die ausgearbeitete, generische Wirkstruktur der Informationsverarbeitung eines fortgeschrittenen mechatronischen Systems verdeutlicht, dass die Systemelemente alle in der Kognitionswissenschaft definierten kognitiven Funktionen erfüllen. Sowohl die Systemelemente als auch die auszuführenden Funktionen besitzen aber einen wesentlich technischeren Charakter als in der Wirkstruktur des Dreischichtenmodells. Für die Konkretisierung der generischen Wirkstruktur werden im Folgenden ausgewählte **Systemelemente der Informationsverarbeitung** vorgestellt. Diese Systemelemente wurden im Rahmen des SFB 614 bisher vermehrt und wiederkehrend zur Beschreibung von Prinziplösungen eingesetzt. Die Kurzbeschreibungen der Systemelemente wurden mit Entwicklern aus verschiedenen Fachdisziplinen aufgestellt, um ihre disziplinübergreifende Anwendung sicherzustellen. Dabei können die aufgelisteten Systemelemente verschiedene spezifische Aufgaben wahrnehmen. Entsprechende Beispiele sind in den Kurzbeschreibungen angeführt. Tabelle 4-1 zeigt diese Auswahl an Systemelementen der Informationsverarbeitung, die den Ebenen des OCM zugeordnet wurden. Auf eine Darstellung von unterschiedlichen haptischen Systemelementen des Grundsystems wird im Rahmen dieser Arbeit verzichtet.

Tabelle 4-1: Ausgewählte Systemelemente (SE) der Informationsverarbeitung

Systemelement	Kurzbeschreibung	OCM-Ebene
Ablaufsteuerung	Das SE steuert die Ausführung nachgeschalteter SE. Notwendige Bedingungen sind bereits fest vorimplementiert, daher handelt sich in der Regel um eine koordinierende Routine.	RO/KO
Analyseelement	Das SE analysiert die übermittelten Daten. Anhand der Ergebnisse werden entsprechende weitere SE angesteuert oder Daten weitergeleitet (z.B. Gefahrenanalyse, Umfeldanalyse).	RO
Aufbereiter	Derartige SE bereiten die eingehenden Informationsflüsse auf. Ggf. erfolgt auch eine Konvertierung. Messwertaufbereiter oder ein Bildkonvertierer sind konkrete Beispiele.	CO/RO
Beobachter	Das SE rekonstruiert aus Eingangsgrößen und Messgrößen nicht messbare Größen eines Systems auf Grundlage eines Modells.	RO
Datenspeicher	Das SE speichert Daten, bspw. in Form einer Datenbank für Notfallstrategien oder Optimierungswerte.	RO/KO
Interpreter	Das SE initialisiert die Ausführung einer ausgewählten Regel und leitet neue Regeln oder andere Informationen ab.	KO
Kommunikationsmodul	Das SE bildet eine Kommunikationsschnittstelle. Dabei können externe (anderes System oder Teilsystem) und interne Kommunikationspartner (andere SE) unterschieden werden.	RO
Konfigurationssteuerung	Das SE definiert, bei welchem Systemzustand welche Konfiguration gültig ist. Darüber hinaus legt es fest, unter welchen Bedingungen zwischen Konfigurationen umgeschaltet wird.	RO
Korrekturelement	Auf Basis einer Simulation eines echtzeitfähigen Modells werden zuvor bestimmte Systemparameter für die Regelung korrigiert.	RO
Optimierer	Das SE wendet bestimmte Optimierungsverfahren an, um geeignete Ziele und Systemparameter zu berechnen, die zu einer zielkonformen Verhaltensanpassung führen.	RO/KO
Planer	Das SE plant verschiedene Aktionsfolgen um ein bestimmtes Ziel zu erreichen. Unterschiedliche Planer können eingesetzt werden (z.B. bedingter oder wissensbasierter Planer)	KO
Sollwertgenerierung	Auf Grundlage der Ergebnisse vorangegangener Analysen, Messungen oder Schätzungen werden Sollwerte für den CO generiert bzw. vorgeben.	RO
Überwachung	Das SE überwacht gezielt bestimmte Systemparameter oder SE wie Sensoren. Im Gegensatz zum Analyseelement kann die Überwachung nur beschränkt selbst aktiv werden.	RO
Wissensbasis	Informationen wie auch Pläne und Strategien werden gespeichert und für den Abruf bereitgestellt. Die Informationen werden derart abgelegt, dass sie Beziehung zueinander stehen.	KO

4.4.3 Funktionen der Informationsverarbeitung

Die funktionale Beschreibung geht der Definition der Wirkstruktur im Entwicklungsprozess voraus. Es ist daher von großer Bedeutung eine eindeutige und dennoch weitestgehend lösungsneutrale Spezifikation der Funktionen zu unterstützen. Methoden der Funktionsbeschreibung in der Produktentwicklung erfüllen diesen Anspruch, berücksichtigen aber nicht die Beschreibung der Informationsverarbeitung. So sollte die hierarchische Gliederung der Gesamtfunktion zur Spezifikation der Gesamtfunktionalität des zu entwickelnden Systems genutzt werden. Es fehlt aber ein an die Informationsver-

arbeitung technischer Systeme angepasstes Vorgehen und geeignete Funktionsverben[52] der Informationsverarbeitung, wie sie für physikalische Zusammenhänge bspw. von PAHL/BEITZ, LANGLOTZ oder BIRKHOFER erarbeitet wurden (vgl. Kap. 3.2.1.1).

Grundlage für die Beschreibung von funktionalen Zusammenhängen der Informationsverarbeitung in fortgeschrittenen mechatronischen Systemen ist das sog. **EVA-Prinzip**. Es ist der Grundsatz der elektronischen Datenverarbeitung (EDV) und unterscheidet die drei Phasen **E**ingabe, **V**erarbeitung und **A**usgabe. Das Schema orientiert sich an der klassischen Von-Neumann-Rechnerarchitektur für einen Computer. Zunächst erfolgt die Eingabe von Daten z.B. durch die Tastatur. Danach verarbeitet der Prozessor diese Daten und gibt sie anschließend bspw. am Monitor aus. Das Speichern von Daten kann der zweiten Phase zugewiesen werden [Dud01, S. 232], [Dwo89, S. 44ff.]. Das EVA-Prinzip beschreibt im Wesentlichen die Datenverarbeitung zwischen Benutzer und Computer oder zwischen zwei Computern. Es lässt sich aber auch auf die Interaktion informationsverarbeitender Systemelemente übertragen. Hierzu kann das Akronym EVA weiter genutzt werden, allerdings mit angepasster Bedeutung. Das angepasste EVA-Prinzip umfasst die folgenden **allgemeinen Funktionen der Informationsverarbeitung,** die durch informationsverarbeitende Systemelemente erfüllt werden müssen:

- **Information erfassen:** Ein Systemelement erfasst Informationen. Es ist nicht definiert, ob die Informationsaufnahme durch direkt (z.B. durch Messen) oder indirekt (z.B. durch Abrufen) stattfindet.
- **Information verarbeiten:** Ein Systemelement verarbeitet Informationen. Dies kann auf vielfältige Weise erfolgen und hängt maßgeblich von der Aufgabe des Systemelements ab.
- **Information ausgeben:** Ein Systemelement gibt Informationen aus. Die Ausgabe von Informationen kann selbstständig oder erst auf Anfrage erfolgen.

Die Beschreibungen der allgemeinen Funktionen zeigen, dass diese hinsichtlich der Funktionalität des Systemelements noch keine genaue Aussage zulassen. Zur weiteren Konkretisierung bedarf es folglich einer Festlegung von Funktionen, die den einzelnen allgemeinen Funktionen der Informationsverarbeitung untergeordnet sind. In den nachfolgenden Tabellen sind Funktionsverben zur Bildung einer informationsverarbeitenden Funktion im Sinne des angepassten EVA-Prinzips aufgeführt. Aufgrund des natürlichen Sprachgebrauchs, der stets in einem Wandel steht, können diese keinen Anspruch auf Vollständigkeit haben. Im Rahmen der Werkzeugunterstützung in Kapitel 4.6 werden den Funktionsverben Synonyme zugeordnet, um die Vergleichbarkeit zu erhöhen.

---

[52] Funktionen werden erst durch die Kombination eines Verbs mit einem Substantiv gebildet. Im Weiteren liegt der Fokus nur auf der Definition geeigneter Funktionsverben der Informationsverarbeitung. Als Substantiv kann grundsätzlich der Begriff Information stehen. Dieser schließt Daten als unverarbeitete Informationen mit ein.

Ziel ist ein **Funktionsverbkatalog der Informationsverarbeitung**, der eine einheitliche Funktionsbeschreibung erleichtert. Die entsprechenden Tabellen enthalten hierzu eine Kurzbeschreibung, die die Verwendung des Funktionsverbs festlegt. Dies geschieht in Anlehnung an die Prozesswortlisten nach RUPP, die derartige Listen für ein einheitliches Verständnis unter den beteiligten Personen in einem Projekt vorschlägt (vgl. Kap. 3.2.1.2). Ferner werden die Ebenen des OCM angeführt, in denen die aus dem Verb resultierende Funktion umgesetzt wird. Tabelle 4-2 zeigt Funktionsverben der allgemeinen Funktion *Information erfassen*.

Tabelle 4-2: Funktionsverben für die Funktion „Information erfassen“

Funktionsverb	Kurzbeschreibung	OCM-Ebene
abrufen	Ein SE ruft aktiv Informationen aus einer Datenbank ab. Die Informationen werden zwangsläufig ausgeben.	RO/KO
aktualisieren	Ein SE aktualisiert z.B. eine Datenbank, indem Informationen ergänzt oder angepasst werden.	RO/KO
anfragen	Ein SE fragt aktiv Informationen an. Ein weiteres SE wird aktiv, um diese zu senden.	RO/KO
empfangen	Ein SE empfängt Informationen, ohne diese explizit anzufordern.	CO/RO/KO
messen	Ein SE misst physikalische Größen des Grundsystems.	CO

Das „Erfassen“ unterscheidet sich im Wesentlichen in der Aktivität der beteiligten Systemelemente. Anschließend erfolgt die Verarbeitung der erfassten Informationen. In Tabelle 4-3 sind Funktionsverben der Funktion „Information verarbeiten“ dargestellt.

Tabelle 4-3: Funktionsverben für die Funktion „Information verarbeiten“

Funktionsverb	Kurzbeschreibung	OCM-Ebene
aufbereiten	Bestimmte Rohdaten müssen für eine weitere Informationsverarbeitung aufbereitet werden.	CO/RO
aufnehmen	Ein SE nimmt eine bestimmte Information auf und veranlasst das gezielte Ablegen dieser Information für eine weitere Verarbeitung.	RO/KO
aufteilen	Ein SE teilt die Informationen anhand bestimmter Merkmale auf. Die Summe des Outputs entspricht der Summe des Inputs.	RO/KO
ausgleichen	Diese Funktion beschreibt die auszugleichende Regeldifferenz durch das SE Regler.	CO
auswählen	Ein SE wählt aus gespeicherten Informationen Alternativen aus.	RO/KO
filtern	Bestimmte Informationen werden gezielt vernachlässigt und spezielle für die Weitergabe gefiltert.	CO/RO/KO
konvertieren	Ein SE konvertiert Daten (z.B. Bilder) in ein anderes Format.	CO/RO
löschen	Ein SE löscht eine bestimmte Information aus einer Datenbank oder einem Speicher.	RO/KO
speichern	Ein SE speichert aktiv Informationen über einen längeren Zeitraum. Dies erfolgt z.B. in einer Datenbank oder Wissensbasis.	RO/KO
vergleichen	Eine erfasste Information mit einer gespeicherten Informationen oder einer anderen erfassten vergleichen.	CO/RO/KO

Es wird hier nur eine Auswahl grundlegender Verarbeitungsfunktionen gezeigt, die im Rahmen der Systemkonzipierung eingesetzt werden können. Es handelt sich dabei um elementare Verarbeitungsschritte informationsverarbeitender Systemelemente. Nach der Verarbeitung müssen wieder Informationen ausgegeben werden. Tabelle 4-3 enthält Verben und entsprechende Kurzbeschreibungen zur Konkretisierung der allgemeinen Funktion „Information ausgeben".

Tabelle 4-4: Funktionsverben für die Funktion „Information ausgeben"

Funktionsverb	Kurzbeschreibung	OCM-Ebene
aktivieren	Ein SE aktiviert ein anderes SE, indem es die Aktivität eines anderen SE freigibt.	CO/RO
ändern	Ein SE ändert die Daten eines anderen SE. In der Regel handelt es sich dabei um einen Datenspeicher.	RO/KO
bereitstellen	SE stellt Informationen zum Abruf durch ein anderes SE bereit.	CO/RO/KO
deaktivieren	Ein SE deaktiviert auf Basis der erfassten oder verarbeiteten Informationen ein anderes SE.	CO/RO
einstellen	Ein bestimmter Datensatz wird in einer Datenbank eingestellt. Im CO beschreibt die Funktion das Einstellen einer Stellgröße.	CO/RO/KO
informieren	Ein SE informiert gezielt ein fest definiertes SE und erhält eine Rückmeldung.	RO
senden	Ein SE sendet bestimmte Informationen an ein anderes SE. Auf diese Informationsausgabe erfolgt kein Feedback.	CO/RO
verteilen	Ein SE verteilt Informationen an mehrere Empfänger.	RO
weiterleiten	Ein SE leitet Informationen direkt weiter. An diesem SE findet keine Verarbeitung der Informationen statt.	RO

Grundsätzlich ist die Art und Weise der Informationsausgabe zu unterscheiden. Zum einen können nur Informationen an andere Systemelemente weitergegeben werden, die dann entscheiden, wie diese zu verwerten sind (z.B. *weiterleiten* oder *senden*). Zum anderen kann das ausgebende Systemelement selbst eine Aktion eines anderen Systemelements veranlassen (z.B. *aktivieren* oder *einstellen*).

Basierend auf Untersuchungen bestehender Systeme ergeben sich viele informationsverarbeitende Funktionen, wie z.B. auch der Großteil der Funktionen der kognitiven Informationsverarbeitung im KO, erst aus dem Zusammenwirken aus Funktionen der drei EVA-Phasen. Derartige Funktionen können als **komplexe Funktionen der Informationsverarbeitung** bezeichnet werden. Sie hängen stark von dem zugrundeliegenden Verfahren ab. Mit ihnen können relativ schnell und einfach kognitive Prozesse beschrieben werden, ohne die Informationsverarbeitung bis auf die elementare EVA-Ebene herunterzubrechen. Tabelle 4-5 zeigt Funktionsverben für komplexe Funktionen der Informationsverarbeitung fortgeschrittener mechatronischer Systeme.

Tabelle 4-5: Funktionsverben für komplexe Funktionen der Informationsverarbeitung

Funktionsverb	Kurzbeschreibung	OCM-Ebene
analysieren	Informationen werden auf einzelne Merkmale analysiert. Bestimmte Zusammenhänge werden erfasst und Inferenzen daraus abgeleitet.	KO
anpassen	Bestimmte Informationen werden abgerufen und verglichen; Vorgabeparameter werden entsprechend geändert.	KO
aufzeichnen	Ein SE zeichnet Daten auf, indem diese über einen längeren Zeitraum beobachtet und abgelegt werden.	RO/KO
beobachten	Beobachten erfolgt durch Erfassung einer Größe anhand eines Modells. Anwendungsfall der Regelungstechnik.	CO/RO
bestimmen	Vorhandene Parameter oder Ziele werden aufgrund erhaltener Daten/Informationen neu bestimmt.	CO/RO/KO
bewerten	SE bewertet Informationen nach Qualität oder Wichtigkeit.	RO/KO
delegieren	Ein SE delegiert eine Aufgabe, indem es diese untersucht, aufteilt und die Teilaufgaben an entsprechende SE überträgt.	RO/KO
erkennen	Ein SE erkennt Daten/Informationen durch die Zuordnung einer Signaländerung auf eine bekannte Informationsgröße.	RO/KO
erzeugen	Anhand erhaltener Informationen werden neue Informationen oder Ausprägungen dieser erzeugt. Ein Spezialfall ist duplizieren.	CO/RO/KO
generieren	Generieren von Inputdaten z.B. Sollwerten. Dies erfolgt größtenteils durch Berechnungen.	RO
gewichten	Ein SE kann Ziele im Rahmen der Zielbestimmung neu gewichten.	KO
identifizieren	SE identifiziert z.B. Parameter aufgrund erhaltener Informationen.	RO/KO
klassifizieren	SE klassifiziert eine Menge von Informationen auf Basis vorhandener Informationen.	KO
kommunizieren	Ein SE kommuniziert mit einem anderen SE, indem es Informationen austauscht.	RO/KO
lernen	SE lernt durch Erfahrungen, Wissen und Verhalten.	KO
lokalisieren	Ein SE lokalisiert Informationen, wie z.B. aktuelle Systemzustände.	RO/KO
optimieren	Ein SE optimiert, indem es die günstigsten Lösungen für bestimmte Zielstellungen ermittelt.	KO
planen	SE plant die Lösung eines Problems durch Erstellung eines Plans oder mehrerer Pläne.	KO
regeln	Das SE misst und vergleicht Daten. Auf Basis der gewonnenen Erkenntnisse werden Einstellungen vorgenommen.	CO
sortieren	Ein SE sortiert Informationen, indem es diese nach bestimmten Merkmalen ordnet.	RO
suchen	Auf Grundlage von Parametern werden Informationen z.B. in einer Datenbank gesucht.	RO/KO
überwachen	Das SE überwacht Daten indem diese erfasst und geprüft oder fortlaufend verglichen werden. Die Ergebnisse werden weitergegeben.	RO
umschalten	Das SE wechselt die Konfiguration, indem es zwischen ihnen umschaltet.	RO
verstehen	Ein SE ist in der Lage, Informationen zu verstehen. Dies erfolgt durch Zurückgreifen auf bestehende Informationen.	KO

Ausgehend von der generischen Wirkstruktur, die das Ergebnis der Übertragung der kognitionswissenschaftlichen Sicht auf die OCM-Struktur ist, und den erarbeiteten Funktionsverben der Informationsverarbeitung, kann eine generische Funktionshierar-

chie fortgeschrittener mechatronischer Systeme definiert werden. Diese kann als **Entwurfsschablone für die Funktionshierarchie** im Rahmen der Funktionssynthese genutzt werden (vgl. Bild 4-10, Bild 4-11 und Bild 4-12).

In Anlehnung an die generische Wirkstruktur gliedert sich ein fortgeschrittenes mechatronisches System zunächst in die Funktionsmodule[53] *Grundsystem inkl. Aktorik, Controller inkl. Sensorik, reflektorischer Operator* und *kognitiver Operator*. Die Sensorik und Aktorik werden dem Controller bzw. dem Grundsystem zugewiesen, da die Sensorik die Schnittstelle zur Informationsverarbeitung ist und die Aktorik direkt auf das Grundsystem wirkt. Komplettiert wird diese Ebene durch die *Benutzerschnittstelle*, die hier fakultativ aufgeführt wird. Eine allgemeingültige Zuordnung der Benutzerinteraktion mit einem fortgeschrittenen mechatronischen System steht nicht im Fokus der Arbeit. Grundsätzlich ist es denkbar, dass Vorgaben direkt über eine installierte Sensorik aufgenommen werden. Eine weitere Möglichkeit wäre die Eingabe von Strategien über eine Schnittstelle zur Informationsverarbeitung. Aus diesen Gründen erhält diese Funktionalität eine Sonderstellung in der Funktionshierarchie.

Grundsystem inkl. Aktorik: Die generische Wirkstruktur aus Kapitel 4.4.2 zeigt, dass das Grundsystem aus passiven und aktiven Systemelementen besteht. Die Zuordnung der passiven Elemente zu dem Bereich des Grundsystems basiert auf den Elementen des klassischen Maschinenbaus und ergibt die Grundstruktur des Systems. Die Entkopplung der Aktorik von dem Controller beruht auf der Überlegung, dass die Entwicklung von ausführenden Systemelementen, wie bspw. einem Gleichstrommotor, auf Wirkprinzipien der Elektrotechnik und des Maschinenbaus basiert. Dabei stehen eher Aufbau, Gestalt und Wirkungsweise im Vordergrund, als die Verbindung des Motors zur Informationsverarbeitung (vgl. Bild 4-10).

Controller inkl. Sensorik: Die Sensoren bilden die Schnittstelle zwischen Grundsystem und Controller. Im Gegensatz zur Aktorik wird die Sensorik der Funktionalität der Informationsverarbeitung zugeordnet. Beim Entwurf des gesamten Systems steht nicht im Vordergrund, wie die Sensordaten im Detail erzeugt werden, sondern vielmehr, welche Daten die Informationsverarbeitung benötigt. Die wesentliche Hauptfunktion des Controllers ist es, das *Systemverhalten zu regeln*, die sich in *Daten vergleichen* und *Reglerausgangsgröße einstellen* weiter unterteilt. Für die Regelung müssen entsprechende *Werte gemessen* werden. Weitere Funktionen wie *Sensordaten ausgeben* oder die Teilfunktionen der Funktion *Sensordaten erzeugen* werden vollständigkeitshalber aufgeführt. Eine Modellierung im Rahmen der Konzipierung ist jedoch nicht zwingend notwendig. Das *Ergebnis der Regelung* sollte an den reflektorischen Operator *weitergeleitet* werden (vgl. Bild 4-10).

---

[53] Im Sinne von übergeordneten funktionserfüllenden Systemelementen. Ohne letztlich den Systemzweck zu kennen, können diese Funktionsmodule nur als Systemelement dargestellt werden und nicht durch Funktionen in aktivistischer Form (vgl. Kap. 3.2.1.1).

Reflektorischer Operator: Die Aufgaben und die spätere Implementierung des reflektorischen Operators sind größtenteils der Softwaretechnik zuzuordnen. Das grundsätzliche Verhalten einzelner Systemelemente lässt sich dabei auf die Funktionen *Daten erfassen, speichern, verarbeiten* und *ausgeben* eingrenzen. Dies entspricht dem angepassten EVA-Prinzip, wobei das Speichern, z.B. in einem Datenspeicher, eine Sonderstellung einnimmt. Weitere Teilfunktionen sind möglich und konkretisieren die zu erfüllende Funktionalität des reflektorischen Operators. So bezieht sich das Erfassen von Daten bspw. auf die benötigten Daten der Sensoren oder auf Daten von externen Systemen. Ferner kann die Art der Datenerfassung detailliert werden (z.B. *Daten abrufen, empfangen* oder *anfragen*). Die anschließende Verarbeitung ist die Wertschöpfung der jeweiligen Systemelemente. Diese lässt sich z.B. weiter unterteilen in *Daten aufnehmen, generieren, vergleichen* oder auch *löschen*. Mit der Ausgabe von Daten im Sinne der Softwareentwicklung sind u.a. Zusammenhänge, wie das *Aktivieren* bzw. *Deaktivieren* von *Systemelementen*, bzw. den entsprechenden Schnittstellen, oder auch nur das simple *Weiterleiten* von *Daten* gemeint (vgl. Bild 4-11).

Die Kombination der erläuterten Funktionen spiegelt sich in spezifischen Ausprägungen der benötigten Systemelemente für die Konzipierung eines fortgeschrittenen mechatronischen Systems wider (vgl. Tabelle 4-1), die auch übergeordnete, komplexe Funktionen der Informationsverarbeitung erfüllen (vgl. Tabelle 4-5). Dies sind z.B. Funktionen wie *Umschalten von Regelkonfigurationen, Beobachten/Schätzen von Daten*, die nicht direkt messbar sind und auf Basis mathematischer Zusammenhänge erzeugt werden können, *Generieren von Sollwerten, Analysieren von Gefahren, Überwachen von Systemabläufen* oder auch *Kommunizieren mit weiteren Systemen*. Für diese übergeordneten Funktionen können entsprechende Lösungen in Form von Lösungsmustern erstellt werden (vgl. Kap. 4.5.1).

Kognitiver Operator: Der kognitive Operator ist das wesentliche Element der kognitiven Informationsverarbeitung und die entscheidende Ebene zur Integration kognitiver Funktionen. Im Kontext fortgeschrittener mechatronischer Systeme kann die Strukturierung seiner Funktionalität in Anlehnung an den Selbstoptimierungsprozess erfolgen. Die oberste Hierarchieebene gliedert sich in die Funktionen *Situation analysieren, Ziele bestimmen* und *Verhalten anpassen*. Die Analyse der Situation umfasst im Wesentlichen zwei Aufgaben. Zunächst müssen die benötigten *Informationen erfasst* und *aufgenommen* werden. Zudem findet eine Aufbereitung der Informationen für die Zielbestimmung statt (z.B. *Informationen vergleichen* oder *klassifizieren*). Das Bestimmen von Zielen ist Kern der kognitiven Informationsverarbeitung. Diesbezüglich werden Funktionen benötigt wie *Systemverhalten planen, Ziele gewichten, Modelle simulieren* oder *Systemverhalten optimieren.* Die Verhaltensanpassung unterteilt sich grundsätzlich in die *Ausgabe* aller betroffenen Informationen, wie *Konfigurationen* oder *Parameter* und deren *Überwachung*. Alle erzeugten und erfassten *Informationen* sind in einer Wissensbasis *zu speichern*, die die Informationen in Beziehung setzten kann (vgl. Bild 4-12).

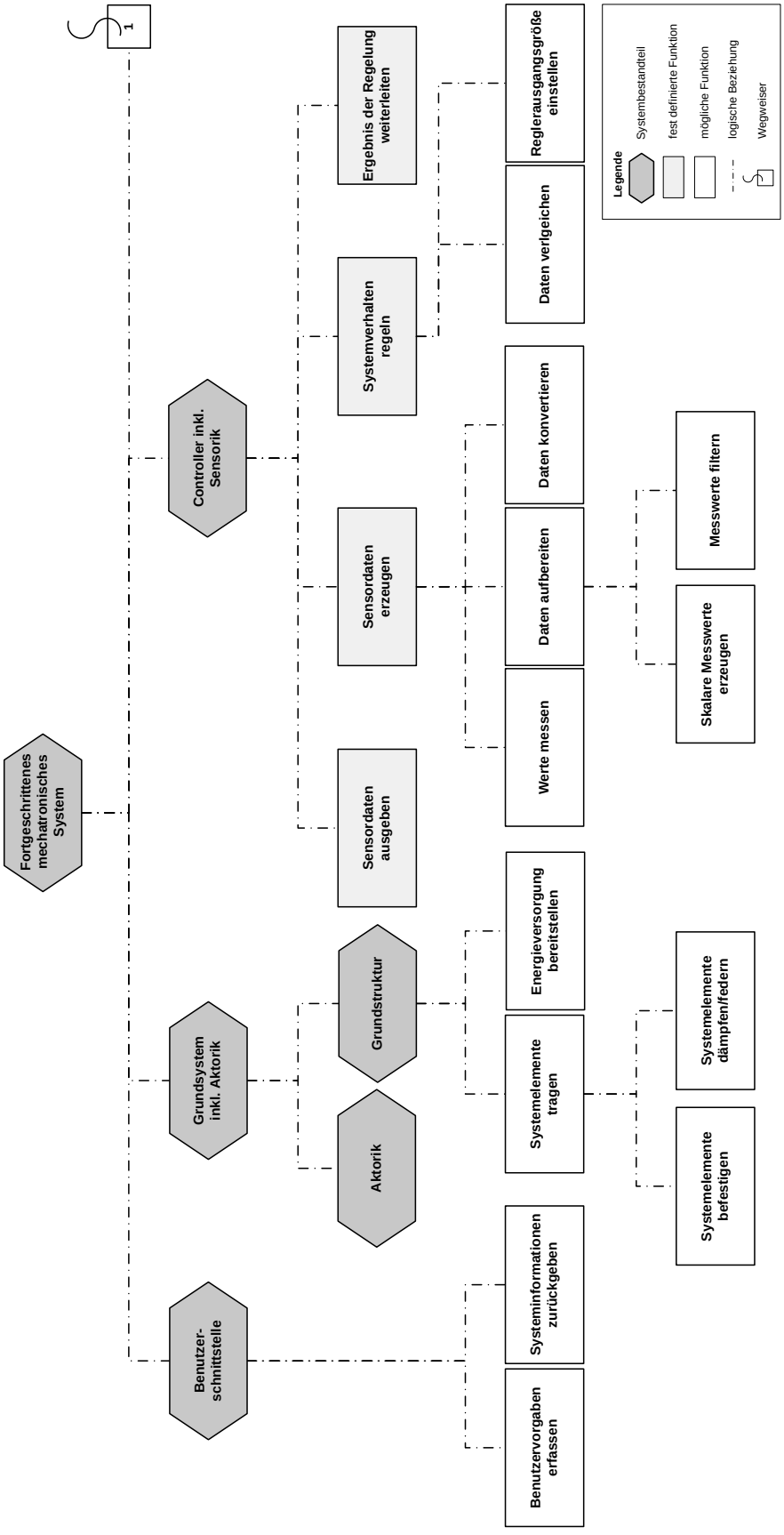


Bild 4-10: Entwurfsschablone für die Spezifikation der Funktionshierarchie eines fortgeschrittenen mechatronischen Systems (Teil 1 von 3)

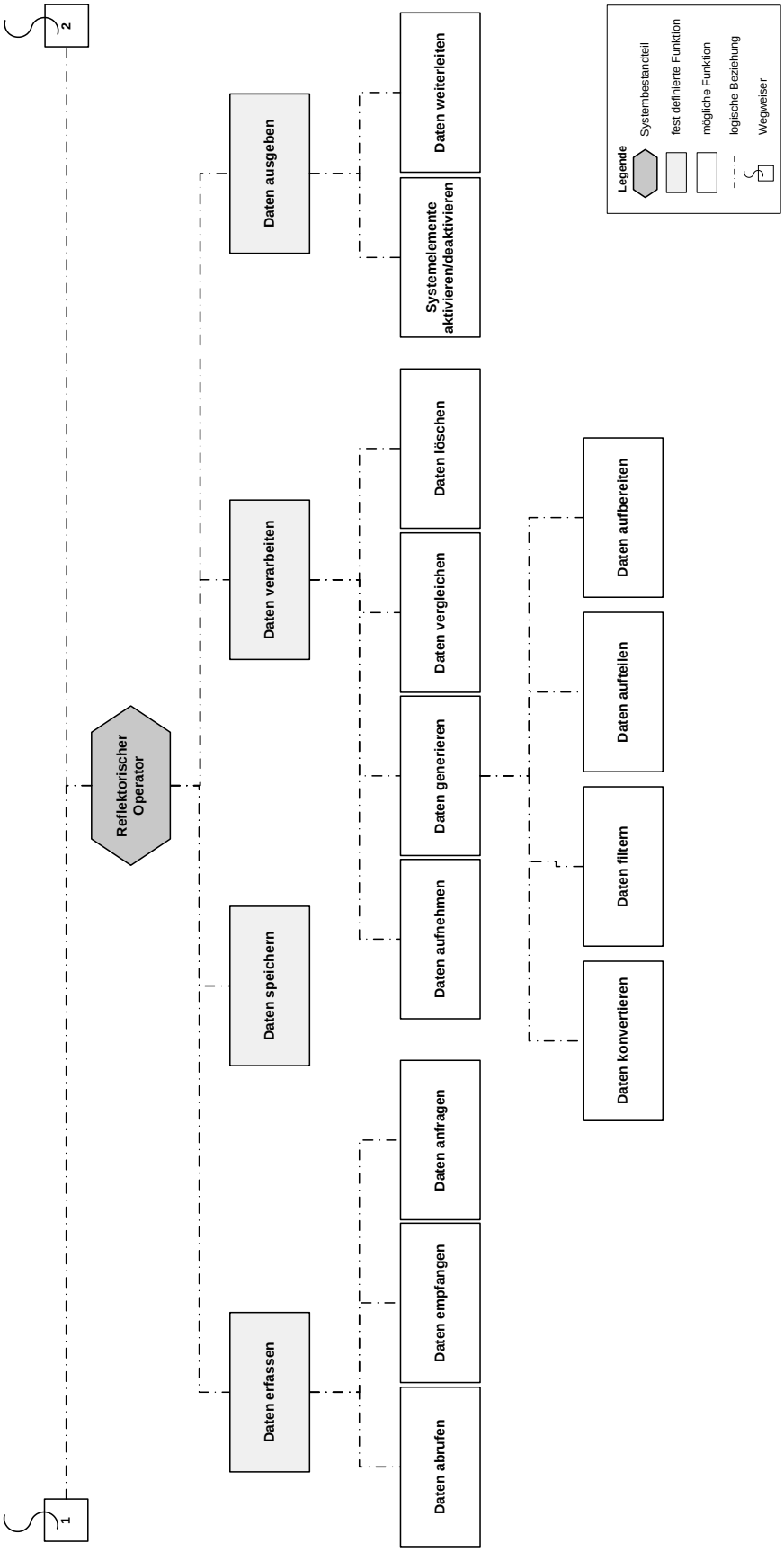


Bild 4-11: Entwurfsschablone für die Spezifikation der Funktionshierarchie eines fortgeschrittenen mechatronischen Systems (Teil 2 von 3)

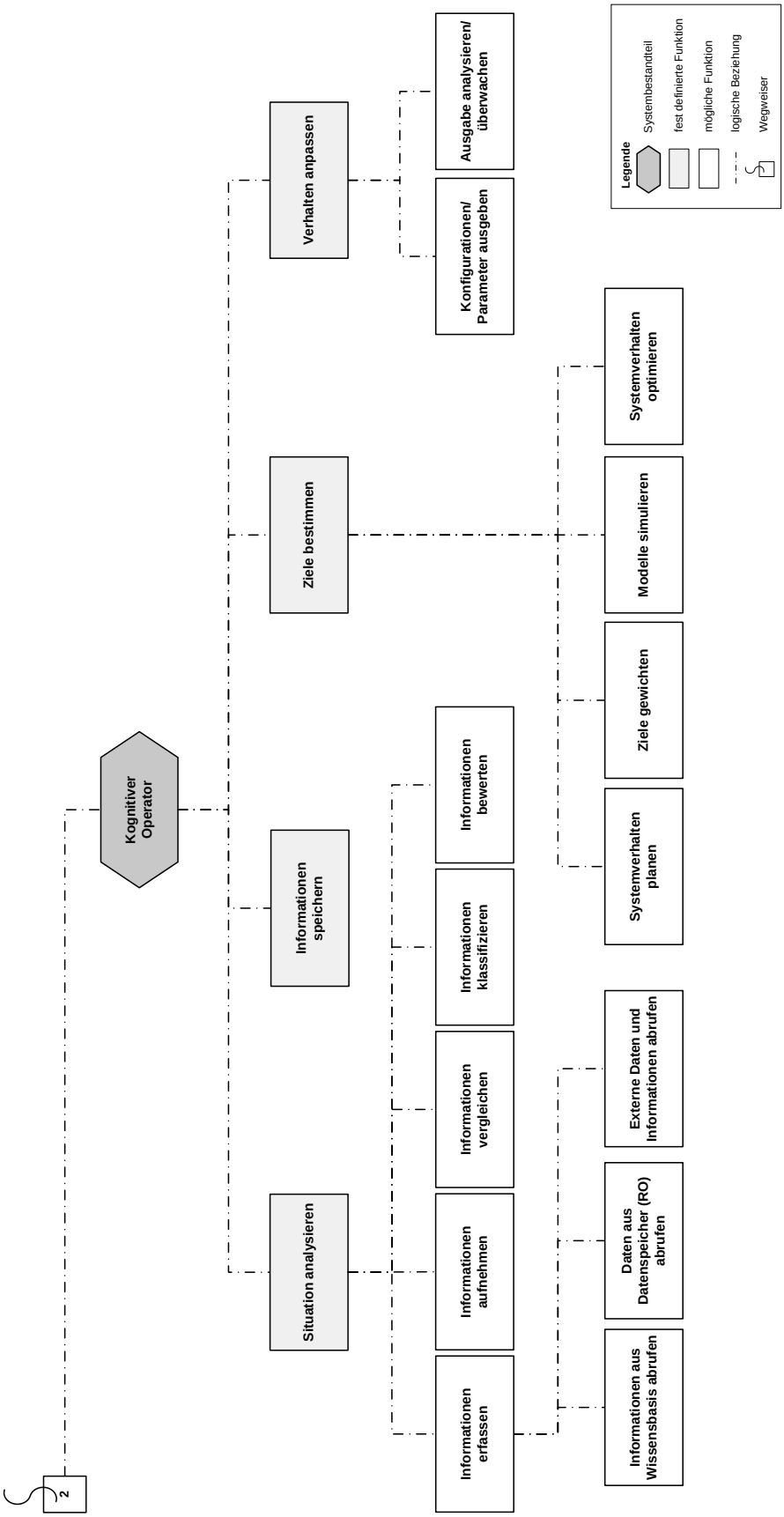


Bild 4-12: Entwurfsschablone für die Spezifikation der Funktionshierarchie eines fortgeschrittenen mechatronischen Systems (Teil 3 von 3)

4.5 Lösungswissen

Wie die vorhergehenden Kapitel gezeigt haben, ist für die Entwicklung der Informationsverarbeitung fortgeschrittener mechatronischer Systeme Wissen notwendig, das in der Regel nur einzelne Experten wie z.B. der Regelungstechnik, der Softwaretechnik oder der künstlichen Intelligenz besitzen. Ferner wird neues Wissen im Laufe einer erfolgreichen Entwicklung erschlossen. Bei beiden Wissensarten handelt es sich zunächst um personengebundenes Wissen, das nur stillschweigend verfügbar ist. Derartiges Wissen wird im Kontext des Systementwurfs als Lösungswissen bezeichnet.

In diesem Kapitel wird eine geeignete Technik für den Umgang mit Lösungswissen vorgestellt, die allen an der Entwicklung beteiligten Personen einen effektiven Zugang zu diesem Wissen ermöglicht. Ziel ist es, Lösungswissen wiederverwenden zu können und Dritten zur Verfügung zu stellen. Kern ist ein musterorientierter Ansatz zur Dokumentation und Präsentation von Lösungswissen für den Entwurf des OCM (Kap. 4.5.1). Dieser wird anhand ausgewählter Beispielmuster validiert (Kap. 4.5.2). Ferner wird eine Klassifikation kognitionsrelevanter Verfahren vorgestellt (Kap. 4.5.3).

4.5.1 Lösungsmuster für fortgeschrittene mechatronische Systeme

Um das Wissen der Mitarbeiter für eine gesamte Organisation nutzbar zu machen, entwickelten NONAKA/TAKEUCHI das sog. SECI-Modell (**S**ocialisation, **E**xternalization, **C**ombination, **I**nternalization) [NT97]. Dieses beschreibt aufbauend auf dem Begriff des impliziten Wissens nach POLANY den Ablauf der Wissenserzeugung in Unternehmen als Wissensspirale [Pol85]. Dabei wird implizites Wissen, das nur stillschweigend und nicht formalisiert vorhanden ist, in explizites, formalisiertes Wissen umgewandelt und umgekehrt. Diese Umwandlung gliedert sich in vier aufeinander folgende Prozesse, die kontinuierlich durchlaufen werden. Das Wissen wird dabei innerhalb eines Unternehmens von individuellem Wissen auf höhere Organisationsstufen wie Personengruppen oder die gesamte Organisation überführt (vgl. Bild 4-13).

Die **Sozialisation** beschreibt den Austausch von implizitem Wissen zwischen Individuen durch gemeinsame Erfahrungen, Beobachtungen oder Nachahmung. Der Wissensaustausch verläuft weitestgehend stillschweigend. Die Umwandlung von implizitem Wissen in explizites Wissen wird **Externalisierung** genannt. Diese verlangt nach Möglichkeiten, nicht trivial beschreibbare Zusammenhänge allgemeinverständlich wiederzugeben. So können alle beteiligten Individuen auf das gleiche explizite Wissen zugreifen. Das durch Externalisierung gewonnene Wissen wird im Rahmen der **Kombination** verteilt und erweitert. Dieser Prozess erfolgt z.B. durch direkte Kommunikation zwischen Individuen. Der Einsatz der Informationstechnologie nimmt hier eine besondere Stellung ein und kann zu einer Anreicherung des Wissens führen. Auf Basis des kombinierten expliziten Wissens kann eine **Internalisierung** erfolgen. Das Wissen wird vom Individuum aufgenommen, verstanden und gelernt [NT97, S. 84ff.].

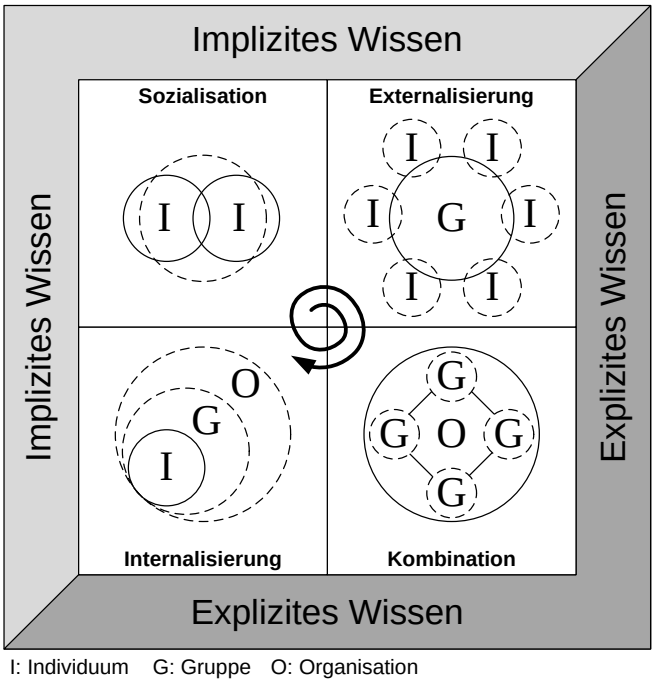


Bild 4-13: SECI-Modell nach NONAKA/TAKEUCHI [NT97, S. 84]

Nach dem SECI-Modell setzt eine erfolgreiche Externalisierung des Lösungswissens eine allgemeinverständliche Beschreibung voraus. Auf diese Weise ist es möglich, dass auch unerfahrene Ingenieure oder Experten anderer Disziplinen spezifische Sachverhalte verstehen können. Eine derart einheitliche Spezifikation des Lösungswissens ist gerade bei der Entwicklung einer kognitiven Informationsverarbeitung, die durch eine verstärkte Interdisziplinarität und oftmals exklusivem Expertenwissen geprägt ist, von entscheidender Bedeutung.

4.5.1.1 Einheitliche Spezifikation eines Lösungsmusters

Ein in der Produktentwicklung etabliertes Konstrukt zur Wissensexternalisierung ist das sog. Lösungsmuster (LM) (vgl. Kap. 2.4.3). Gemäß den Anforderungen muss sich dessen Struktur an der Definition von ALEXANDER orientieren und das zugrundeliegende Problem als auch dessen Lösung beschreiben (vgl. Kap. 2.6). Ferner ist der Einsatz eines Lösungsmusters an einen bestimmten Kontext gebunden, der ebenfalls dokumentiert werden muss. Bestehende Ansätze aus dem Stand der Technik werden diesen Anforderungen nur teilweise gerecht. Da auch keines der untersuchten Lösungsmuster den frühzeitigen Entwurf der Informationsverarbeitung unterstützt, wurde eine neue Spezifikation für Lösungsmuster für den Entwurf fortgeschrittener mechatronischer Systeme erarbeitet. Die Spezifikation eignet sich zur Aufnahme von Lösungen sowohl für physikalische als auch informationsverarbeitende Systembestandteile. Letztere stehen aber vor dem Hintergrund der Entwicklung fortgeschrittener Systeme mit kognitiven Funktionen im Vordergrund. Bild 4-14 gibt einen Überblick über die **einheitliche Spezifikation eines Lösungsmusters** für fortgeschrittene mechatronische Systeme.

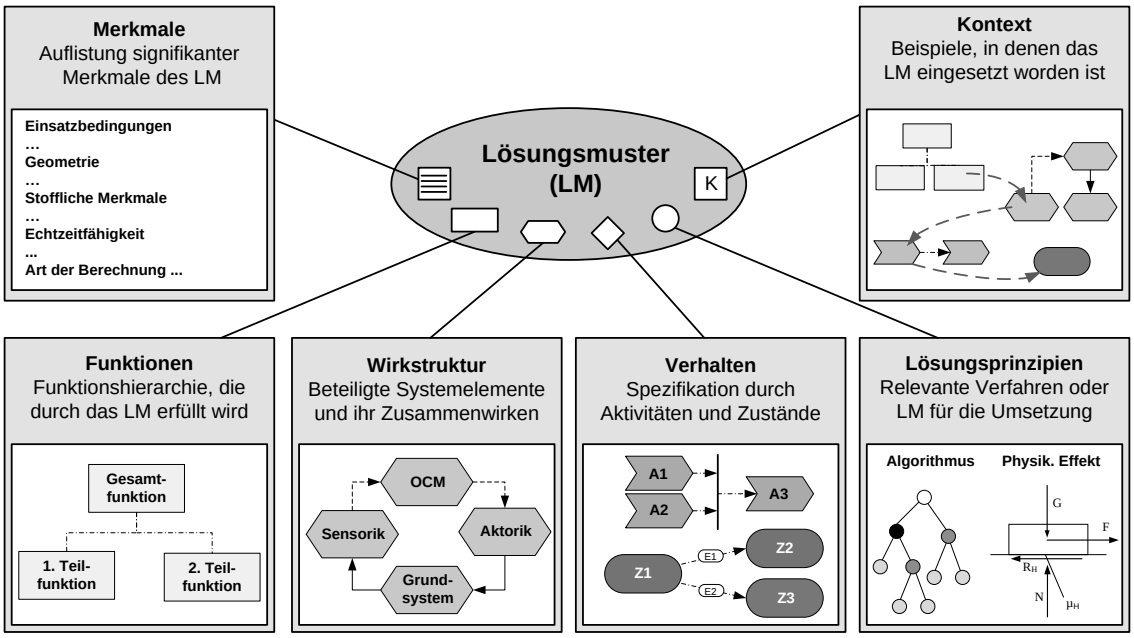


Bild 4-14: Einheitliche Spezifikation eines Lösungsmusters im Überblick

Da die Inhalte eines Lösungsmusters in derselben Sprache spezifiziert werden sollten, die auch im Systementwurf eingesetzt wird, basiert die einheitliche Spezifikation der Lösungsmuster auf der in Kap. 4.4 konkretisierten Form der Spezifikationstechnik des SFB 614[54]. Hierzu werden sechs Aspekte unterschieden, die eine disziplinunabhängige Beschreibung ermöglichen und im Weiteren erläutert werden.

Merkmale

In diesem Aspekt werden sämtliche Eigenschaften spezifiziert, die für das jeweilige Muster charakteristisch sind. Die Merkmale erlauben Rückschlüsse auf die Anforderungen, die das Lösungsmuster erfüllen kann. Dabei variieren je nach Lösungsmuster nicht nur die Ausprägungen, sondern auch die Merkmale selbst. Die Merkmale werden zum besseren Abgleich mit den Systemanforderungen im Normalfall in einer Liste dokumentiert. Grundsätzlich empfiehlt es sich, Merkmale für die Informationsverarbeitung und Merkmale für das (physikalische) Grundsystem zu unterscheiden. Die folgenden Merkmale bieten erste Anhaltspunkte und haben keinen Anspruch auf Vollständigkeit. Die Liste wird mit der Aufnahme von Lösungsmustern stets erweitert. *Merkmale für Lösungsmuster für das Grundsystem* können in Übereinstimmung mit den Hauptmerkmallisten zur Anforderungsbestimmung technischer Systeme nach PAHL/BEITZ beschrieben werden [PBF+07, S. 220]:

[54] Grundsätzlich ist die Spezifikation aber auch mit anderen Sprachen wie z.B. der SysML umsetzbar. Diese unterstützen aber nicht die im Fokus der Arbeit stehende vollständige Beschreibung der Informationsverarbeitung inkl. kognitiver Funktionen.

- *Einsatzbedingungen:* Unter diesem Gesichtspunkt ist festzuhalten, unter welchen Umgebungsbedingungen physikalische Systemelemente eingesetzt werden können.
- *Geometrie:* Hierunter fallen grobe Angaben zu den Abmaßen. Mit ihnen lässt sich u.a. frühzeitig der benötigte Raumbedarf ermitteln.
- *Stoffliche Merkmale:* Im Hinblick auf die verträgliche Kombination mehrerer Lösungsmuster sind diese von hoher Bedeutung. Es ist nicht nur festzuhalten, aus welchen Materialien die beteiligten Elemente bestehen, sondern auch, welche Eigenschaften das zusätzlich benötigte Arbeitsmedium hat.
- *Kinematik:* Grundsätzlich ist zwischen rotatorischer und translatorischer Bewegung zu unterscheiden. Hinzu kommen Angaben, in welche Richtung oder um welche Koordinatenachse die Bewegung stattfindet.
- *Montage:* Es ist denkbar, dass die Wirkungsweise von Lösungsmustern mit Besonderheiten bei der Montage einhergehen. In diesem Fall sollte Art und Weise der Montage dokumentiert werden.
- *Normung:* Sollte das Lösungsmuster genormte Lösungselemente beinhalten, sind diese explizit aufzuführen.

Es sind nur wenige Ansätze zur Beschreibung charakteristischer Merkmale für Lösungen der Informationsverarbeitung vorhanden. Grundsätzlich hängen diese stark von den einzusetzenden informationsverarbeitenden Verfahren ab (vgl. Kap. 4.5.3). Erste Ansätze auf übergeordneter Ebene liefern FRANK et al. und BALZERT. In Anlehnung an diese konnten folgende *Merkmale für Lösungsmuster für die Informationsverarbeitung* identifiziert werden [Bal05], [FGK+04, S. 29ff.]:

- *Einsatzbedingungen:* Der Einsatz der Lösungsmuster ist an bestimmte Bedingungen und Systemeigenschaften gekoppelt. So ist die Realisierung einer Umschaltstrategie nur möglich, wenn sich im Controller mehr als eine Regelkonfiguration befindet.
- *Zeitpunkt der Aktivität*: Die Systemelemente des Lösungsmusters können kontinuierlich oder nur in festgelegten Zeitpunkten aktiv sein. Sie können auch nur aktiv werden, wenn das System einen vordefinierten Zustand erlangt oder ein bestimmtes Ereignis stattfindet.
- *Echtzeitfähigkeit:* Es sollte festgelegt werden, welche zeitlichen Anforderungen das Lösungsmuster erfüllt. Es kann zwischen harter Echtzeit, weicher Echtzeit oder einem nicht zeitkritischen Prozess unterschieden werden. Hieraus ergeben sich wiederum Anforderungen an die einzusetzende Hardware.
- *Modellierbarkeit:* Es ist festzuhalten, ob das Lösungsmuster nur für Systeme bzw. Systemteile eingesetzt werden kann, deren Verhalten vollständig physikalisch modelliert werden kann.

- *Art der Berechnung:* Dieses Merkmal ist besonders durch die zugrundeliegenden Verfahren geprägt. Grundsätzlich ist zu klären, ob die eingehenden Daten für die Berechnungen kontinuierlich oder diskret erfasst werden. Ferner unterscheiden sich die Berechnungen z.B. bezüglich der Linearität, Determiniertheit oder Finitheit.
- *Entität:* Die Systemelemente des Lösungsmusters können isoliert oder kollektiv in Kooperation mit anderen Komponenten die Informationen und Daten verarbeiten. So können bspw. bei Agentensystemen einzelne Rollen definiert werden.
- *Signalart:* Einer Informationsverarbeitung liegt immer ein Signal zugrunde, das die Informationen transportiert (vgl. Kap. 4.1). So kann zwischen analoger, binärer oder digitaler Signalverarbeitung differenziert werden, die sich aus der physikalischen Eigenschaft des Signals ergeben.
- *Modellierungs-/Programmiersprache:* Die Sprache, in der die Lösung programmiert wurde, ist zu dokumentieren. Es kann aber auch eine andere oder weitere Programmiersprache empfohlen werden. Grundsätzlich gibt es Sprachen für eine hardwarenahe Programmierung (Maschinen- oder Assemblersprachen) und komplexere Sprachen, die vorab in eine maschinenlesbare Form kompiliert werden.
- *Datenstruktur*: Ist das Lösungsmuster bereits sehr detailliert spezifiziert, können hinsichtlich der späteren Implementierung Typen oder die Datenstruktur definiert werden. Der Ressourcenbedarf eines Algorithmus ist von der Verwendung der Datenstruktur abhängig.

Funktionen

Durch diesen Aspekt werden die erfüllbaren Funktionen und somit die Aufgabe des Lösungsmusters festgelegt. Im einfachsten Fall kann das eine Funktion sein, wie dies bei Wirkprinzipien der Konstruktionslehre der Fall ist (vgl. Kap. 3.3.1.1). Komplexere Lösungen umfassen mehrere Funktionen, die in einer Funktionshierarchie beschrieben werden. Derart können auch logische funktionale Zusammenhänge dargestellt werden. Nach den drei prinzipiellen Flussarten mechatronischer Systeme können stoffbestimmte (z.B. Stoff transportieren), energiebestimmte (z.B. Energie umwandeln) und informationsbestimmte Funktionen (z.B. Informationen erfassen) unterschieden werden (vgl. Kap. 2.2.1). Für den Anwender der Lösungsmuster, der im Zuge des Systementwurfs Lösungsmuster für Funktionen sucht, repräsentiert dieser Aspekt in Anlehnung an die Definition nach ALEXANDER das zu lösende Problem. Für die Dokumentation dieses Aspekts können die entsprechenden Funktionen aus der bestehenden Funktionsbeschreibung des Systems entnommen werden. Eine Orientierung an der Entwurfsschablone zur Erstellung einer Funktionshierarchie für fortgeschrittene mechatronische Systeme wird empfohlen (vgl. Kap. 4.4.3). Sie bietet einen guten Überblick über die grundsätzlichen Problemstellungen, die sich bei der Konzipierung derartiger Systeme ergeben. Ferner wird eine Einordnung in die Gesamtproblematik unterstützt, wodurch auch die Suche nach einem geeigneten Lösungsmuster über die Funktionen verbessert wird.

Wirkstruktur

Dieser Aspekt bildet den Kern der Lösungsbeschreibung. Er umfasst alle Systemelemente, die die im Lösungsmuster definierten Funktionen erfüllen. Zudem werden deren Interaktionen spezifiziert, wodurch die prinzipielle Wirkungsweise des Lösungsmusters beschrieben wird. Auch hier werden die drei Flussarten mechatronischer Systeme unterschieden. Insbesondere im Hinblick auf die Erstellung von Lösungsmustern für die Informationsverarbeitung sollte die allgemeine Wirkstruktur fortgeschrittener mechatronischer Systeme im Sinne einer Entwurfsschablone genutzt werden (vgl. Kap. 4.4.2). So können bei der Erstellung die entsprechenden Schnittstellen zu anderen Systemelementen definiert werden, ohne diese im Detail mit auszugestalten. Es empfiehlt sich, diese Systemelemente, die nicht direkt zur prinzipiellen Lösung beitragen, als Umfeldelemente zu kennzeichnen.

Verhalten

Die Spezifikation des Verhaltens komplettiert die Beschreibung der Lösung aus der Wirkstruktur. Sie ist insbesondere für die Spezifikation der informationsverarbeitenden Ebenen des OCM relevant. Aber auch aktorische Abläufe sollten beschrieben werden. Für statische, passive Lösungen, wie z.B. einen Lagersitz, ist eine Verhaltensabbildung nicht zwingend erforderlich. In der Regel werden zwei Verhaltensarten unterschieden:

- Der Aspekt *Verhalten – Aktivitäten* ist notwendig, um Ablaufdiagramme zu erstellen. Sie spezifizieren in welcher Reihenfolge ein Systemelement seine Funktionen ausführt. Zudem wird das Zusammenspiel aller beteiligten Systemelemente dargestellt. Das ist insbesondere bei der Lösungsbeschreibung von Softwareproblemen relevant. Daher ist dieser Aspekt für alle Lösungsmuster auszuarbeiten. Eine Ausnahme bilden passive Systemelemente des Grundsystems, da sie keine eigenständigen Aktivitäten aufweisen. Für fortgeschrittene mechatronische Systeme, die einen Selbstoptimierungsprozess ausführen, empfiehlt es sich die entsprechenden Aktivitäten den drei Phasen Situationsanalyse, Zielbestimmung und Verhaltensanpassung zuzuordnen.
- Nach der Ausführung von Aktivitäten ändert sich in der Regel der Zustand des Systems, bzw. der Systemelemente. Auch passive Elemente können unterschiedliche Zustände einnehmen. Diese Zustände und Zustandsübergänge werden im Aspekt *Verhalten – Zustände* dokumentiert. Insbesondere im Hinblick auf eine mögliche Systemsimulation sind elementare Zustände wie *defekt* und *nicht defekt* aufzugreifen. Daher kann das Verhaltensmodell für jedes Lösungsmuster aller beteiligten Fachdisziplinen erstellt werden.

Für sehr konkrete Lösungen können anstelle der grundsätzlich lösungsneutralen Verhaltensbeschreibungen auch *Simulationsmodelle* treten. Diese können derart aufbereitet sein, dass der Entwickler nur noch Eingangsparameter für ein gewünschtes Verhalten sucht.

Lösungsprinzipien

Die in der Wirkstruktur und im Verhalten spezifizierte Lösung und deren Wirkungsweise beruht auf bestimmten Lösungsprinzipien. Diese bilden die Basis für die Umsetzung der beschriebenen Lösung und sind daher von besonderer Bedeutung für die weitere Konkretisierung der Lösung in den späteren Phasen der Entwicklung. Grundsätzlich beruhen alle Lösungsprinzipien auf *Verfahren*. In Anlehnung an SAUER[55] beschreibt ein Verfahren die Abfolge von physikalisch-technischen, chemischen, biologischen oder informationstechnischen Wirkungsabläufen, die zur Realisierung einer gewünschten Funktion notwendig sind. Es konkretisiert eine abstrakt formulierte Funktion, die die Transformation eines Operanden von einem Ausgangszustand in einen Endzustand beschreibt. Es existieren zwei Verfahrensarten im Kontext des Systementwurfs fortgeschrittener mechatronischer Systeme:

- *Physikalische Effekte:* Lösungsprinzipien des physikalischen Grundsystems lassen sich auf physikalische Effekte zurückführen. Diese können wiederum in thermische, mechanische, elektrische, magnetische und optische Effekte unterteilt werden. Eine Vielzahl von physikalischen Effekten ergibt sich aus der Kombination der unterschiedlichen physikalischen Prinzipien (z.B. elektrisch-mechanisch: Piezoeffekt, thermisch-elektrisch: Seebeck-Effekt, magnetisch-elektrisch: Hall-Effekt oder optisch-elektrisch: Fotoelektrischer Effekt) [Czi08, S. 43]. Aufgrund der Vielzahl von vorhandenen Effekten sind sog. Effektkataloge bei der Beschreibung zu verwenden [Kol98], [Rod91], [AH07]. Diese stellen eine vergleichbare und wiedererkennbare Spezifikation sicher. Wichtige Bestandteile der Beschreibung sind die Bezeichnung, eine kurze textuelle Beschreibung, mathematische Formeln der zugrundeliegenden Gesetzmäßigkeit, das physikalische Prinzip sowie eine prägnante Skizze des Effekts.

- *Informationsverarbeitende Verfahren:* Sie sind Kern der Lösungsmuster für die Informationsverarbeitung und beschreiben den Ablauf aufeinander aufbauender Befehle. Jedes (informationsverarbeitende) Verfahren basiert auf Algorithmen, die immer auf eine bestimmte Datenstruktur zugreifen. Algorithmen liefern eine formale Handlungsvorschrift zur Lösung eines Problems und können zudem graphisch dargestellt werden [Bal05, S. 471f.]. Datenstrukturen stellen ein mathematisches Objekt zur Speicherung von Daten dar. Die unterschiedliche Strukturierung der Daten führt zu verschiedenen Datenstrukturen, wie z.B. Feld (array), Zeichenkette (string) oder Stapel (stack) [Bal05, S. 427], [Dud01, S. 136]. Regelalgorithmen beinhalten mathematische Zusammenhänge, wie z.B. die Aneinanderreihung von Übertragungsfunktionen. Softwarealgorithmen basieren auf einer Befehlsfolge, die sich in einem speziell ausgearbeiteten Softwarecode widerspiegelt. Optimierungs-

[55] SAUER analysierte und konkretisierte den Begriff Verfahren im Rahmen seiner Dissertation. Seine Definition bezieht sich allerdings auf Stoffänderungen im Sinne der Verfahrenstechnik [Sau06, S. 74].

verfahren liegen mathematische Gesetzmäßigkeiten sowie spezifische Softwarealgorithmen zu Grunde. Die Verfahrensdarstellung unterscheidet sich je nach Verfahren stark, sollte aber stets eine Kurzbeschreibung, eine Verfahrensskizze und einen Verweis auf weitere Informationen umfassen. Kognitionsrelevante Verfahren werden in Kapitel 4.5.3 näher erläutert.

Neben Verfahren können Lösungsmuster auf einer abstrakteren Ebene auch *Technologien* definieren. Derartige Lösungsmuster eignen sich in erster Linie für eine strategische Entscheidung im Rahmen der Produktfindung und unterstützen bspw. das Erstellen von Technology-Roadmaps. Technologien sollten mit einem Technologiesteckbrief beschrieben werden [GBI09, S. 42f.]. Ferner kann ein *Lösungsmuster* wiederum selbst ein Lösungsprinzip eines anderen Lösungsmusters sein. Auf diese Weise kann ein übergeordnetes Lösungsmuster aus untergeordneten Lösungsmustern zusammengestellt werden (vgl. Bild 3-18). Die Beschreibung der untergeordneten Lösungsmuster entspricht wieder der einheitlichen Spezifikation eines Lösungsmusters.

Lösungsprinzipien werden in der Regel nur aufgelistet. Deren Auswahl und mögliche Kombination wird erst im weiteren Verlauf der Konkretisierung endgültig geklärt, da eine Teillösung nicht isoliert von anderen Entwurfsentscheidungen getroffen werden sollte. Zudem ergibt sich so ein erhöhtes Innovationspotential, besonders durch neue Kombinationen von Lösungsprinzipien. Sollten schon genaue Informationen über die bestmögliche Umsetzung vorliegen, kann der Wirkzusammenhang durch Kombination der Lösungsprinzipien dargestellt werden. Hier geht es in erster Linie um eine sequentielle Verkettung und um eine Identifikation von Abhängigkeiten bzw. Widersprüchen bei der Verwendung unterschiedlicher Lösungsprinzipien. In beiden Fällen ist eine detaillierte Beschreibung der einzelnen Lösungsprinzipien obligatorisch.

Kontext

Der letzte Aspekt der Lösungsmusterspezifikation gibt den Kontext der Umsetzung wieder und beschreibt, wie das Lösungsmuster in eine Systemspezifikation einzubinden ist. Ziel ist es, funktionale und bauliche Zusammenhänge aus systemischer Sicht abzubilden. Dies erfolgt durch die Dokumentation verschiedener Beispiele, in denen das Lösungsmuster bereits erfolgreich eingesetzt wurde. Jede vollständige Spezifikation eines Lösungsmusters muss über mindestens ein Beispiel verfügen. Neben einer Bezeichnung sollte das Beispiel kurz textuell beschrieben werden. Eine aussagekräftige Skizze zur Ergänzung der Kontextbeschreibung ist empfehlenswert. Da der Kern eines Lösungsmusters stets die Problembeschreibung sowie die dazugehörige Lösung umfasst, sind für jedes Beispiel die Aspekte *Funktionen*, *Wirkstruktur* und *Verhalten* zu dokumentieren. In ihnen sollten sowohl das verwendete als auch andere bekannte Lösungsmuster gekennzeichnet sein, die bei der Entwicklung des Beispielsystems verwendet wurden. Hierfür wird das Konstrukt *Mustergruppe* eingeführt, das die entsprechenden Bestandteile eines Lösungsmusters innerhalb eines Aspekts des Beispiels zusammenfasst. Ferner spielen die Querverweise zwischen den Aspekten eine entscheidende

Rolle, da diese wichtige Hinweise für eine erneute Verwendung des Lösungsmusters geben. Prinzipiell ist es möglich, dass ein Beispiel wiederum selbst als ein übergeordnetes Lösungsmuster dokumentiert werden kann. Die Kontextbeschreibung ist dann um die weiteren Aspekte zu ergänzen.

4.5.1.2 Klassifikation und Einsatz der Lösungsmuster

Die erarbeitete einheitliche Spezifikation eines Lösungsmusters stellt eine Anwendung in allen, an der Entwicklung fortgeschrittener mechatronischer Systeme beteiligten Fachdisziplinen sicher. Da sich die Spezifikation zur Beschreibung von Teillösungen des Grundsystems und der Informationsverarbeitung eignet, kann die Einteilung der Lösungsmuster nach GAUSEMEIER et al. (vgl. Kap. 2.4.3) hinsichtlich der Struktur des OCM angepasst werden (Bild 4-15).

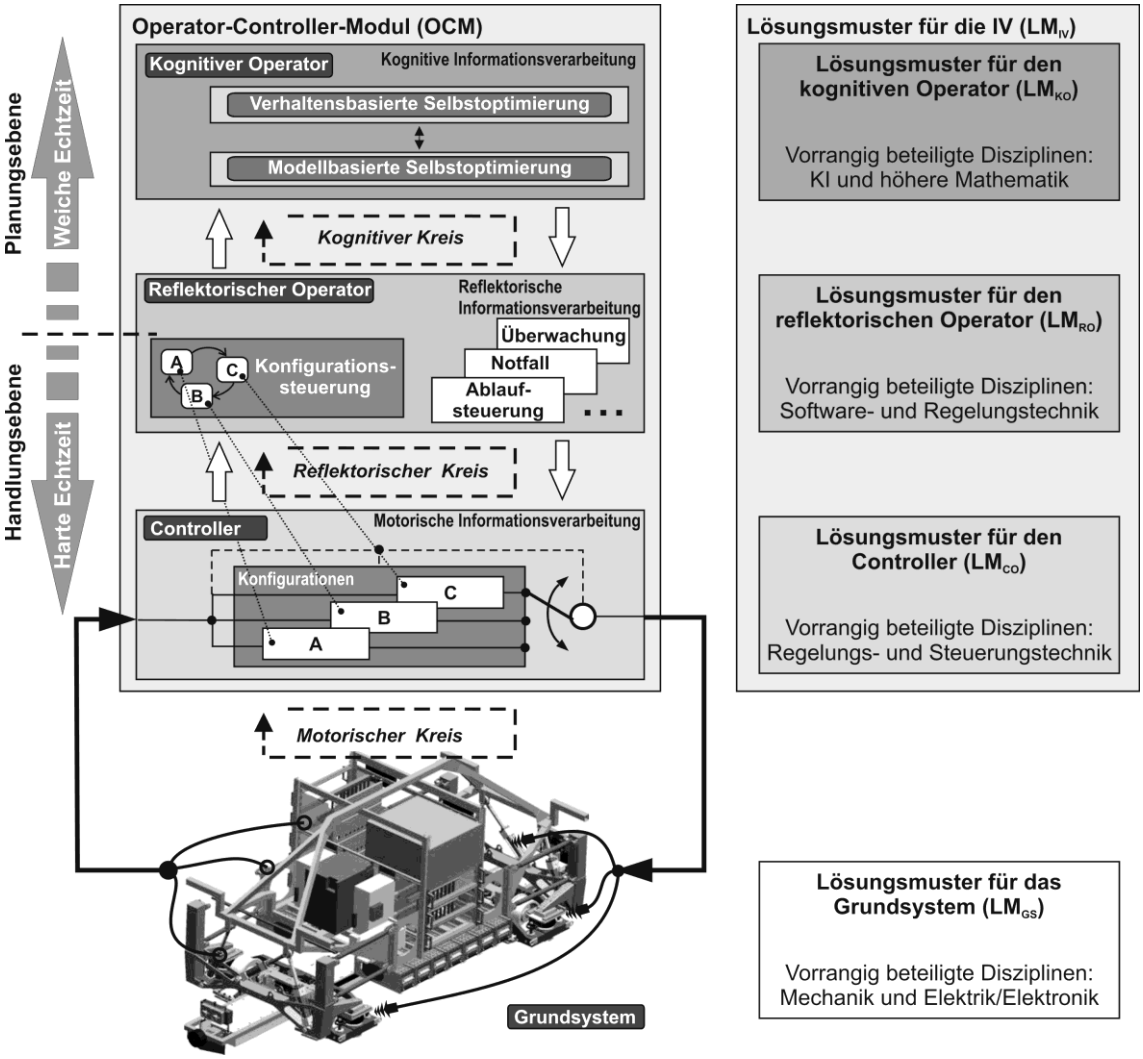


Bild 4-15: Klassifikation der Lösungsmuster für fortgeschrittene mechatronische Systeme anhand der OCM-Struktur

Die Kategorisierung unterscheidet Lösungsmuster für die Informationsverarbeitung (LM_{IV}) und Lösungsmuster für das Grundsystem (LM_{GS}). Die LM_{IV} unterteilen sich in Lösungsmuster für den Controller (LM_{CO}), Lösungsmuster für den reflektorischen Operator (LM_{RO}) und Lösungsmuster für den kognitiven Operator (LM_{KO}) entsprechend den drei Ebenen des OCM. Die LM_{GS} können noch weiter in Lösungsmuster für die Sensorik, Aktorik oder der Grundstruktur klassifiziert werden, worauf im Rahmen dieser Arbeit aber nicht näher eingegangen wird. Von besonderer Bedeutung für die Integration kognitiver Funktionen sind die LM_{KO}. In diesen Lösungsmustern können die wissensverarbeitenden Prozesse, wie z.B. der Selbstoptimierungsprozess im Sinne des SFB 614, und entsprechende Verfahren zur Realisierung aufgenommen werden. Ferner können übergeordnete Lösungsmuster aus einer Kombination der unterschiedlichen Lösungsmuster entstehen. Derartige Lösungsmuster können die gesamte Struktur eines mechatronischen Systems abbilden. Speziell bei Anpassungs- oder Variantenentwicklungen würden solche ganzheitlichen mechatronischen Lösungsmuster hohen Nutzen stiften.

Aufgrund der disziplinübergreifenden und einheitlichen Spezifikation der Lösungsmuster können diese zunächst ohne zusätzliches disziplinspezifisches Wissen eingesetzt werden. Bild 4-16 gibt am Beispiel der Entwicklung des Miniaturroboters BeBot einen generischen und stark vereinfachten Überblick, wie die Lösungsmuster für fortgeschrittene mechatronische Systeme im Verlauf des Systementwurfs genutzt werden und so wesentlich zur Systemkonkretisierung beitragen.

Der prinzipielle Einsatz der Lösungsmuster orientiert sich an dem Vorgehen im Systementwurf (vgl. Kap. 2.4.3) und ordnet sich in das Vorgehensmodell der Entwicklungssystematik ein (vgl. Kap. 4.3.1). Ausgehend von der Funktionshierarchie des zu entwickelnden Miniaturroboters werden für einzelne oder gleich mehrere Teilfunktionen geeignete Lösungsmuster gesucht. Die Teilfunktionen können Aufgaben verschiedener Disziplinen umfassen. So erfüllt das LM_{GS} „Elektr. Antrieb“ die elektro-mechanische Funktion „BeBot antreiben” und das LM_{KO} „Probabilistische Planung“ die komplexe informationsverarbeitende (kognitive) Funktion „Energieverbrauch optimieren“. Anschließend müssen die Aspekte der identifizierten Lösungsmuster in die bestehende Systemspezifikation integriert werden. So werden die Wirkstruktur und die beiden Verhaltensmodelle durch die entsprechenden Aspekte der Lösungsmuster aufgestellt. Zudem wird geprüft, ob Funktionen, die in den Lösungsmustern zusätzlich dokumentiert sind, die bestehende Funktionshierarchie ergänzen sollten. Auf diese Weise unterstützen die Aspekte der Lösungsmuster die frühe Systemspezifikation in Form der Prinziplösung. In der weiteren Konkretisierung kann diese dann in disziplinspezifische Sichten, wie die Baustruktur, Komponentenstruktur und State-Chart, überführt werden.

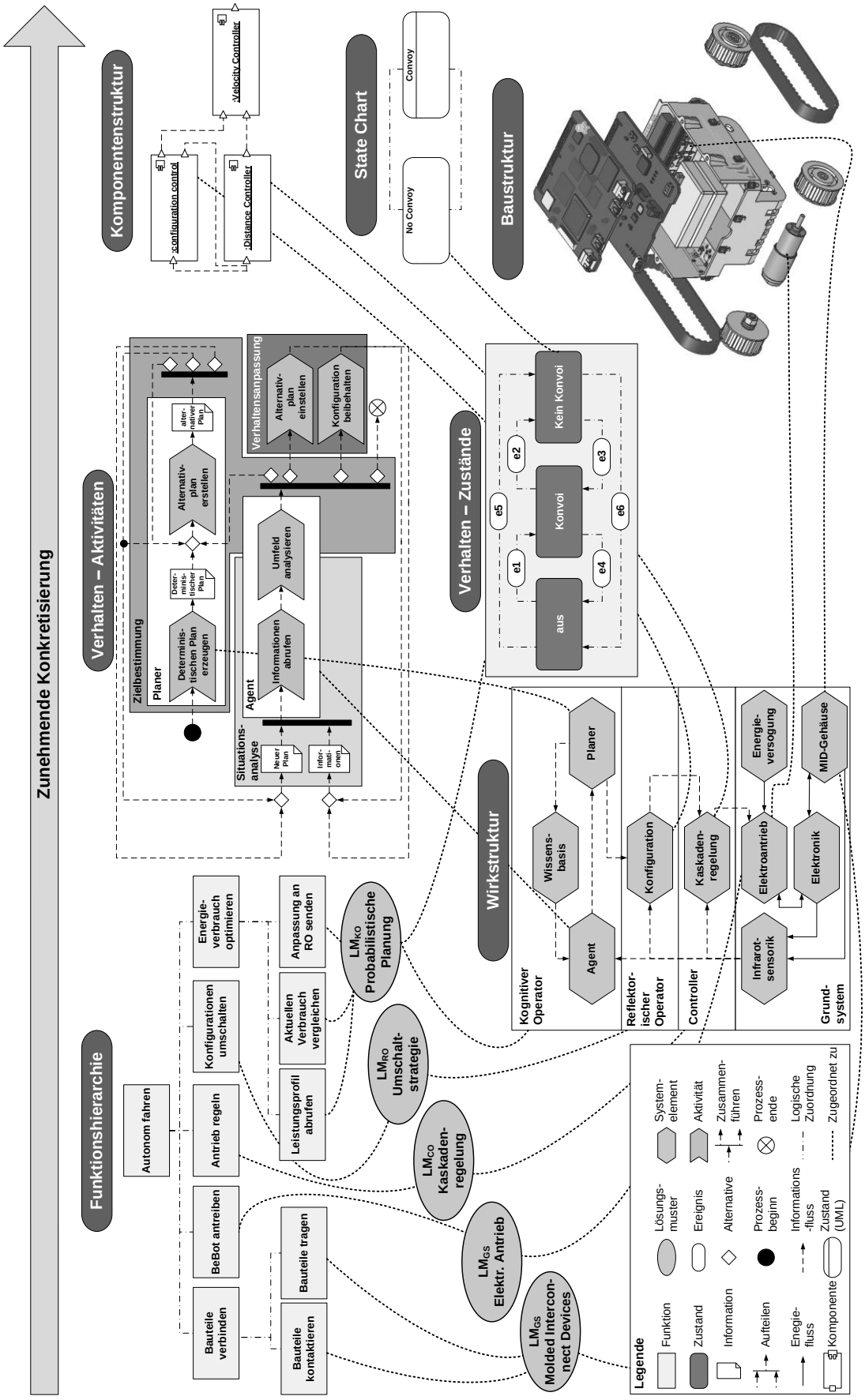


Bild 4-16: Einsatz der Lösungsmuster im Systementwurf (Aspekte stark vereinfacht)

4.5.2 Ausgewählte Lösungsmuster für den OCM-Entwurf

In den nächsten Abschnitten werden konkrete Lösungsmuster aus Bild 4-16 dargestellt. Da im Fokus der Arbeit die frühzeitige Beschreibung der Informationsverarbeitung fortgeschrittener mechatronischer Systeme steht, werden nur Lösungsmuster für die drei Ebenen des OCM vorgestellt (vgl. Bild 4-15). Ein Lösungsmuster für das Grundsystem findet sich im Anhang (vgl. A2.1). Es belegt die einheitliche und disziplinübergreifende Verwendung der erarbeiteten Lösungsmusterspezifikation.

4.5.2.1 LM_{KO} „Probabilistische Planung“

Hinsichtlich fortgeschrittener mechatronischer Systeme ist die Umsetzung des kognitiven Operators von entscheidender Bedeutung. Für dessen Entwurf entwickelte Lösungsmuster adressieren insbesondere die kognitive Informationsverarbeitung sowie entsprechende Funktionen wie das „Systemverhalten eigenständig optimieren“. Im Weiteren wird das LM_{KO} „Probabilistische Planung“ vorgestellt. Es wurde im Rahmen des SFB 614 erarbeitet und basiert im Wesentlichen auf den Arbeiten zur Planung unter Unsicherheit für mechatronische Systeme von KLÖPPER [Klö09, S. 60ff.]. Bild 4-17 zeigt das Lösungsmuster im Überblick.

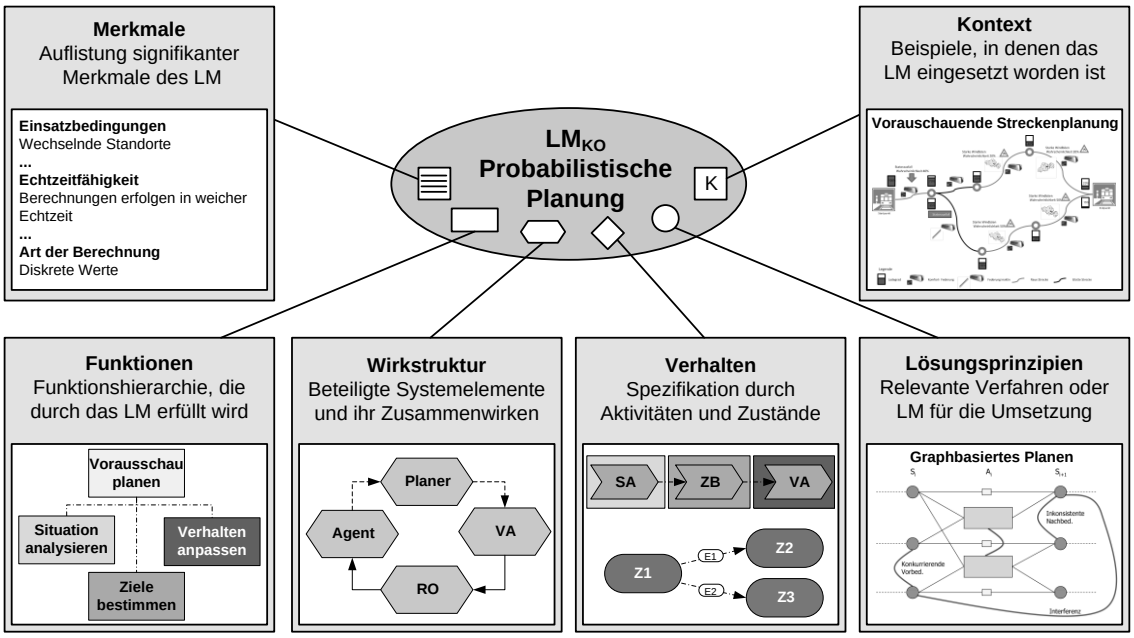


Bild 4-17: Lösungsmuster für den kognitiven Operator – Probabilistische Planung

Gegenstand des Lösungsmusters ist das Vorausplanen möglicher Zukunftssituationen unter der Berücksichtigung von Unsicherheiten. Während der Planung wird ein Planbaum aufgestellt, in dem verschiedene Pfade zum selben Ziel führen. Es werden ein idealer Pfad und mehrere mögliche Verzweigungen erstellt. Die Verzweigungen besitzen Bedingungen, die angeben, wann eine Verzweigung ausgeführt wird. Eine Bedingung ist dabei eine Menge von Grenzwerten über eine oder mehrere Zustandsvariablen

(bedingte Planung). Die probabilistische Planung beinhaltet die zuvor beschriebene bedingte Planung und die Ausführungsüberwachung. Ferner kann eine Neuplanung berücksichtigt werden. Die Grundidee besteht darin, die bedingte Planung als Standardplanungsverfahren einzusetzen und auf die Neuplanung nur fallweise zurückzugreifen. Die Neuplanung stellt demnach eine Rückfallebene für Zustände dar, die aus unvorhersehbaren bzw. unwahrscheinlichen Ereignissen resultieren. Tritt ein solches Ereignis ein, ist der bereits erstellte bedingte Plan durch weitere Verzweigungen zu ergänzen. Je nach Situation, in der die Neuplanung erforderlich wird, variieren die Anforderungen hinsichtlich der Planungsgeschwindigkeit. Ist in sehr kurzer Zeit eine Alternative zu bestimmen, kann sich dies zu Lasten der Planungsqualität auswirken. Daher ist die Anwendung der Rückfallebene soweit wie möglich zu vermeiden.

Die charakteristischen **Merkmale** des LM_{KO} „Probabilistische Planung“ sind:

- *Einsatzbedingungen:* In erster Linie eignet sich die probabilistische Planung für mechatronische Systeme mit wechselnden Standorten und der sich daraus ergebenden Unsicherheit, wie z.B. Fahrzeuge oder mobile Roboter. Bei stationären Systemen lassen sich unvorhersehbare Ereignisse weitestgehend ausschließen.
- *Art der Berechnung:* Die Werte werden diskret verarbeitet. Dies gilt in gleichem Maße für die Daten zur Erstellung eines bedingten Plans als auch für die notwendigen Daten zur Ausführungsüberwachung.
- *Echtzeitfähigkeit:* Die Anforderungen an die Planungszeit variieren stark. Die Eintrittszeit eines unerwarteten Ereignisses hat entscheidenden Anteil an der Qualität der Neuplanung. Die Berechnungen müssen zeitlich vom Systemverhalten entkoppelt sein. Sie erfolgen in weicher Echtzeit, wobei die Größe der Zeitschranken erheblich variiert.
- *Modellierbarkeit:* Das Lösungsmuster setzt keine vollständige, physikalische Modellierung des Systemverhaltens voraus.
- *Entität:* Die Systemelemente (Planer und Agent) arbeiten kollektiv.
- *Modellierungssprache:* Das Lösungsmuster wurde auf Basis der PDDL (Planning Domain Definition Language) modelliert. Die PDDL ist ein akademischer Standard und verfügt über geeignete Lösungsverfahren. Die Modellierung wurde nicht erweitert, sondern um ein passendes probabilistisches Modell ergänzt.

Kernaufgabe des LM_{KO} „Probabilistische Planung“ ist die Umsetzung eines Selbstoptimierungsprozesses durch das Vorausplanen wahrscheinlichkeitsbehafteter Zukunftssituationen. Daher gliedern sich dessen **Funktionen** zunächst in die drei Phasen des Selbstoptimierungsprozesses (Bild 4-18) (vgl. Kap. 2.2.3).

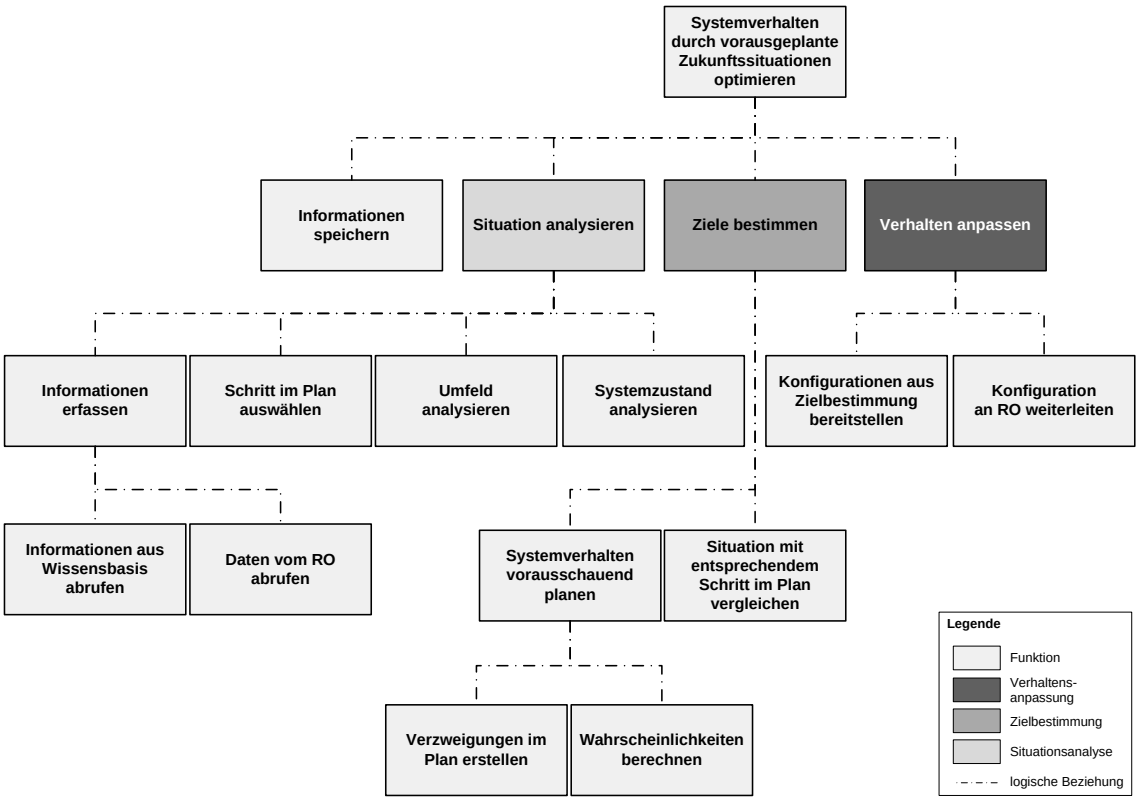


Bild 4-18: Funktionen des LM_{KO} „Probabilistische Planung“

Um eine eigenständige Optimierung funktional beschreiben zu können, sind die Funktionen *Situation analysieren*, *Ziele bestimmen* und *Verhalten anpassen* notwendig. Die Situationsanalyse ruft alle relevanten Informationen und Daten ab. Dies betrifft neben den vorhandenen Informationen aus der Wissensbasis auch vorverarbeitete Daten im RO. Hinzu kommen notwendige Funktionen zur Umsetzung der Planüberwachung. Zur Zielbestimmung sind die Funktionen *Systemverhalten vorausschauend planen* und *Situation mit entsprechendem Schritt im Plan vergleichen* durchzuführen. Die Verhaltensanpassung stellt eine Konfiguration zum Erreichen des nächsten Schrittes im Plan zum Abruf bereit. Besteht das Gesamtsystem aus mehreren mechatronischen Teilsystemen (vgl. Bild 2-8), wird je Teilsystem eine Konfiguration erstellt. Diese wird an den RO des entsprechenden Teilsystems weitergeleitet.

Eine Zuordnung dieser kognitiven Funktionen zu den Systemelementen kann im Rahmen der **Wirkstruktur** erfolgen (vgl. Bild 4-19). Der *Agent* und der *Planer* sind die beiden zentralen Systemelemente des LM_{KO} „Probabilistische Planung“. Die Aufgaben des Agenten lassen sich dabei nicht eindeutig einer Phase des Selbstoptimierungsprozesses zuordnen. Aufgrund der Analyse des Umfelds sowie der Analyse des Systemzustands ist er maßgeblich an der Situationsanalyse beteiligt. Da der Agent durch die Bewertung eines entsprechenden Schritts im Plan eine Neuplanung hervorrufen kann, wird er auch in der Zielbestimmung aktiv. Der Planer hingegen ist eindeutig der Zielbestimmung zugeordnet.

Die für den Planer notwendigen Informationen – in der Regel handelt es sich dabei um Modelle – befinden sich in der *Wissensbasis*. Mit Hilfe der Informationen kann der Planer die Wahrscheinlichkeitsverteilung für alle diskreten Systemzustände bestimmen. Die Informationen werden entweder von extern in der Wissensbasis abgelegt oder entstehen zur Laufzeit. Weitere notwendige Daten werden von dem Systemelement *Datenabruf* bereitgestellt. Es empfängt Daten vom *Datenspeicher* des RO und führt, zusammen mit dem Agenten, die Situationsanalyse aus. Komplettiert wird die Mustergruppe durch die *Verhaltensanpassung*. Der Datenspeicher ist nicht Bestandteil der Mustergruppe, wurde aber als Schnittstelle zu dem RO in das Lösungsmuster mit aufgenommen (Bild 4-19).

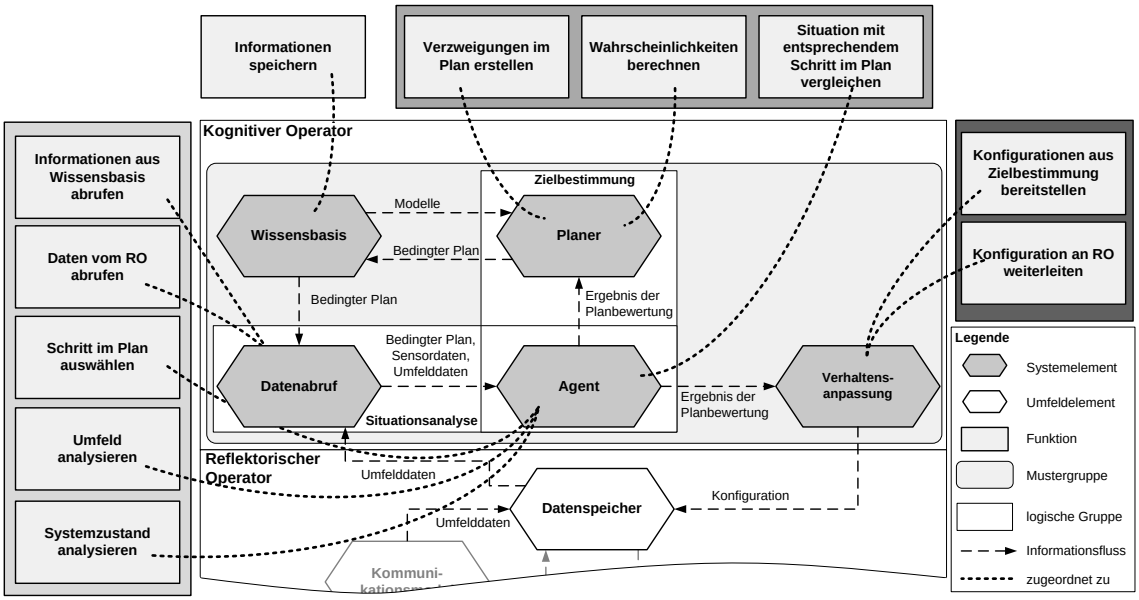


Bild 4-19: Wirkstruktur des LM_{KO} „Probabilistische Planung“ inkl. der Funktionen

Die **Verhalten – Aktivitäten** beschreiben die Abläufe der zuvor beschriebenen Systemelemente (vgl. Bild 4-20). Unmittelbar nach Prozessbeginn wird zunächst ein deterministischer Plan erzeugt. Dies geschieht offline, da das System ohne eine erstellte Strategie handlungsunfähig ist. Der erforderliche Planbaum besteht aus einer endlichen Anzahl von alternativen Verzweigungen, die in einer sich wiederholenden Schleife erzeugt werden. Nach Beendigung einer solchen Schleife findet eine Überprüfung statt, ob der Plan soweit ausgereift ist, so dass das System mit der Handlung beginnen kann. Hierbei ist zwischen drei Entscheidungsgrößen zu unterscheiden. Ist die Bedingung für einen Handlungsbeginn nicht erfüllt, werden weitere Verzweigungen erstellt (A). Dabei kann die normale sowie die Neuplanung ausgeführt werden. Ist die Bedingung hingegen erfüllt, kann das System mit der Zielverfolgung beginnen (B und C). Es kann durchaus sein, dass der Plan an dieser Stelle noch nicht fertig ist. In diesem Fall wird parallel zum ausführenden System die Planung fortgeführt (B).

Wie bereits beschrieben, wird die Situationsanalyse durch den Datenabruf und den Agenten durchgeführt. Der Fokus des Lösungsmusters liegt dabei auf dem Agenten.

Dieser ist für die Ausführungsüberwachung verantwortlich. Eine wesentliche Aufgabe ist der Vergleich des nächstmöglichen Kontrollpunktes im Plan mit den aktuellen Umfeldbedingungen sowie dem Systemzustand. Nach einer Analyse der Ist-Situation erfolgt eine Bewertung mit vier Entscheidungsgrößen. Tritt etwa ein unerwartetes Ereignis ein, das in den bestehenden Verzweigungen des Plans nicht berücksichtigt wird, veranlasst der Agent eine Neuplanung durch den Planer (A). Wenn hingegen für das Ereignis eine alternative Verzweigung existiert, wird in der anschließenden Verhaltensanpassung entsprechend reagiert und die Konfiguration gemäß der Alternative eingestellt (B). Hat die Bewertung ergeben, dass das gesamte Verhalten nach Plan verläuft, gilt es lediglich zu klären, ob das angestrebte Ziel bereits erreicht ist oder nicht. Ist es erreicht, wird das System abgeschaltet (D). Ist es hingegen notwendig weitere Schritte im Plan abzuarbeiten, dann wird der Verhaltensanpassung vermittelt, dass die Konfiguration bestehen bleiben soll (C). Der gesamte Prozess wird dabei solange wiederkehrend durchlaufen, bis das Ziel im Plan erreicht ist (Bild 4-20).

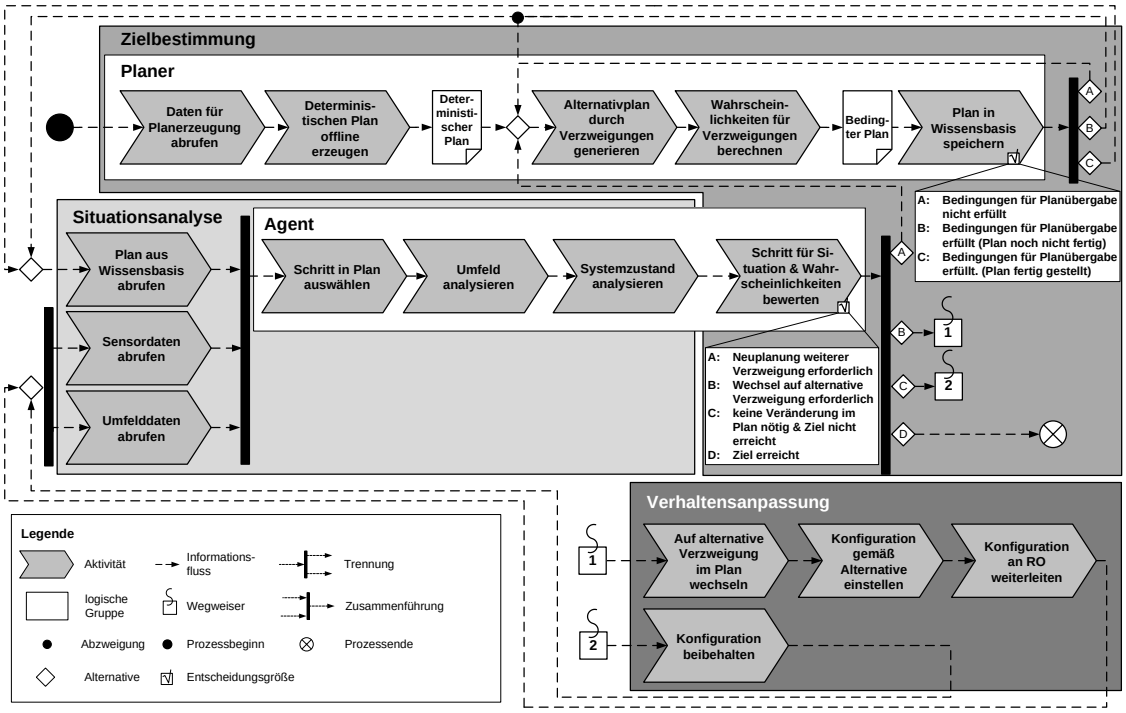


Bild 4-20: Verhalten – Aktivitäten des LM_{KO} „Probabilistische Planung“

Mögliche **Verhalten – Zustände**, die die Mustergruppe im Systemverhalten einnehmen kann, zeigt das Bild 4-21. Vor Prozessbeginn bzw. nach Erreichen des Ziels ist das *System aus*. Unmittelbar nach Prozessbeginn ist das *System an.* Es ist jedoch inaktiv, da mindestens der erste Schritt im Plan vor Handlungsbeginn geplant werden muss. Hinzu kommen zwei weitere mögliche Zustände, in denen sich das System bei der Ausführung des Plans befindet. Existiert ein vollständig erstellter Plan, so ist der *Agent aktiv* und der *Planer inaktiv*. Ist der Plan noch nicht vollständig fertiggestellt oder hat der Agent den Befehl einer Um- oder Neuplanung gegeben, wird der *Planer aktiv.*

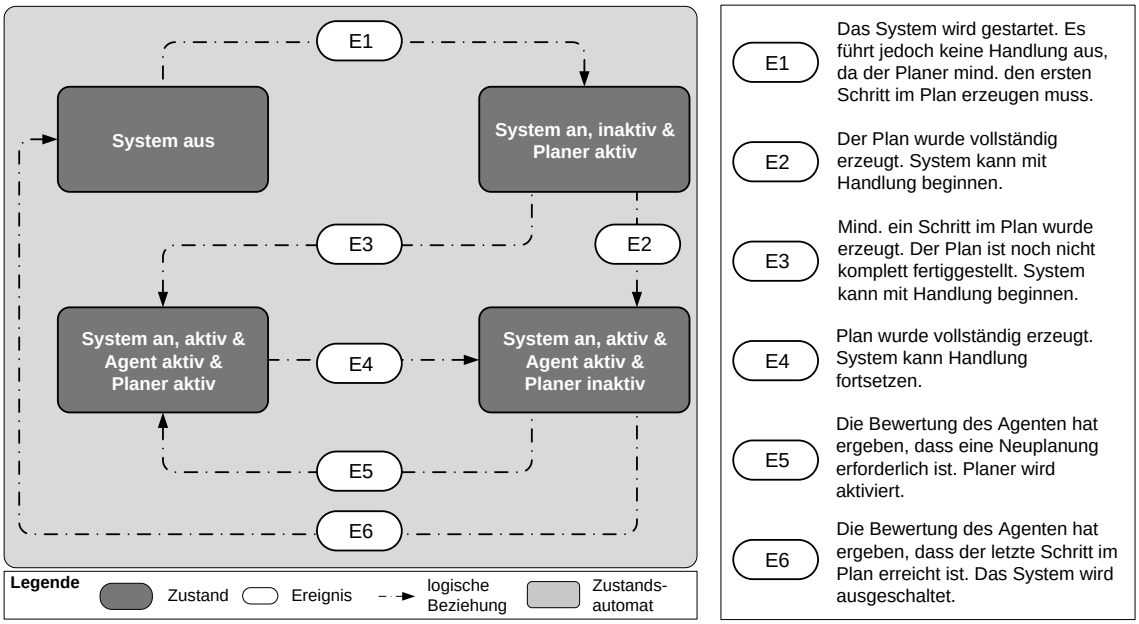
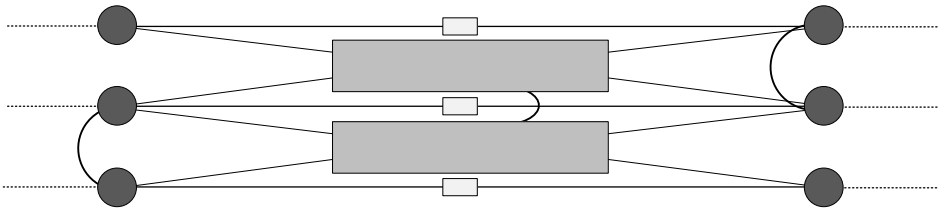


Bild 4-21: Verhalten – Zustände des LM_{KO} „Probabilistische Planung“

Mögliche **Lösungsprinzipien**, die für die Umsetzung des Lösungsmusters in Frage kommen, sind grundsätzlich Planverfahren (vgl. Kap. 4.5.3). Bild 4-22 zeigt beispielhaft ein Verfahren, das im LM_{KO} „Probabilistische Planung“ verwendet werden kann.

Graphbasiertes Planen

Verfahrensskizze



Kurzbeschreibung des Verfahrens

Ein Planungsgraph besteht aus zwei Knotenarten: Propositionsknoten und Aktionsknoten, wobei Propositionsknoten nur mit Aktionsknoten verbunden sind und umgekehrt. Der Graph wird in Ebenen unterteilt, die einen Zeitschritt im Plan enthalten. Die erste Ebene enthält dabei die Literale des Initialzustands, die Aktionen, die darauf ausgeführt werden können, und die Literale, die durch die Ausführung dieser Aktionen erzeugt werden können. Damit enthält der Plangraph *in etwa* alle Literale, die in einem Zeitschritt wahr sein könnten, und *in etwa* alle Aktionen, die in einem Zeitschritt ausgeführt werden könnten. Ein Algorithmus zur Erzeugung eines Plans aus einem Plangraphen iteriert über zwei Lösungsschritte. Im ersten Schritt werden die Aktionen der aktuellen Ebene und die Literale der nächsten Ebene eingefügt. Die folgende Lösungsextraktion überprüft, ob alle Literale des Zielzustands in der aktuellen Ebene enthalten sind und ob zwischen den enthaltenen Literalen Mutex-Relationen bestehen.

Weitere Informationen

[BF97], [GRS03], [RN07], [Klö09]

Bild 4-22: Kurzdarstellung des Verfahrens „Graphbasiertes Planen“

Grundsätzlich ist bei der Wahl der Verfahren zwischen den Aufgabenbereichen des Planers und des Agenten zu unterscheiden sowie zwischen den Phasen des Selbstoptimie-

rungsprozesses. Für die Verhaltensanpassung sind keine spezifischen Verfahren nötig. Neben dem graphbasierten Planen ist noch die Zustandsraumsuche von hoher Relevanz für die Implementierung und Umsetzung des Lösungsmusters. Auf eine detaillierte Darstellung aller möglichen Verfahren wird an dieser Stelle verzichtet. Diese wird im Rahmen der Werkzeugunterstützung (vgl. Kap. 4.6.2) sowie im Anhang (vgl. A3) aufgegriffen.

Das LM_{KO} „Probabilistische Planung“ kann in einer Vielzahl von Anwendungen eingesetzt werden. Diese Anwendungen sowie die Art und Weise der Anwendung beschreibt der Aspekt **Kontext**. Eine Anwendung im Bereich des RailCab-Systems (vgl. Kap. 2.2.3) ist die *vorausschauende Streckenplanung*. Sie ist besonders wichtig für das Energiemanagement eines RailCabs. Dieses koppelt sämtliche Teilsysteme im RailCab über deren Leistungsbedarf. Hierzu muss stets gewährleistet werden, dass ausreichend elektrische Leistung für die Verbraucher zur Verfügung steht. Die probabilistische Planung erstellt hierfür zukünftige Leistungsprofile. Für diese werden gewünschte Aktionen definiert, um bspw. in einer Notsituation die Versorgung der sicherheitsrelevanten Teilsysteme für die noch zu fahrende Strecke zu priorisieren.

Die vollständige Beschreibung des Kontexts besteht wiederum aus mehreren konkretisierten Aspekten, die die Gesamtspezifikation der Anwendung umfasst. In diesen Aspekten wird die Mustergruppe des LM_{KO} „Probabilistische Planung“ – und ggf. weitere Lösungsmuster – gekennzeichnet, um die letztliche Integration zu beschreiben. Aus Gründen des Umfangs einer solchen Kontextdarstellung, wird diese für sämtliche Lösungsmuster im Rahmen der vorliegenden Arbeit nicht weiter dargestellt.

Weitere Lösungsmuster für den kognitiven Operator, die im Rahmen dieser Arbeit dokumentiert wurden, finden sich im Anhang (vgl. A2). In den nächsten beiden Abschnitten wird für die beiden anderen Ebenen der Informationsverarbeitung im OCM jeweils ein Beispielmuster vorgestellt.

4.5.2.2 LM_{RO} „Umschaltstrategie“

Fortgeschrittene mechatronische Systeme definieren die Ausprägung ihrer Systemziele zur Laufzeit und passen dementsprechend ihr Verhalten an. Der kognitive Operator, in dem diese Ziele bestimmt werden, greift dabei nicht unmittelbar auf die Aktorik des Systems bzw. dessen Regelung im Controller zu. Er bestimmt vielmehr neue Systemparameter oder Reglerkonfigurationen, um das gewünschte Systemverhalten zu erreichen. Die Ergebnisse der Optimierung erfordern bspw. eine Umschaltung zwischen verschiedenen Reglern im Controller. Der reflektorische Operator steuert diesen Umschaltvorgang. Da es sich hierbei um eine vielfach eingesetzte Methode handelt, die Verhaltensanpassung erfolgreich umzusetzen, wurde hierfür das Lösungsmuster für den reflektorischen Operator „Umschaltstrategie“ aufgenommen (Bild 4-23).

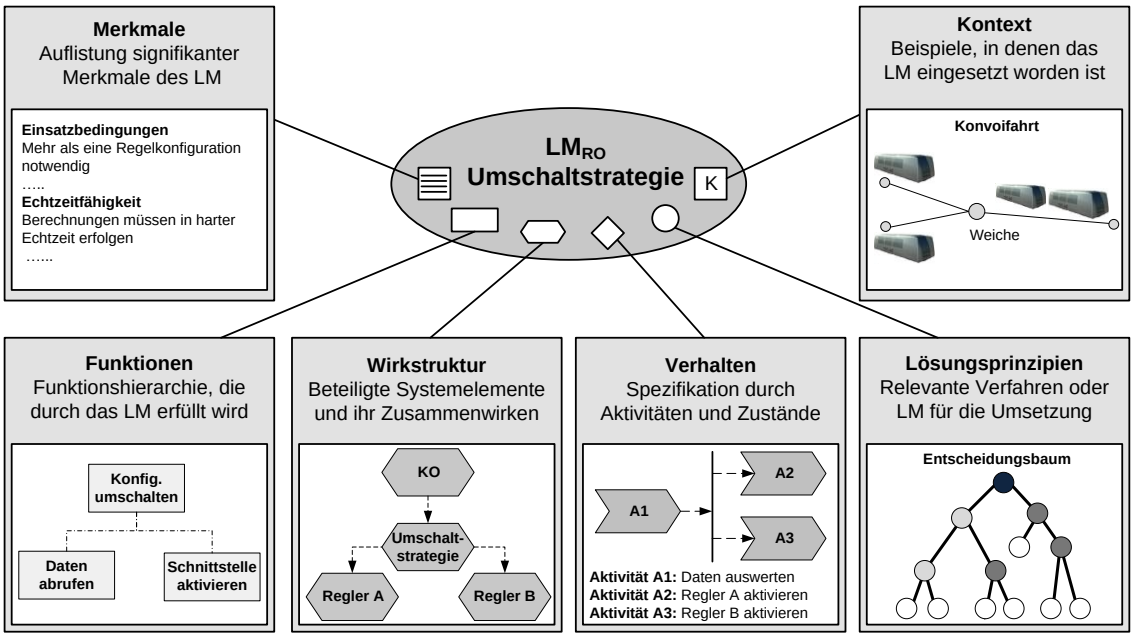


Bild 4-23: Lösungsmuster für den reflektorischen Operator „Umschaltstrategie“

Für das LM_{RO} „Umschaltstrategie“ wurden folgende **Merkmale** identifiziert:

- *Einsatzbedingungen:* Eine Konfiguration besteht im Allgemeinen aus einer festgelegten Regelschleife. Um einen Umschaltvorgang umsetzen zu können, sind daher mindestens zwei Regler im Controller erforderlich, die zu unterschiedlichen Konfigurationen führen. Eine weitere Möglichkeit ergibt sich, wenn Sollwerte an mehreren Stellen in die Regelschleife eingebunden werden können.
- *Echtzeitfähigkeit:* Die Durchführung der Berechnung erfolgt in harter Echtzeit. Bei der Anwendung dieses Lösungsmusters ist daher eine geeignete Hardware zu berücksichtigen.
- *Art der Berechnung:* Da die Konfigurationseinstellung an bestimmte Bedingungen gekoppelt ist, werden ausschließlich diskrete Werte verarbeitet. Die enge Verbindung zu dem Controller setzt eine hybride Berechnung voraus, die auch eine quasikontinuierliche Kommunikation zwischen reflektorischem Operator und Controller ermöglicht.
- *Entität:* Der Einsatz der Umschaltstrategie ist in der Regel an eine Kooperation mit anderen Systemen gebunden; d.h. es findet eine kollektive Informationsverarbeitung über die Grenzen des eigenen System-OCM statt. Hierfür werden entsprechende Rollen zur Durchführung definiert.

Das LM_{RO} „Umschaltstrategie“ beschreibt das Umschalten zwischen unterschiedlichen Konfigurationen von Reglern. Bild 4-24 zeigt die hierfür notwendigen **Funktionen**, die die Aufgabe des Lösungsmusters lösungsneutral beschreiben.

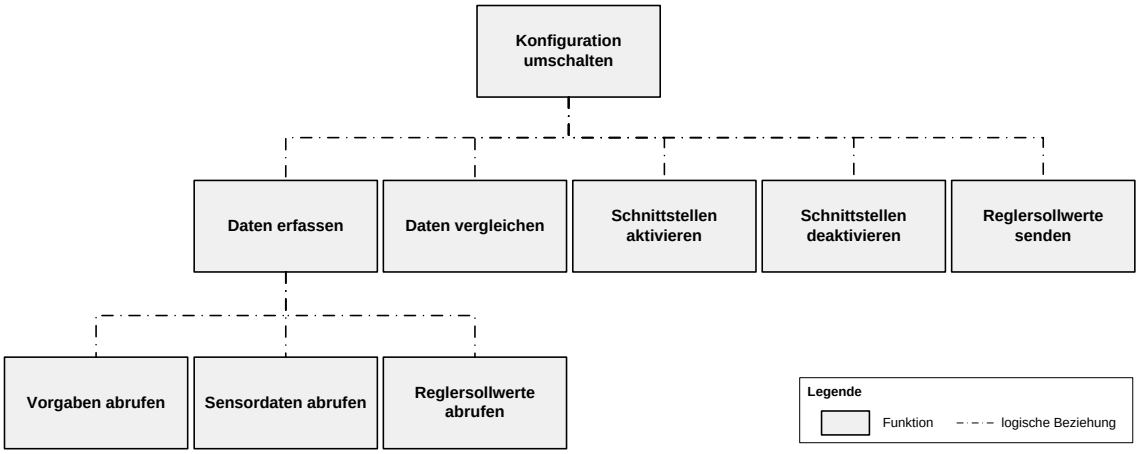


Bild 4-24: Funktionen des LM_{RO} „Umschaltstrategie“

Um die Berechnung für den Umschaltvorgang durchführen zu können, sind Vorgaben aus dem Datenspeicher (Ergebnis der Optimierung aus dem kognitiven Operator), Daten der Sensoren und Sollwerte für die anzusteuernden Regler erforderlich (*Daten erfassen*). Ferner werden die gemessenen Istwerte mit den Sollvorgaben aus dem Datenspeicher verglichen (*Daten vergleichen*) und es werden die Schnittstellen zu den Reglern aktiviert bzw. deaktiviert (*Schnittstellen aktivieren* bzw. *deaktivieren*). Der Umschaltvorgang steht in engem Bezug zu mindestens zwei Reglern, die sich im Controller des Systems befinden. Diesen werden die Sollwerte entsprechend weitergeleitet (*Reglersollwerte senden*). Die Sollwertgenerierung selbst ist nicht Aufgabe der Mustergruppe. Die **Wirkstruktur** verdeutlicht das Zusammenspiel der Systemelemente, die diese Funktionen erfüllen (Bild 4-25).

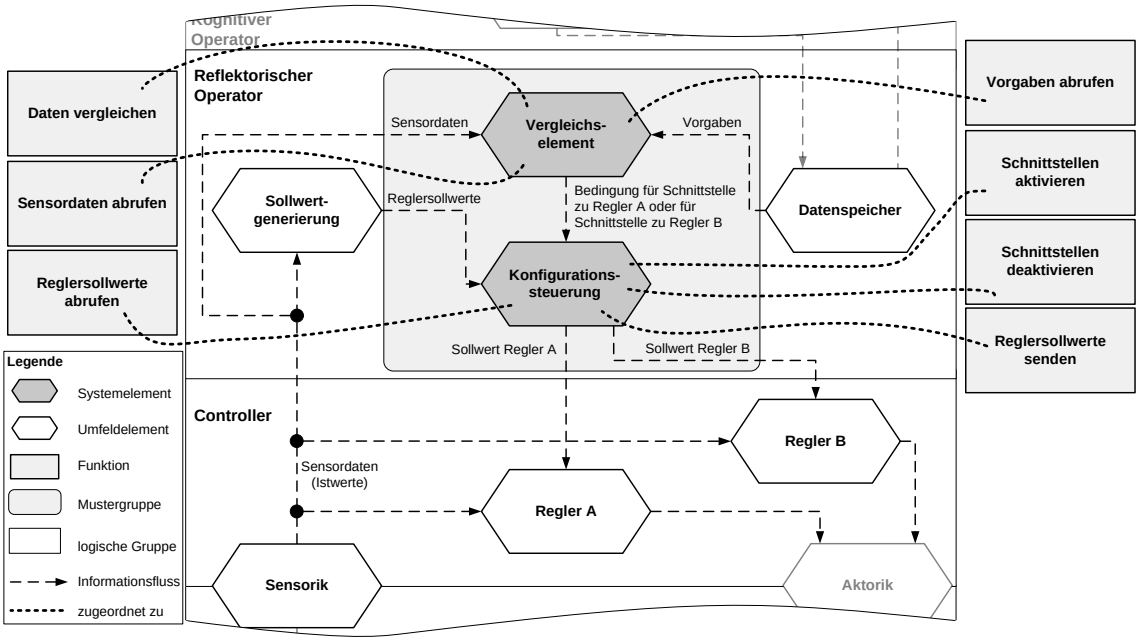


Bild 4-25: Wirkstruktur des LM_{RO} „Umschaltstrategie“ inkl. der Funktionen

Die Mustergruppe besitzt mehrere Schnittstellen, über die sie mit weiteren Systemelementen kommunizieren kann. Die erforderlichen Daten gelangen von den externen Systemelementen *Sensorik*, *Sollwertgenerierung* und *Datenspeicher* zur Mustergruppe. Die Sollwertgenerierung und der Datenspeicher befinden sich ebenfalls im reflektorischen Operator. Das Systemelement *Vergleichselement* ruft die Sensordaten sowie die Vorgaben aus dem Datenspeicher ab und überprüft, welche Bedingungen für eine entsprechende Reglerkonfiguration erfüllt sind. Das Ergebnis geht an das Systemelement *Konfigurationssteuerung*. Dieses aktiviert bzw. deaktiviert auf Basis der Ergebnisse des Vergleichselements die Schnittstellen zu den Reglern im Controller und sendet diesen die Sollwerte. Das LM_{RO} „Umschaltstrategie“ unterscheidet zwei Regler. Grundsätzlich kann das Lösungsmuster auch für mehrere Regler bzw. Reglerkonfigurationen eingesetzt werden. Da zwischen der Mustergruppe und den Reglern im Controller ein enger Bezug herrscht, ist dieser ebenfalls in der Wirkstruktur abgebildet. Die Abläufe innerhalb der Mustergruppe gibt der Aspekt **Verhalten – Aktivitäten** wieder (Bild 4-26).

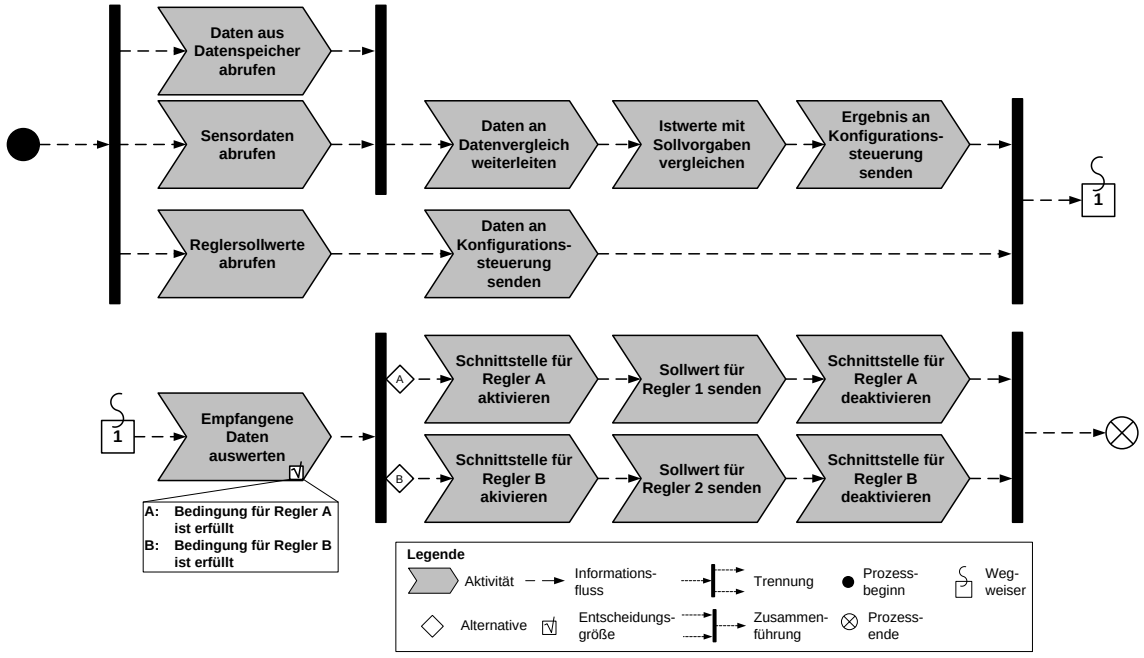


Bild 4-26: Verhalten – Aktivitäten des LM_{RO} „Umschaltstrategie“

Nach dem Abruf der Sensordaten und der Vorgaben aus dem Datenspeicher findet zunächst ein Datenvergleich statt. Im Anschluss erfolgt eine Weiterleitung an die Konfigurationssteuerung. Parallel zu diesem Prozess werden die Sollwerte abgerufen und ebenfalls weitergeleitet. Erst wenn beide Aktivitätsketten durchlaufen sind, erfolgt die Auswertung der Daten. Je nach Ergebnis führt das System eine der beiden alternativen Aktivitätsfolgen (A oder B) aus.

Mögliche Zustände der Konfigurationssteuerung zeigt der Aspekt **Verhalten – Zustände** (Bild 4-27). Dabei wird grundsätzlich davon ausgegangen, dass das gesamte System aktiv ist. Der Zustand *Regler A aktiv* tritt ein, wenn die Schnittstelle zu Regler A aktiviert wurde und die Schnittstelle zu Regler B inaktiv ist. Das System verharrt solange in

diesem Zustand, bis die Bedingungen für die Aktivierung der Schnittstelle zu Regler B erfüllt sind (*Regler B aktiv*). Die (Fehler-)Zustände *Regler A und B aktiv* und *Regler A und B inaktiv* ergeben sich aus möglichen Fehlern innerhalb der zugrundeliegenden Algorithmen. In diesen Fällen ist die Umschaltstrategie nicht auszuführen. Das System sollte in einen Notzustand kehren, bis der Fehler wieder behoben ist.

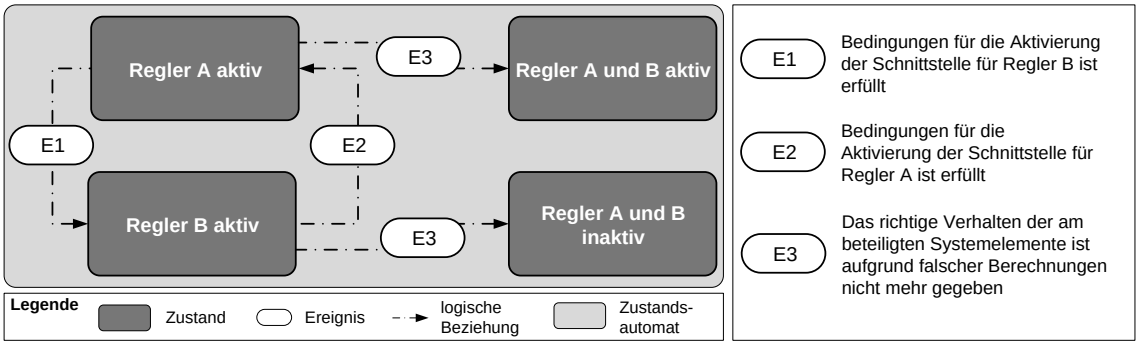
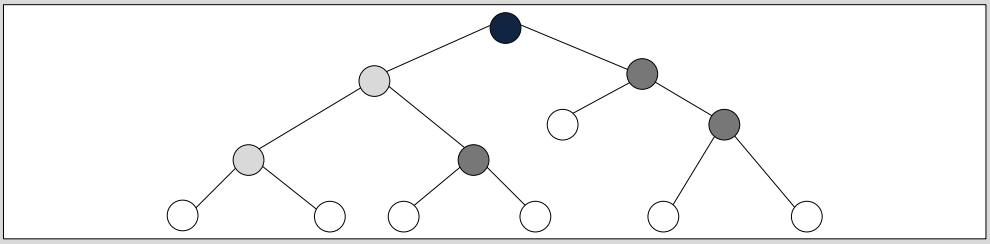


Bild 4-27: Verhalten – Zustände des LM_{RO} „Umschaltstrategie“

Als grundsätzliche **Lösungsprinzipien** kommen nur informationsverarbeitende Verfahren in Frage. Der kontinuierliche Vergleich von Bedingungen und abgespeicherten Informationen legt die Verwendung des Verfahrens „Entscheidungsbaum“ nahe. Der Aspekt Lösungsprinzipien beschreibt dieses Verfahren in Kürze und verweist auf weitere Informationsquellen (Bild 4-28).

Entscheidungsbaum

Verfahrensskizze



Kurzbeschreibung des Verfahrens

Entscheidungsbäume in der Informatik sind Abbildungen von Zuständen sowie Eigenschaften, dargestellt in Form einer Wurzel (schwarzer Kreis), welche die Kernaussage enthält, und ihrer Äste. Jeder Ast besitzt Abzweigungen, die dann zu sog. Knoten (hell grauer und dunkel grauer Kreis) und letztendlich zu sog. Blättern (weißer Kreis), dem Ende eines Baumes führen. Mit Hilfe dieses Verfahrens werden Softwareprogramme in die Lage versetzt, Entscheidungen zu treffen und so zu einer Lösung zu gelangen. Im Wesentlichen beschreibt ein Entscheidungsbaum eine Menge von Implikationen (ineinander verschachtelte WENN/DANN-Regeln), die auf unterschiedlichste Problemstellungen übertagen werden können. Die spezifische Ausprägung ist je nach Programmiersprache individuell auszuführen.

Erste Ansätze das vorgestellte Verfahren in Form eines eigenen Musters festzuhalten, liefert die Gang of Four mit dem Kompositum-Muster.

Weitere Informationen

[GHJ+04], [GRS03], [RN07], [Klö09]

Bild 4-28: Kurzdarstellung des Verfahrens „Entscheidungsbaum“

Ein Anwendungsbeispiel für den **Kontext** des LM_{RO} „Umschaltstrategie“ ist der Zusammenschluss mehrerer autonomer Fahrzeuge zu einem Konvoi. An dem Demonstrator RailCab des SFB 614 wurde diese *Konvoibildung* umgesetzt (vgl. Kap. 2.2.3). So kann z.B. der Energieverbrauch des im Windschatten fahrenden Fahrzeugs aufgrund des abnehmenden Luftwiderstandes erheblich reduziert werden. Dadurch lässt sich die Energieeffizienz deutlich erhöhen.

4.5.2.3 LM_{CO} „Kaskadenregelung“

Die Lösungsmuster für den Controller beschreiben in abstrakter Weise erste Lösungsansätze für die Regelungsaufgabe, die in den späteren Entwurfsphasen durch Modellbildung und Reglerentwurf umgesetzt wird. Ein oft verwendeter Lösungsansatz ist die Kaskadenregelung. Im Gegensatz zu einem einfachen, einschleifigen Regelkreis existieren weitere unterlagerte Regelkreise (Kaskaden). Insbesondere wenn Regelstrecken sehr große Verzögerungen besitzen, kann so ein schnelleres und besseres Führungs- und Störverhalten erreicht werden. Bei der Kaskadenregelung werden neben der eigentlichen Regelgröße, weitere Hilfsregelgrößen messtechnisch erfasst und in den Kreis zurückgeführt. Der Vorteil einer solchen komplexen Struktur ist, dass bei korrekter Auslegung die innere Kaskade schneller auf Störgrößeneinflüsse reagieren und Verzögerungen in der äußeren Regelung verringern kann. So kann das Störverhalten und die Stabilität der gesamten Regelung verbessert werden [RRV+02, S. 134f.], [Lun07, S. 540]. Bild 4-29 zeigt das LM_{CO} „Kaskadenregelung“ im Überblick, das den einfachsten Fall einer Kaskadenregelung mit einem inneren und einem äußeren Regelkreis beschreibt.

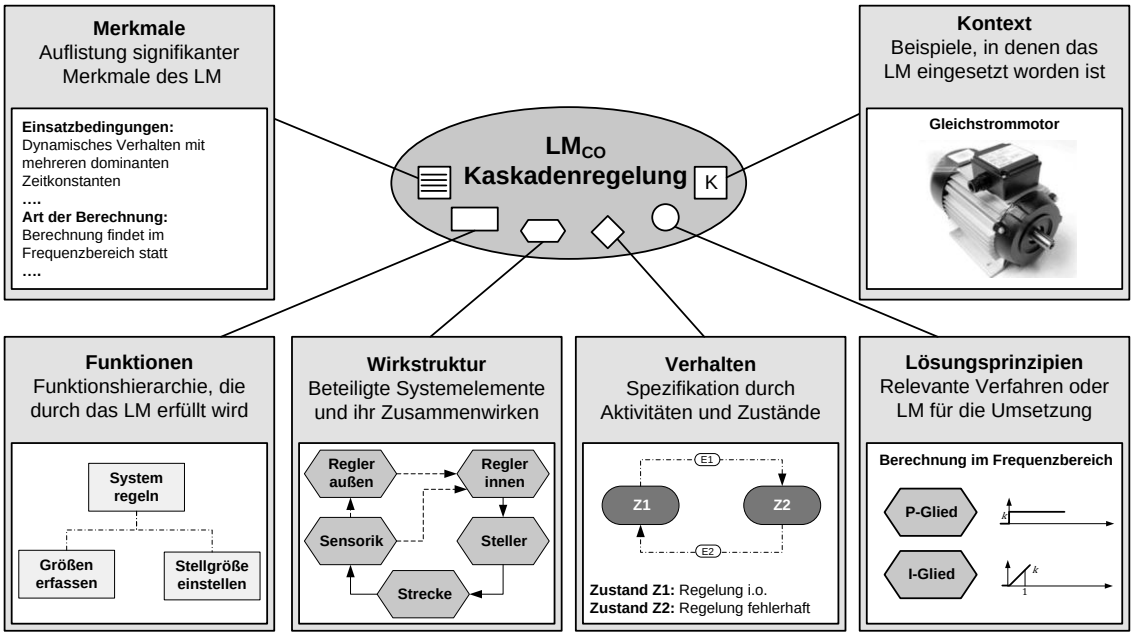


Bild 4-29: Lösungsmuster für den Controller „Kaskadenregelung“

Relevante **Merkmale** und deren Ausprägungen, die bei der Auswahl des LM_{CO} „Kaskadenregelung“ unterstützen sollen, sind:

- *Einsatzbedingungen:* Zur Umsetzung der Kaskadenregelung sind mindestens zwei Regelgrößen messtechnisch zu erfassen. Darüber hinaus muss es möglich sein, das Gesamtsystem in zwei Teilsysteme mit dominanten Zeitkonstanten zu unterteilen. Ferner müssen die Zeitkonstanten der inneren Regelkreise kleiner sein, als die der äußeren, d.h. die inneren Kaskaden müssen schneller geregelt werden können als die äußeren.
- *Echtzeitfähigkeit:* Alle beteiligten Berechnungen unterliegen harten Echtzeitbedingungen.
- *Modellierbarkeit:* Für den späteren Entwurf ist eine mathematische Beschreibung mit Differentialgleichungen notwendig. Nicht lineare Bestandteile sind zu linearisieren.
- *Art der Berechnung:* Die eingehenden Signale für die Berechnungen sind sowohl wert- als auch zeitkontinuierlich. Ferner sollte die Auslegung im Frequenzbereich mittels geeigneter Übertragungsfunktion und nicht im Zeitbereich durchgeführt werden.

Grundsätzlich besteht jede Reglung aus den drei Funktionen *messen*, *vergleichen* und *stellen*. Diese beschreiben die prinzipielle Wirkungsweise, können aber auf verschiedene Arten umgesetzt werden [Lun07, S. 6]. Aufgabe des LM_{CO} „Kaskadenregelung“ ist die Verbesserung des dynamischen Systemverhaltens durch die Aufteilung in zwei separat regelbare Kaskaden. Der Aspekt **Funktionen** ist in Bild 4-30 dargestellt.

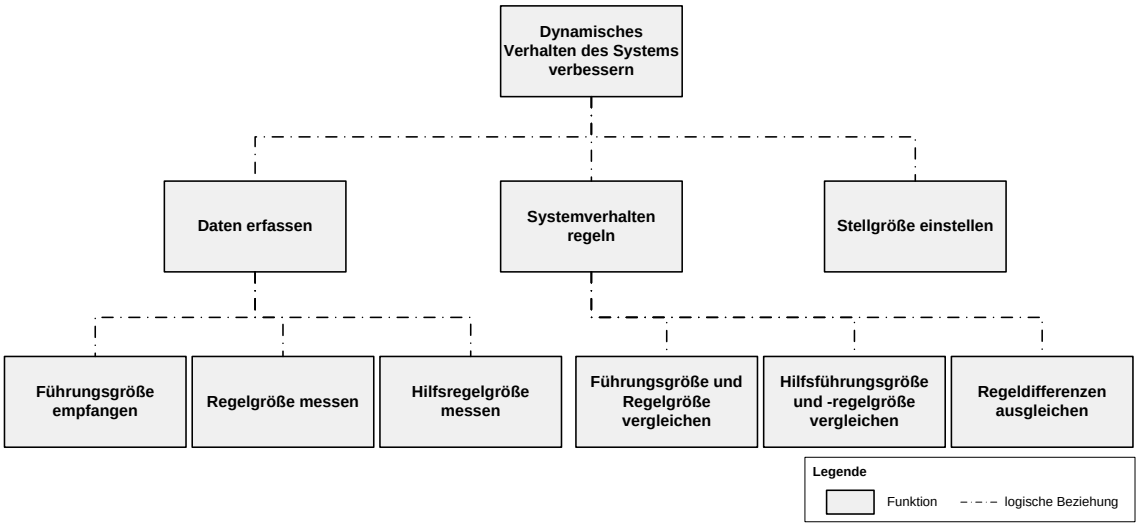


Bild 4-30: Funktionen des LM_{CO} „Kaskadenregelung“

Auf der ersten Hierarchieebene ergeben sich die Funktionen *Daten abrufen*, *Systemverhalten regeln* und *Stellgröße einstellen*. Da die Kaskadenregelung in diesem Fall nur eine äußere und eine innere Regelschleife besitzt, sind zwei Größen zu messen. Zum einen die Regelgröße der äußeren Regelschleife und zum anderen die Hilfsregelgröße der inneren Regelschleife. Zusätzlich ist die Führungsgröße zu empfangen. Die Rege-

lung des Systemverhaltens erfordert, dass die Führungsgröße mit der Regelgröße (äußere Schleife) und die Hilfsführungsgröße mit der Hilfsregelgröße verglichen werden. Die Regeldifferenzen sind jeweils auszugleichen, ehe diese dann eingestellt werden können.

Die notwendigen Systemelemente zur Umsetzung des LM_{CO} „Kaskadenregelung“ befinden sich größtenteils im Controller. Aus der **Wirkstruktur** geht hervor, dass eine enge Beziehung zu den Elementen des Grundsystems besteht, da die Sensorik die notwendigen Messgrößen erfasst und die Aktorik zur energetischen Einstellung der Regelgrößen notwendig ist (Bild 4-31).

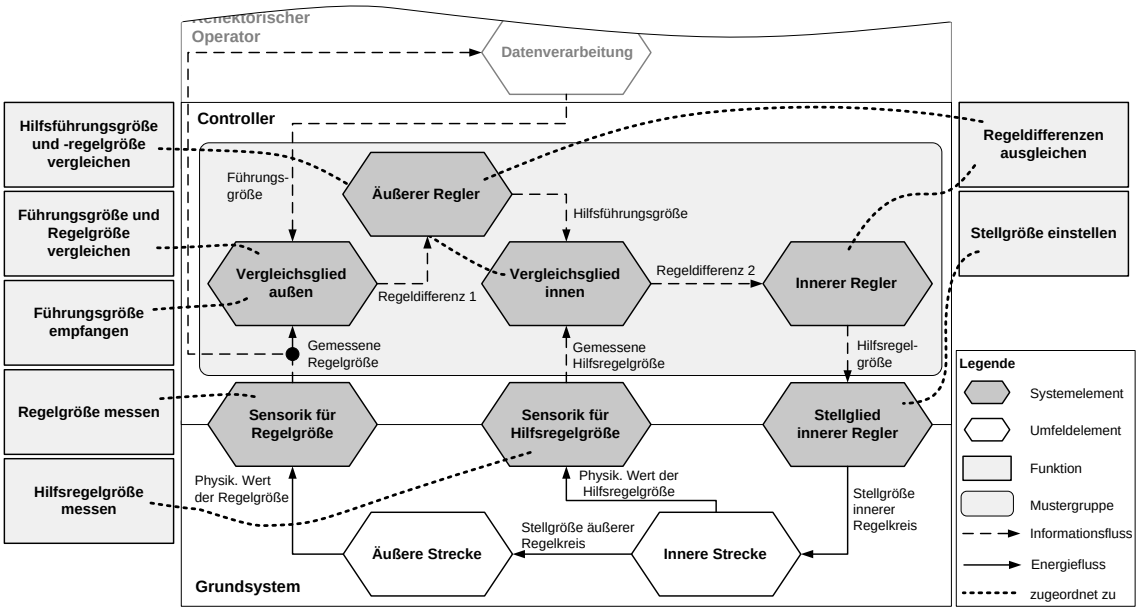


Bild 4-31: Wirkstruktur des LM_{CO} „Kaskadenregelung“ inkl. der Funktionen

Die Struktur einer Kaskadenregelung setzt sich zusammen aus einem inneren und einem äußeren Regelkreis. Während der innere Regelkreis auch nur die innere Strecke umfasst, durchläuft der äußere Regelkreis alle Systemelemente bis auf die *Sensorik für die Hilfsregelgröße*. Die Eingangsdaten für das *Vergleichsglied außen* sind die Führungsgröße sowie die gemessene Regelgröße. Die errechnete Regeldifferenz 1 wird anschließend an den *äußeren Regler* weitergeleitet. Das Berechnungsergebnis dieses Systemelements ist die Hilfsführungsgröße für die innere Regelschleife. Die gemessene Hilfsregelgröße wird an das *Vergleichsglied innen* weitergeleitet. Die Regeldifferenz wird vom *inneren Regler* ausgeglichen und die errechnete Größe von dessen *Stellglied* eingestellt. Der Ausgang der *inneren Strecke* ergibt die Stellgröße für die *äußere Strecke*.

Die Lösungsbeschreibung der Kaskadenregelung wird durch die Verhaltensbeschreibung vervollständigt. Der Aspekt **Verhalten – Aktivitäten** spezifiziert die Abläufe und das Zusammenwirken der beiden Regelschleifen. Dabei werden im Prinzip zwei Aktivitätsstränge nacheinander durchlaufen (Bild 4-32).

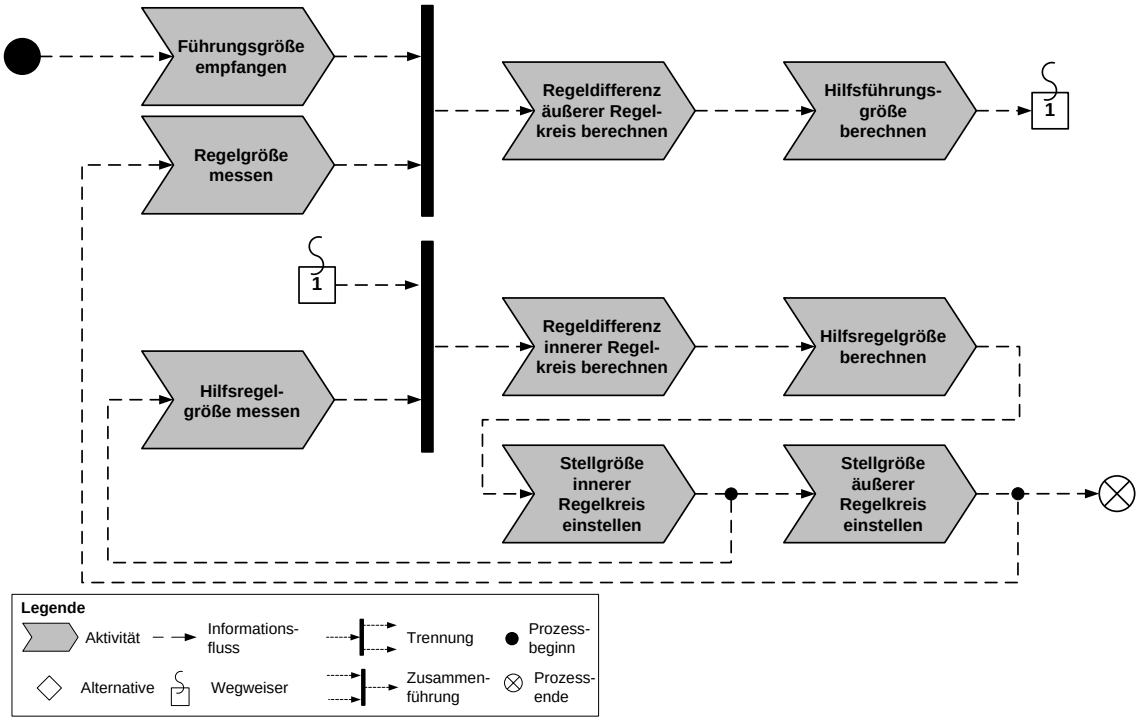


Bild 4-32: Verhalten – Aktivitäten des LM_{CO} „Kaskadenregelung“

Die Regeldifferenz hinsichtlich des äußeren Regelkreises ergibt sich aus der Regelgröße und der Führungsgröße. Aus dieser wird die Hilfsführungsgröße für den inneren Regelkreis berechnet. Nach der Berechnung der Hilfsregelgröße wird die entsprechende Stellgröße physikalisch eingestellt und das Ergebnis rückgeführt. Abschließend wird der äußere Regelkreis durch die Rückführung der Regelgröße geschlossen.

Der zweite Aspekt der Verhaltensbeschreibung sind die **Verhalten – Zustände** der Mustergruppe. Dabei wird angenommen, dass die Regelung immer aktiv ist, so dass sich zwei mögliche Zustände ergeben. Im Zustand *Regelung in Ordnung* läuft diese problemlos. Der Zustand *Regelung fehlerhaft* tritt dagegen ein, wenn ein Systemelement der Mustergruppe defekt ist (Bild 4-33).

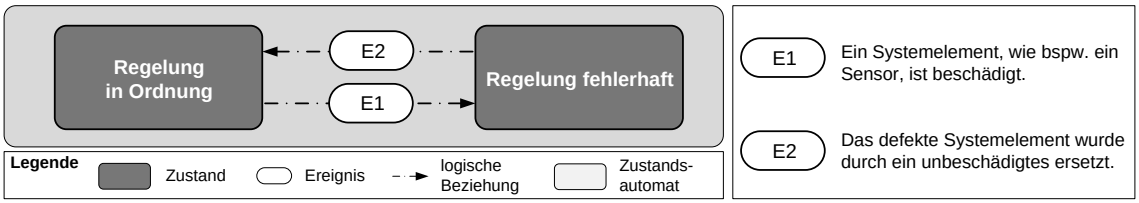


Bild 4-33: Verhalten – Zustände des LM_{CO} „Kaskadenregelung“

Für die Modellbildung und den Reglerentwurf werden einige **Lösungsprinzipien** für Lösungsmuster für den Controller empfohlen. Bei dem LM_{CO} „Kaskadenregelung“ handelt es sich um ein etabliertes Vorgehen zum Entwurf. Dabei wird grundsätzlich von innen nach außen vorgegangen, d.h. zunächst wird der innere Regler ausgewählt. Dabei ist darauf zu achten, dass dieser eine ausreichend schnelle Dynamik und Stabilität besitzt. Auf diese Weise verhält sich der Regler aus Sicht der äußeren Regelung quasi-

statisch und nimmt keinen wesentlichen Einfluss auf diesen. Im zweiten Schritt wird die gesamte innere Schleife zusammengefasst. Die Wahl des äußeren Reglers schließt den Entwurf ab. Durch die Wahl der beiden Regler ergeben sich unterschiedlich gut geeignete Kombinationen. Diese hängen von den Grundanforderungen der gewählten Reglertypen, wie z.B. P-Regler, PI-Regler oder PID-Regler, ab [Föl08, 270], [Lun07, S. 540].

Eine klassische Anwendung, die den **Kontext** des LM_{CO} „Kaskadenregelung“ beschreibt, ist die Drehzahlregelung eines *Gleichstrommotors*. Der innere Regelkreis stellt den Ankerstrom ein und ist auf optimales Führungsverhalten ausgelegt. Der äußere Regelkreis für die Drehzahl ist hingegen auf optimales Störverhalten ausgelegt.

4.5.3 Klassifikation kognitionsrelevanter Verfahren

Grundlage zur Implementierung und Umsetzung kognitiver Funktionen sind komplexe informationsverarbeitende Verfahren. Diese werden im Rahmen der vorliegenden Arbeit als kognitionsrelevante Verfahren bezeichnet. Sie sind der Kern eines Lösungsmusters für den kognitiven Operator und bestimmen dessen Struktur und Dokumentation in einem hohen Maß. Aufgrund ihrer Komplexität und in der Regel problemspezifischen Darstellung werden viele bestehende Verfahren aber erst spät oder unter Umständen gar nicht im Entwurf technischer Systeme berücksichtigt. Eine nachträgliche Integration der „Intelligenz“ in ein fertig entworfenes oder gar bereits gefertigtes System wird aber auch in Zukunft wenig erfolgsversprechend sein. Insbesondere die Vielzahl unterschiedlicher sowie ähnlicher Verfahren und die fortschreitende Entwicklung derartiger Verfahren erschwert eine methodische Auswahl während des Systementwurfs. Hinzu kommt, dass die Mehrzahl der Verfahren weder in einer einheitlichen noch vergleichbaren Form entwickelt und beschrieben wird.

Gegenstand dieses Kapitels ist daher eine Klassifikation zur Einordnung bestehender sowie zukünftiger kognitionsrelevanter Verfahren. Sie soll die Auswahl eines Verfahrens bereits in den frühen Phasen der Entwicklung protegieren und so die Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme intensivieren. Ziel ist dabei nicht, sämtliche existierende und relevante Verfahren aufzunehmen. Es werden nur Verfahren eingeordnet, die zur Dokumentation der Lösungsmuster für den kognitiven Operator verwendet wurden (vgl. Anhang A2.2 bis A2.7).

In Anlehnung an die Arbeiten im SFB 614 werden grundsätzlich modellorientierte und verhaltensorientierte (Optimierungs-)Verfahren unterschieden (vgl. Kap. 2.2.3). Diese können wiederum hinsichtlich ihrer Verwendung weiter klassifiziert werden. Bild 4-34 zeigt die **Klassifikation kognitionsrelevanter Verfahren** für fortgeschrittene mechatronische Systeme im Überblick.

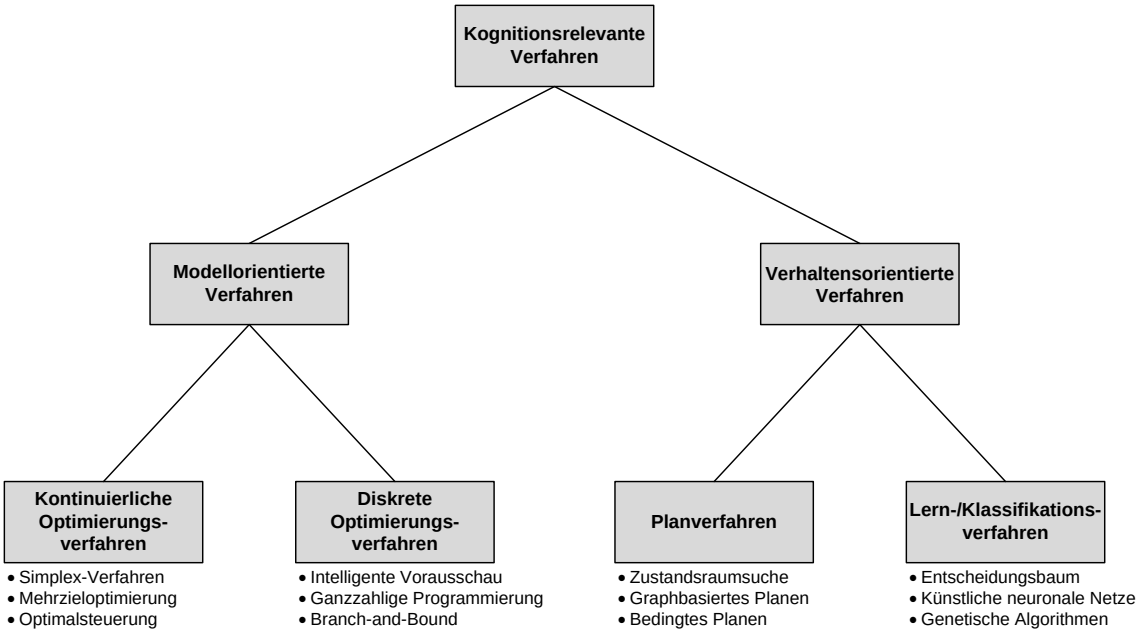


Bild 4-34: Klassifikation kognitionsrelevanter Verfahren für fortgeschrittene mechatronische Systeme im Überblick

Modellorientierte Verfahren werden größtenteils in der höheren Mathematik entwickelt. Im Fokus stehen Optimierungsverfahren. Sie optimieren ein oder mehrere Zielfunktionen des Systems. Mathematische modellorientierte Verfahren können hinsichtlich der Werte der Zielfunktionsvariablen unterteilt werden:

- **Kontinuierliche Optimierungsverfahren:** Bestehen die Zielfunktionen ausschließlich aus kontinuierlichen Variablen, die beliebige reelle oder komplexe Werte annehmen, liegt ein kontinuierliches Optimierungsproblem vor. Die entsprechenden Verfahren können danach unterschieden werden, ob eine *gesamte Trajektorie* oder nur *einzelne Lösungen* optimiert werden sollen. Ferner ist zu berücksichtigen, ob ein *lineares* oder *nicht lineares Problem* vorliegt. Wichtige Verfahren bzw. Verfahrensklassen sind das *Simplex-Verfahren*, die *Mehrzieloptimierung* und die *Optimalsteuerung* [ADG+09, S. 125ff.].
- **Diskrete Optimierungsverfahren:** Im Unterschied zur kontinuierlichen Optimierung kann bei diskreten Optimierungsproblemen eine, mehrere oder alle Variablen der Zielfunktion ganzzahlige (diskrete) Werte annehmen. Diskrete Optimierungsprobleme treten meist dann auf, wenn langfristige Einflüsse auf das Systemverhalten zu betrachten sind. Entsprechende Verfahren werden auch kombinatorische Optimierung genannt. Dabei kann die diskrete Optimierung auf Basis von *deterministischen* (und somit vollständigen) oder *stochastischen Modellen* erfolgen. Untergeordnete Verfahren sind die *Intelligente Vorausschau*, die *Ganzzahlige Programmierung* und *Branch-and-Bound-Algorithmen* [ADG+09, S. 130ff.], [OZK+08].

Verhaltensorientierte Verfahren sind in erster Linie dem Bereich der künstlichen Intelligenz zugeordnet. Diese Verfahren liefern eine effektive Möglichkeit, auf Grundlage

weniger bekannter Systemgrößen eine Optimierung des Systemverhaltens zu erzielen. Die erarbeitete Klassifikation unterscheidet dabei zwei Verfahrensklasen:

- **Planverfahren:** Als Planung wird die Aufgabe bezeichnet, eine Folge von Aktionen zu finden, die ein gewünschtes Ziel erreicht [RN07, S. 465]. Sie ermöglicht die Ermittlung eines zukünftigen Zustands anhand einer auszuführenden Aktionsabfolge. Planverfahren sind somit in der Lage, mit unvorhergesehenen Abweichungen von dem aktuellen Plan umzugehen. Wesentliche Kriterien zur Klassifikation der Planverfahren sind die *Komplexität des Planungsproblems*, die *Unsicherheit der Planungsinformationen* und die *Zeit zur Planausführung*. Die Planverfahren sind Bestandteil der Mehrzahl der aufgenommen Lösungsmuster, wie z.B. die *Zustandsraumsuche,* das *Graphbasierte Planen* oder das *Bedingte Planen.*
- **Lern-/Klassifikationsverfahren:** Hierbei handelt es um Verfahren zur Wissensakquisition, um ein System in die Lage zu versetzen, zukünftig in ähnlichen Situationen ein verbessertes Verhalten auszuführen. Durch Beobachtung und Interaktion im Systemumfeld müssen Entscheidungsprozesse abgeleitet werden, die dann unter Umständen zu einem Lernprozess führen können. Lern- und Klassifikationsverfahren können auf vielfältige Art und Weise systematisiert werden. So unterscheiden sie sich hinsichtlich der zugrundliegenden *Lernstrategie* (bspw. induktiv oder deduktiv) [Kel00, S. 346]. Ein weiteres wichtiges Kriterium ist die *Art der Informationsrückkopplung*. Hier wird zwischen überwachtem, unüberwachten und teilweise überwachtem Lernen unterschieden. Ferner wird zwischen symbolischer und subsymbolischer *Repräsentationsform* differenziert. Verfahren dieser Klasse sind der *Entscheidungsbaum, Künstliche Neuronale Netze* oder *Genetische Algorithmen.*

Die Verfahren, die im Verlag dieser Arbeit zur Spezifikation der Lösungsmuster für den kognitiven Operator eingesetzten wurden, sind im Anhang (vgl. A3) tabellarisch aufgelistet. Auf die entsprechenden Lösungsmuster wird dort verwiesen.

4.6 Werkzeugunterstützung

Wie die vorhergehenden Kapitel gezeigt haben, spielen Experten- und Erfahrungswissen in der Entwicklung von fortgeschrittenen mechatronischen Systemen eine entscheidende Rolle. Die vorgestellten Lösungsmuster ermöglichen die Externalisierung und Dokumentation dieses Wissens. Um den Entwickler im Umgang mit den Lösungsmustern zu unterstützen, bedarf es aber auch eines geeigneten IT-Konzepts. Dieses ist so zu konzipieren, dass sowohl ein zuverlässiges Hinterlegen als auch effektives Abrufen des Lösungswissens sichergestellt wird. Die Analyse des Stands der Technik hat gezeigt, dass wissensbasierte Systeme diese Anforderungen prinzipiell erfüllen, ihre Umsetzung aber letztlich spezifisch auszuarbeiten ist. Aus diesen Gründen wird der Ansatz eines wissensbasierten Systems in Form einer Wissensbasis aufgegriffen. Dessen prinzipielle Aufgaben im Rahmen der Entwicklungssystematik sowie dessen Konzept werden in

Kapitel 4.6.1 vorgestellt. Anschließend erfolgt in Kapitel 4.6.2 eine Beschreibung der prototypischen Implementierung des Konzepts.

4.6.1 Aufgaben und Konzept des Werkzeugs

Die zentrale Anforderung an ein Werkzeug für den Einsatz von Lösungsmustern für fortgeschrittene mechatronische Systeme ist, dass es die Anwendung durch Personen aus verschiedenen Fachdisziplinen unterstützt. Dabei nehmen an der Entwicklung beteiligte Personen in der Regel auch unterschiedliche Rollen im Umgang mit dem Lösungswissen ein (Bild 4-35).

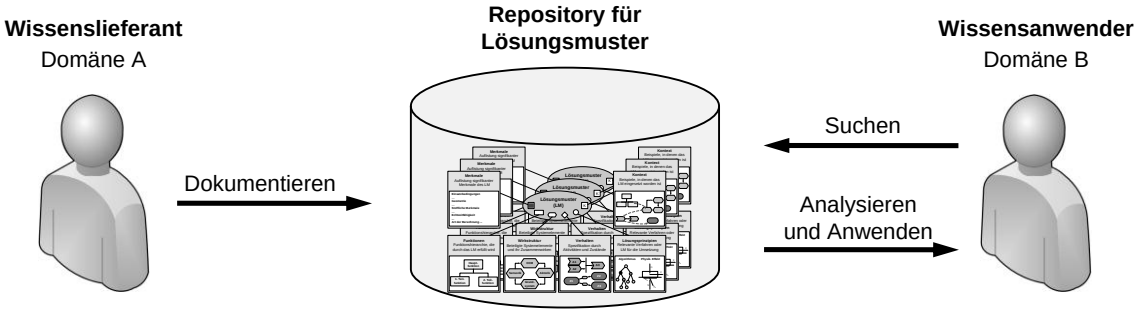


Bild 4-35: Aufgaben eines Werkzeugs für den Umgang mit Lösungsmustern

Die grundlegende Idee der Lösungsmuster ist, die Wiederverwendung von Wissen zu ermöglichen. Daher ist ein *Repository* notwendig, das die Lösungsmuster und deren Informationen festhält. In diesem Repository kann ein *Wissenslieferant* Informationen gezielt dokumentieren. Das können ein komplettes Lösungsmuster oder auch nur einzelne Aspekte als Ergänzung eines Lösungsmusters sein. Eine Sonderstellung nehmen dabei die Aspekte Kontext und Lösungsprinzipien ein. Da Lösungsmuster für verschiedene Anwendungen eingesetzt werden können, wird die Zahl der Bespiele und somit der Aspekt Kontext, der diese enthält, mit der Zeit stark wachsen. Lösungsprinzipien können zunächst auch unabhängig von einem Lösungsmuster dokumentiert werden und erst danach einem Lösungsmuster zugewiesen werden.

Auf der anderen Seite sucht ein *Wissensanwender* innerhalb des Repository nach einem Lösungsmuster, analysiert dieses im Hinblick auf seine Problemstellung und wendet dieses – falls er ein geeignetes ausgewählt hat – an. Entscheidendes Kriterium bei der Suche ist, ob der Wissensanwender Lösungsmuster identifiziert, die ähnlich zu lösende Aufgaben beschreiben. Eine einfache Suchmöglichkeit nach möglichen Analogien wäre eine Volltextsuche innerhalb der Aspekte der Lösungsmuster. Hier ergibt sich eine wesentliche Herausforderung für ein derartiges Werkzeug. Denn im Normalfall sind der Wissenslieferant und der Wissensanwender nicht nur zwei unterschiedliche Personen, sondern im Falle fortgeschrittener mechatronischer Systeme auch aus verschieden Domänen. So sind Wissenslieferanten bspw. Experten auf dem Gebiet der höheren Mathematik oder der künstlichen Intelligenz, während die Wissensanwender Entwicklungsingenieure sind. Eine reine Volltextsuche kann daher aufgrund der unterschiedlichen Ter-

minologien zu großen Ungenauigkeiten in der Suche führen. Ferner könnte das Wissen in den Lösungsmustern derart aufbereitet sein, dass es eine Person einer anderen Domäne nicht korrekt versteht.

Aufgrund der hohen Interdisziplinarität in der Entwicklung fortgeschrittener mechatronischer Systeme tauchen derartige terminologische Hürden bei der Wiederverwendung von Lösungswissen vermehrt auf und beeinträchtigen die Entwicklungsqualität. Deshalb kann ein Werkzeug für den effektiven Umgang mit den Lösungsmustern nicht nur ein einfaches Repository sein, sondern muss eine domänenübergreifende Dokumentation und Suche unterstützen. Ziel ist daher eine Wissensbasis, mit der Lösungsmuster aufgenommen und verwaltet werden können. Ferner verfügt diese über ein Zugriffssystem, das die gezielte Suche und Auswahl dokumentierter Lösungsmuster erlaubt. Das zu erarbeitende Werkzeug wird daher **Lösungsmusterwissensbasis** genannt. Da die einheitliche Spezifikation der Lösungsmuster aus Kapitel 4.5.1.1 die domänenübergreifende Dokumentation von Lösungswissen bereits unterstützt, orientiert sich die Lösungsmusterwissensbasis an den Aspekten der Lösungsmusterspezifikation (Bild 4-36).

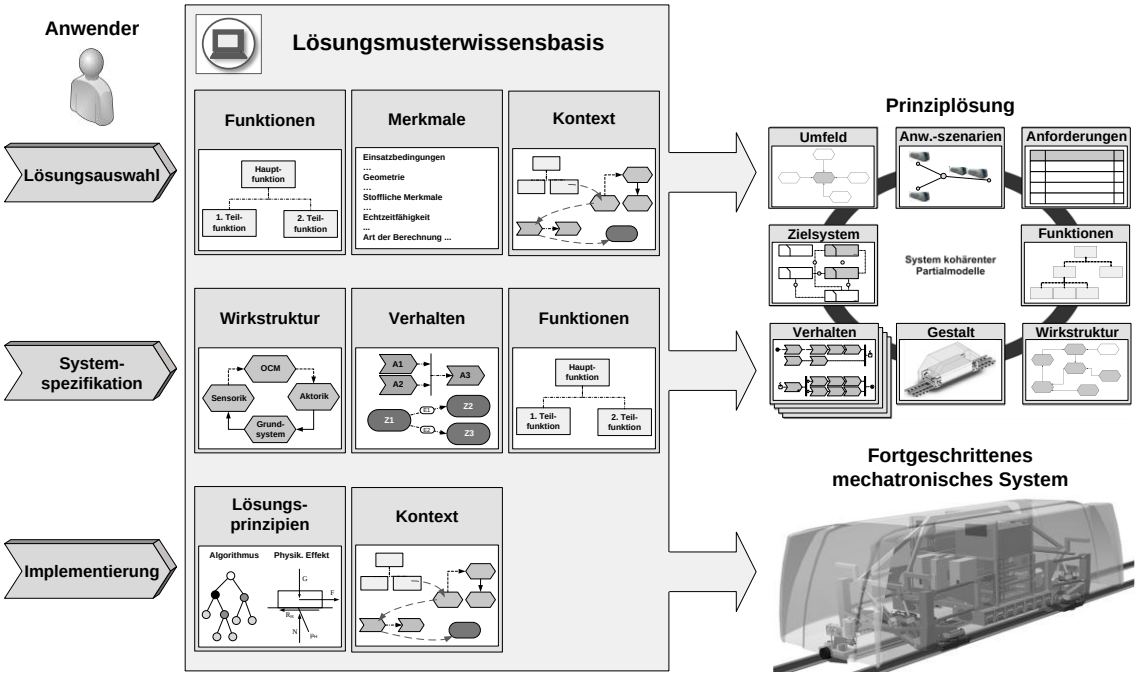


Bild 4-36: Lösungsmusterwissensbasis – Werkzeugunterstützung im Umgang mit den Lösungsmustern für fortgeschrittene mechatronische Systeme

Grundsätzlich sind im Umgang mit den Lösungsmustern drei Anwenderphasen zu unterscheiden. Die Phase **Lösungsauswahl** wird zunächst durch den Aspekt Funktionen ermöglicht. Die Funktionen bieten einen methodischen Zugang zu den Lösungsmustern. In Anlehnung an das etablierte Vorgehen im Systementwurf (vgl. Kap. 2.4.3) werden zu erfüllende Funktionen mit denen der Lösungsmuster verglichen. Zusätzlich können die spezifizierten Merkmale abgefragt und dokumentierte Beispiele im Kontext des Lösungsmusters zur Entscheidungsfindung herangezogen werden.

Für die **Systemspezifikation** sind die Aspekte von entscheidender Bedeutung, die den Kern der Lösung umfassen. Das sind die Wirkstruktur, die entsprechende Systemelemente enthält, und das Verhalten der Systemelemente. Ferner sollte geprüft werden, ob nicht noch weitere Funktionen des Lösungsmusters relevant sind, die bislang in der Spezifikation nicht berücksichtigt wurden. Die beiden ersten Phasen finden sich auch im Vorgehensmodell der Entwicklungssystematik wieder und führen schließlich zur Integration eines oder mehrerer Lösungsmuster in die Prinziplösung (vgl. Kap. 4.3.1).

Die letzte Phase ist die **Implementierung**. Diese soll die Lösungsmusterwissensbasis durch die Beschreibung der Lösungsprinzipien unterstützen. Hinzu können bereits implementierte Beispiele auf Analogien hinsichtlich der Implementierungstätigkeiten hinweisen. Die Implementierungstätigkeiten selbst werden nicht unterstützt. Resultat ist das vollständig entwickelte fortgeschrittene mechatronische System.

Für die rechnerinterne Abbildung der Lösungsmuster ist es nicht notwendig eine Modellierungsfunktion in die Wissensbasis zu integrieren. Werkzeuge, die die Modellierung der Aspekte der Lösungsmuster mit der Spezifikationstechnik des SFB 614 in Form von Partialmodellen ermöglichen, sind vorhanden und sollen für die Spezifikation der Aspekte Merkmale, Funktionen, Wirkstruktur und Verhalten genutzt werden. Hierfür muss ein passendes Modellierungswerkzeug an die Lösungsmusterwissensbasis angebunden werden, so dass der Anwender direkt die entsprechenden Partialmodelle aus der Lösungsmusterwissensbasis heraus generieren und im Modellierungswerkzeug einsehen und ändern kann. Dabei ist sicherzustellen, dass die Lösungsmusterwissensbasis auf alle Informationen innerhalb der Partialmodelle zugreifen kann, um eine Suche in diesen zu ermöglichen. Von zentraler Bedeutung ist dabei die Suche nach Funktionen. Das Konzept der Lösungsmusterwissensbasis soll semantisch ähnliche Funktionsverben bei der Suche berücksichtigen, um u.a. den Einfluss unterschiedlicher Terminologien zu reduzieren. Die Funktionsverben zur Beschreibung der Informationsverarbeitung wurden bereits in Kapitel 4.4.3 vorgestellt. Um diesen Synonyme oder ähnliche Verben zuzuordnen wird ein entsprechendes Konzept notwendig. Die Lösungsmusterwissensbasis soll hierfür auf eine Funktionsverbenontologie zurückgreifen.

Ontologien sind Systeme zur Wissensrepräsentation und umfassen eine explizite Spezifikation einer Begriffsbildung. Durch die formale Darstellung von Begriffen und Beziehungen wird das Verständnis eines Anwendungsbereichs abgebildet [Gru93, S. 199]. So kann Wissen derart formalisiert werden, dass ein automatisierter Datenaustausch erfolgen kann. Darüber hinaus verfügen Ontologien über einfache Inferenzmechanismen, mit denen logisch vorhandene, aber nicht spezifizierte Relationen automatisiert erschlossen werden [Stu09, S. 8], [Web10, S. 32]. Bild 4-37 zeigt die grundlegende Struktur der Funktionsverbenontologie für die Lösungsmusterwissensbasis, um in dem Partialmodell Funktionen der Lösungsmuster nach ähnlichen Funktionen zu suchen.

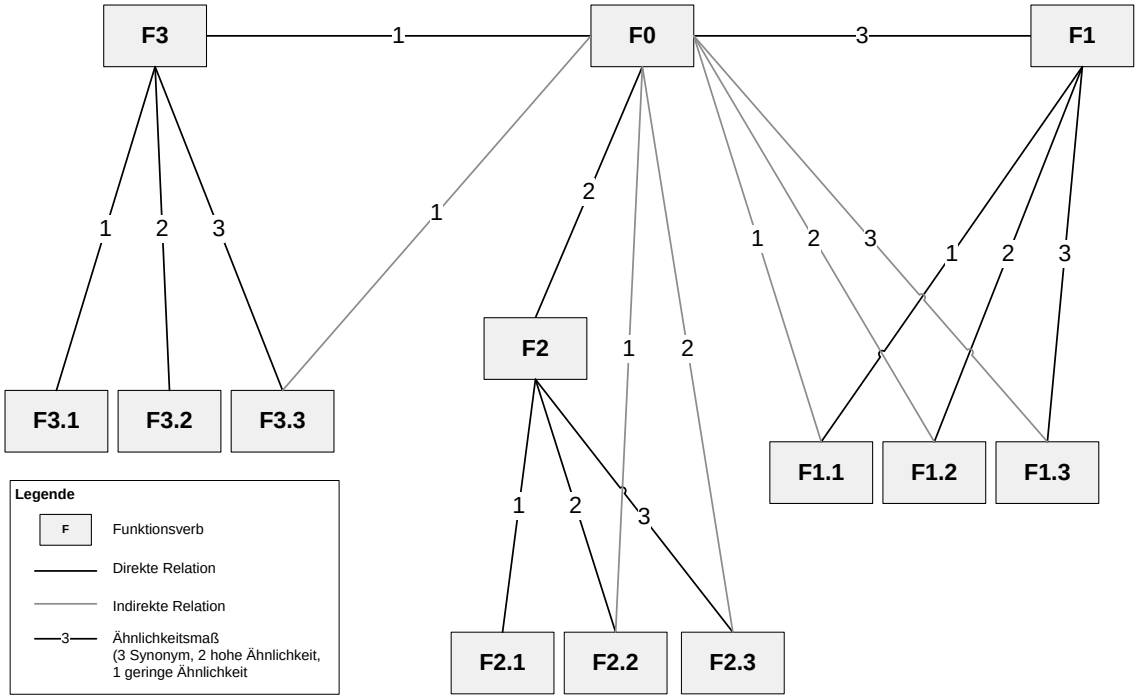


Bild 4-37: Struktur der Funktionsverbenontologie zur Suche nach Lösungsmustern

Um die Suche nach ähnlichen Funktionsverben zu ermöglichen, erhält jede Relation ein Ähnlichkeitsmaß. Dabei werden Synonyme (3), Verben mit hoher Ähnlichkeit (2) und Verben mit geringer Ähnlichkeit (1) unterschieden. Auf dieser Basis kann das Suchergebnis gewichtet werden. Wird bspw. nach einer Funktion mit dem Funktionsverb F0 gesucht, werden ebenfalls die Funktionsverben F1, F2 und F3 abgefragt, deren Ergebnis aber entsprechend des Ähnlichkeitsmaßes ausgegeben wird. So werden die Treffer für F1 genauso gewichtet wie die für das ursprüngliche Suchverb F0, wohingegen die Treffer für F2 und F3 geringer bewertet werden. Neben einer solchen direkten Relation existieren noch indirekte Relationen. Diese entstehen, wenn ein Funktionsverb zwar keine direkte Relation zu dem gesuchten Funktionsverb besitzt, aber durch ein drittes Funktionsverb diese Relation gesetzt werden kann. Dabei werden die indirekten Relationen entsprechend des Ähnlichkeitsmaßes des koppelnden Funktionsverbs gewichtet. Synonyme vererben ihre Relationen unverändert weiter, Verben mit hoher Ähnlichkeit reduzieren das Ähnlichkeitsmaß ihrer indirekten Relationen um den Wert „1“ und Verben mit geringer Ähnlichkeit um den Wert „2“ (vgl. Bild 4-37).

Der Aspekt Kontext wird aufgrund seines Aufbaus im Konzept ähnlich wie ein Lösungsmuster behandelt (vgl. Kap. 4.5.1.1). Dessen Aspekte Funktionen, Wirkstruktur und Verhalten werden wiederum mit einem externen Modellierungswerkzeug in Partialmodelle überführt, auf die die Lösungsmusterwissensbasis zugreifen kann.

Der einzige Aspekt der vollständig in der Lösungsmusterwissensbasis abzubilden ist, sind die Lösungsprinzipien. Im Kontext der Integration kognitiver Funktionen stehen hier kognitionsrelevante Verfahren im Mittelpunkt. Entsprechend deren Klassifizierung aus Kapitel 4.5.1.1 müssen hierfür eine Kurzbeschreibung, eine Skizze und weitere In-

formationen aufgenommen werden. Um die Implementierung im weiteren Verlauf der Entwicklung zu unterstützen, sollten auch Dokumente mit konkreten Hinweisen und Beschreibungen der Verfahren in die Lösungsmusterwissensbasis ablegbar sein. Die einzelnen Verfahren eines Lösungsmusters sollten auch der Phase des Selbstoptimierungsprozesses zugeordnet werden. So kann dem Anwender jederzeit ein Überblick gewährt werden, welche Verfahren zu welchem Zweck im kognitiven Operator einzusetzen sind. Dies soll auch die Entdeckung und Entwicklung unbekannter Verfahrenskombinationen für neue Lösungsmuster des kognitiven Operators fördern.

4.6.2 Prototypische Implementierung

Im Rahmen der vorliegenden Arbeit wurden die wesentlichen Bestandteile des Konzepts prototypisch als Softwarewerkzeug umgesetzt[56]. Bild 4-38 zeigt die grafische Benutzeroberfläche des entwickelten Werkzeugs.

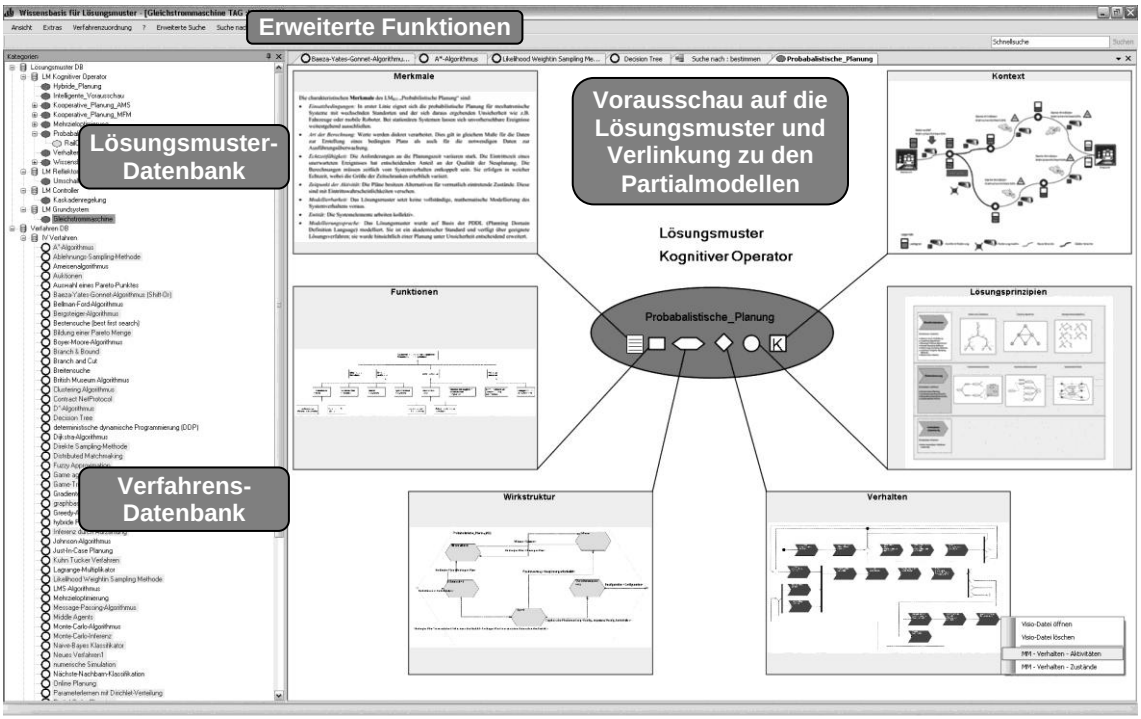


Bild 4-38: Grafische Benutzeroberfläche der Lösungsmusterwissensbasis (LMWB)

Die Lösungsmusterwissensbasis (LMWB) umfasst zwei Datenbanken: eine Lösungsmusterdatenbank, die die dokumentierten Lösungsmuster verwaltet und nach der Kategorisierung in Bild 4-15 ordnet, und eine Verfahrensdatenbank, die prinzipielle Lösungsprinzipien der Lösungsmuster beinhaltet. Hinsichtlich der Unterstützung der Ent-

[56] Die LMWB wurde mit Microsoft Visual C# 2008 in der entsprechenden Entwicklungsumgebung (Microsoft Visual Studio 2008) programmiert. Die Oberfläche der GUI wurde mit Hilfe von DockPanel Suite erstellt. Für die Verlinkung und Speicherung der Daten wurde eine MySQL-Datenbank angelegt, so dass mehrere Entwickler an unterschiedlichen Standorten auf die Daten zugreifen können.

wickler bei der Integration kognitiver Funktionen wurden in erster Linie Lösungsmuster für den kognitiven Operator und kognitionsrelevante Verfahren in der LMWB dokumentiert. Die Menüleiste erlaubt den Zugriff auf erweiterte Funktionen, wie z.B. die Suchfunktionen. Das Hauptfenster der Anwendung nimmt die Darstellung der Lösungsmuster ein. Es kann immer ein Lösungsmuster angezeigt werden inkl. einer Vorschau auf die einzelnen Partialmodelle. Über die Vorausschaubilder gelangt der Anwender schnell und einfach zu den modellierten Partialmodellen.

In Anlehnung an das vorher beschriebene Konzept werden die Partialmodelle zum größten Teil durch ein externes Modellierungswerkzeug spezifiziert. An die LMWB sind hierfür zwei weitere Softwarewerkzeuge angebunden, mit denen die Partialmodelle erstellt werden können. Das eine ist Microsoft Visio, das mit der LMWB im Hintergrund gestartet wird. Passende Shapeschablonen unterstützen die Erstellung der entsprechenden Visiodiagramme. Das zweite Tool ist der eigens für die Spezifikationstechnik des SFB 614 entwickelte Mechatronic Modeller (vgl. Kap. 3.2.3.3). Dieser ist für die Erstellung der Partialmodelle der Lösungsmuster vorzuziehen, da er zusätzlich über dedizierte Funktionen verfügt wie z.B. die Verlinkung der Partialmodelle untereinander. Auch dieser wird direkt aus der LMWB gestartet. Bild 4-39 veranschaulicht die Werkzeugkopplung der LMWB und des Mechatronic Modellers.

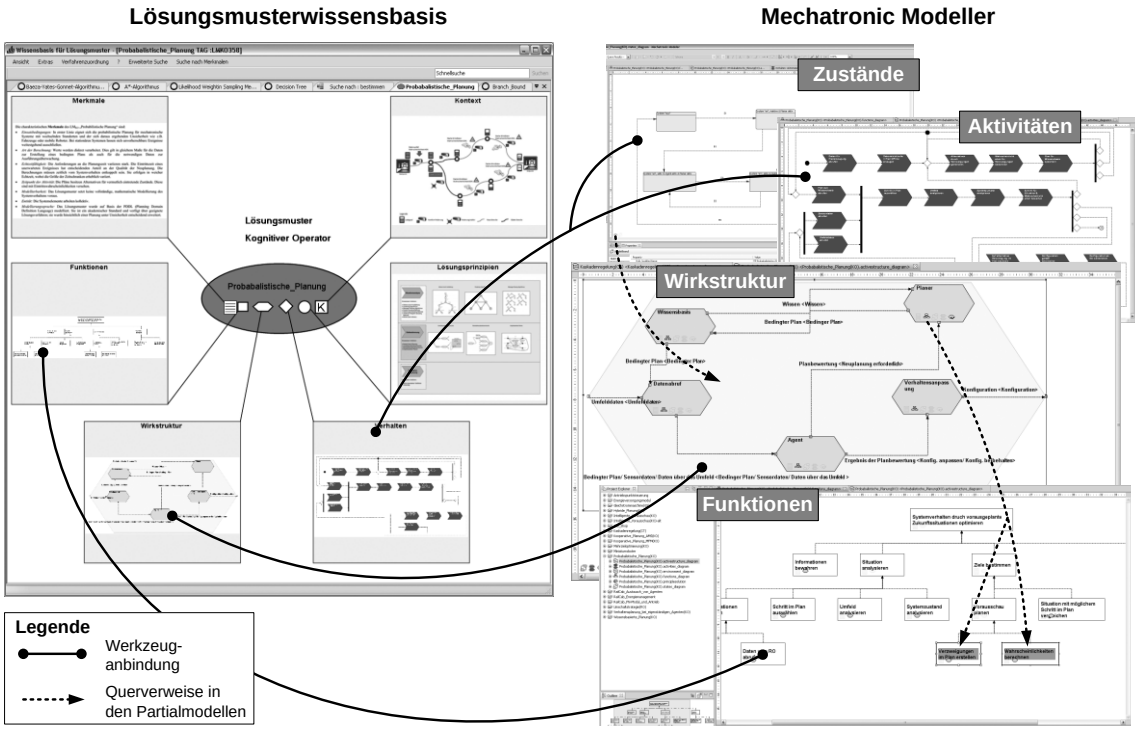


Bild 4-39: Anbindung des Mechatronic Modellers an die Lösungsmusterwissensbasis

Die Schnittstellen der LMWB und des Mechatronic Modellers kommunizieren mit Hilfe von WebServices. Auf diese Weise kann die LMWB direkt auf den Mechatronic Modeller zugreifen und es können zusätzlich wichtige Informationen ausgetauscht werden.

Bspw. werden die Vorausschaubilder der Lösungsmuster in der LMWB durch im Mechatronic Modeller intern erzeugte Screenshots aktualisiert.

Im Weiteren folgt eine kurze Erläuterung, wie die LMWB die in Bild 4-35 dargestellten Aufgaben eines Werkzeugs für den Umgang mit Lösungsmustern erfüllt.

Dokumentieren

Der Anwender kann neue Lösungsmuster oder Verfahren anlegen. Hierzu öffnet sich per Mausklick ein entsprechendes Kontextmenü. Es können Lösungsmuster entsprechend der Ebenen des OCM sowie für das Grundsystem erstellt werden. Danach wird in der LMWB das Lösungsmuster mit Vorausschaubildern angelegt. Gleichzeitig wird der Mechatronic Modeller gestartet und ein Projekt mit demselben Namen angelegt. Das Lösungsmuster in der LMWB und das Projekt im Mechatronic Modeller sind über die Partialmodelle Merkmale, Funktionen, Wirkstruktur und Verhalten direkt verbunden (vgl. Bild 4-39). Im Mechatronic Modeller erfolgt dann die Modellierung der genannten Partialmodelle. Dazu können auch Querverweise zwischen den Partialmodellen gezogen werden. Auf diese Wiese werden bspw. Funktionen bestimmten Systemelementen zugeordnet, wie es die Lösungsmusterspezifikation auch vorsieht. Um den Kontext eines Lösungsmusters zu spezifizieren, können mehrere Anwendungsbeispiele über die LMWB angelegt werden. Jedes Anwendungsbeispiel erhält wiederum ein eigenes Projekt im Mechatronic Modeller.

Zwei Arten von Verfahren können angelegt werden. Verfahren der Informationsverarbeitung und physikalische Effekte. Letztere sind nicht Bestandteile der vorliegenden Arbeit. Bild 4-40 zeigt die Dokumentation eines Verfahrens in der LMWB.

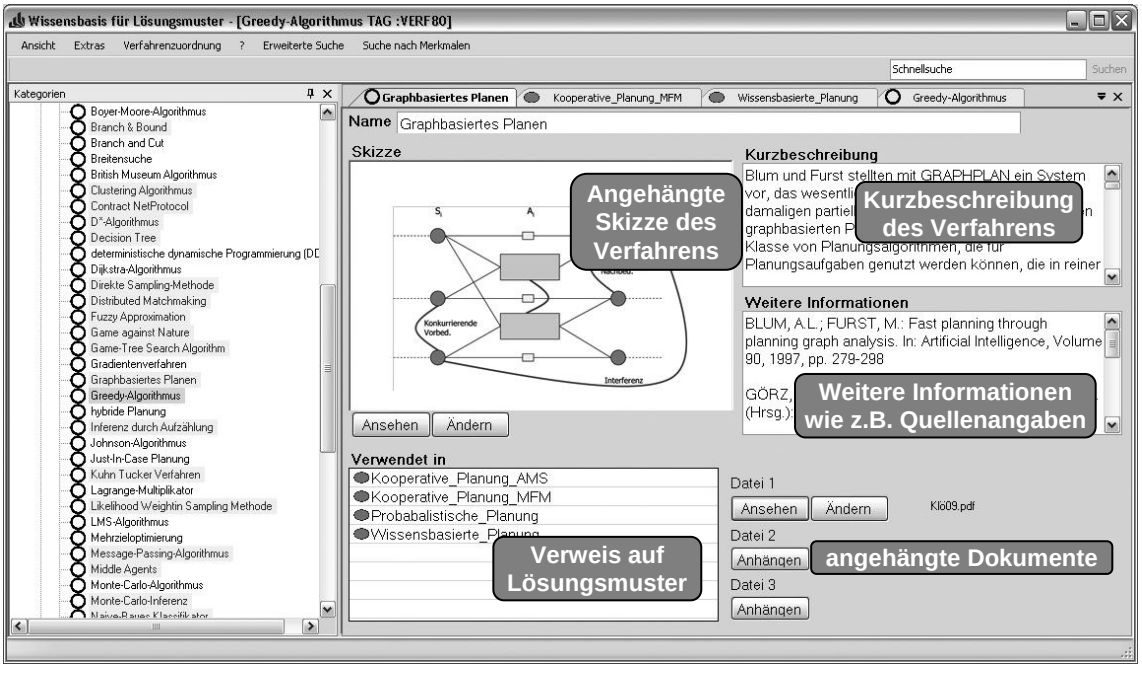


Bild 4-40: Dokumentation eines Verfahrens in der LMWB

Jedem Verfahren kann eine Skizze zugewiesen werden, die prägnant den Kern des Verfahrens darstellt. Für eine Kurzbeschreibung des Verfahrens als auch für weitere Informationen existieren zwei Textkonsolen. Darüber hinaus können bis zu drei Dokumente an die Verfahrensdokumentation angehängt werden. Falls ein Verfahren in einem Lösungsmuster verwendet wird, zeigt ein Fenster in der unteren linken Ecke die jeweiligen Lösungsmuster an. Per Mausklick gelangt der Anwender direkt zu diesen.

Die Spezifikation der Lösungsprinzipien eines Lösungsmusters wird als einziges Partialmodell vollständig in der LMWB durchgeführt. Hierzu wird eine Zuweisungsmaske geöffnet, in der aus bereits angelegten Verfahren und Lösungsmustern relevante Lösungsprinzipien ausgewählt und zugegewiesen werden können. Für fortgeschrittene mechatronische Systeme können die Lösungsprinzipien auch den drei Phasen des Selbstoptimierungsprozesses zugeordnet werden (Bild 4-41).

Bild 4-41: Spezifikation der Lösungsprinzipien eines Lösungsmusters

Suchen

Für die Suche nach Informationen in der LMWB wurde eine Volltextsuche implementiert, die aber für bestimmte Bereiche eingegrenzt werden kann. Die Volltextsuche ist eine effiziente und schnelle Suchfunktion, die im Vorfeld als geeignete Technik identifiziert wurde. Die LMWB bietet dem Anwender drei Suchoptionen. Die erste ist die **Schnellsuche**. Mit ihr können Begriffe ohne weitere Konkretisierung der Suchanfrage innerhalb der Datenbank abgefragt werden. Als Ergebnis werden alle dokumentierten Informationen, die den Suchbegriff enthalten aufgelistet. Da die Schnellsuche nur sehr grobe Suchergebnisse liefert, wurde eine **erweiterte Suche** umgesetzt (vgl. Bild 4-42). Diese unterstützt bei der gezielten Suche nach Verfahren oder Lösungsmustern. Ferner kann auch nur nach einer bestimmten Kategorie von Verfahren oder Lösungsmuster gesucht werden. Um die Suche dem Vorgehen in der Phase der Lösungsauswahl anzupassen, wurde die Suche noch weiterer fokussiert, so dass in bestimmten Partialmodel-

len der Lösungsmuster gesucht werden kann. Dies ist insbesondere für die Funktionen von hoher Relevanz. Auf diese Weise kann nach Lösungsmustern gesucht werden, die bestimmte Funktionen erfüllen. Durch die beschriebene Anbindung des Mechatronic Modellers kann die LMWB direkt in dessen Spezifikationen nach Begriffen suchen.

Für die funktionale Beschreibung der Informationsverarbeitung und der Suche nach entsprechenden Lösungsmustern wurde die im vorhergehenden Kapitel beschriebene Funktionsverbenontologie aufgestellt[57]. Dazu wurden den Funktionsverben der Informationsverarbeitung aus Kapitel 4.3.3 Synonyme mit einem entsprechendem Ähnlichkeitsmaß zugeordnet. Die Synonyme wurden unter Zuhilfenahme zweier Lexika identifiziert und in einem Kreis von Experten verschiedener Fachdisziplinen mit einem Ähnlichkeitsmaß bewertet [Dud07a], [Dud07b]. Die erstellte Funktionsverbenontologie wurde direkt an die LMWB angebunden. Sie funktioniert als Schnittstelle zwischen der Suchfunktion der LMWB und den im Mechatronic Modeller erstellten Partialmodellen. Auf diese Weise werden neben dem eigentlichen Suchbegriff zusätzlich ähnliche Funktionsverben gesucht. Die gefunden Funktionen werden mit einer dem Ähnlichkeitsmaß des Funktionsverbs entsprechenden Wahrscheinlichkeit in der LMWB ausgegeben. Die Verwendung der Synonymontologie ist optional und kann vom Anwender aktiviert bzw. deaktiviert werden (Bild 4-42).

Bild 4-42: Erweiterte Suche der LMWB

[57] Hierfür wurde der plattformunabhängige Ontologie-Editor Protégé genutzt.

Neben der Suche nach einer zu erfüllenden Funktion können auch mehrere Funktionen gleichzeitig gesucht werden. Das Ergebnis kann in einem morphologischen Kasten ausgegeben werden. Hierzu wurde die dritte Suchfunktion implementiert, die **Suche nach Merkmalen**. Diese wird in einer eigenen Suchmaske geöffnet und der Anwender kann Merkmale auflisten, die mit denen die Lösungsmuster abgeglichen werden. Die Merkmale der Lösungsmuster werden hierzu im Mechatronic Modeller spezifiziert. Die LMWB kann auf diese Spezifikation direkt zugreifen und die in den Lösungsmustern hinterlegten Merkmale mit den geforderten Merkmalen vergleichen.

Analysieren und Anwenden

Werden durch die Suche mögliche Lösungsmuster gefunden, muss der Anwender noch entscheiden, ob und wie diese anzuwenden sind. Hierbei unterstützt ihn die Möglichkeit Querverweise zwischen den Partialmodellen im Mechatronic Modeller zu definieren (vgl. Bild 4-39). Auf diese Weise kann der Anwender, das Zusammenwirken der Lösungsmuster Schritt für Schritt nachvollziehen. Da auch die Anwendungsbeispiele des Kontexts im Mechatronic Modeller abgebildet werden, können so Schnittstellen der Lösungsmuster in ähnlichen Aufgabenstellungen übernommen werden.

Im Hinblick auf die Entwicklung der kognitiven Informationsverarbeitung wurde eine übergreifende Verfahrensauswertung implementiert. Sie erlaubt es, auf einen Blick alle kognitionsrelevanten Verfahren hinsichtlich ihrer Anwendung im Selbstoptimierungsprozess zu untersuchen.

Für die Integration der Lösungsmuster in die bestehende Systemspezifikation müssen die entsprechenden Partialmodelle im Mechatronic Modeller zusammengesetzt werden. Die dafür notwendige Export- und Import-Funktionalität war nicht Bestandteil dieser Arbeit.

5 Validierung der Entwicklungssystematik

In diesem Kapitel wird die im vorgehenden Kapitel vorgestellte *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* validiert. Anhand eines intelligenten hybriden Energiespeichersystems soll dargestellt werden, wie die Entwicklungssystematik in der Anwendung funktioniert und welche die wesentlichen Ergebnisse sind. Die Entwicklung des Energiespeichersystems wird dabei in Kapitel 5.1 retrospektiv betrachtet. Kapitel 5.2 bewertet auf Basis des Anwendungsbeispiels die erarbeitete Entwicklungssystematik an den Anforderungen der Problemanalyse.

5.1 Anwendungsbeispiel: Hybrides Energiespeichersystem

Ein wichtiger Demonstrator des SFB 614 ist das RailCab-System der Neuen Bahntechnik Paderborn. Es ist ein fortgeschrittenes mechatronisches System, dessen intelligentes Verhalten auf dem Wirkparadigma der Selbstoptimierung beruht (vgl. Kap. 2.2.3). Ein RailCab besteht wiederum aus mehreren intelligenten Modulen, wie z.B. dem Antriebsmodul inkl. aktiver Luftspaltverstellung, der aktiven Spurführung und dem aktiven Feder-Neigesystem. Ein elektrisches Bordnetz versorgt die Module mit Leistung. Die Realisierung des Antriebsmoduls, ein doppelt gespeister Linearmotor, der asynchron betrieben wird, ermöglicht eine berührungslose Leistungsübertragung aus der Strecke auf das System. Eine Oberleitung oder eine Stromschiene sind nicht notwendig. Allerdings ist eine ausreichende Leistungsübertragung nicht in jeder Fahrsituation möglich, da sie vom Arbeitspunkt des Antriebsmoduls abhängt. Die maximale Leistungsübertragung ist in weiten Bereichen in etwa proportional zur gestellten Antriebskraft. Das Antriebsmodul kann folglich den Momentan-Leistungsbedarf nicht vollständig decken, insbesondere bei geringen Geschwindigkeiten oder im Stillstand [Pot05, S. 80].

Aus diesem Grund muss das RailCab-Fahrzeug mit einem eigenen Energiespeichersystem ausgestattet sein, das den Leistungsbedarf deckt, der über die durch das Antriebsmodul übertragende Leistung hinaus geht. Nur so kann die Versorgung und somit der Betrieb aller Verbraucher im RailCab sichergestellt werden. Die beiden wesentlichen Anforderungen an ein solches Energiespeichersystem sind:

- **Hohe Speicherkapazität:** Das System soll in der Lage sein, eine große Menge an Energie zu speichern, um einen möglichst langen Betrieb zu ermöglichen.
- **Hohe Leistung:** Beim Laden als auch beim Entladen soll ein hoher Energiedurchsatz ermöglicht werden, um alle Leistungsanfragen der Verbraucher zu bedienen und die maximale übertragene Leistung des Antriebsmoduls aufzunehmen.

Hinzu kommen weitere nicht funktionale Anforderungen. So sollte das System möglichst leicht und klein sein, da es auf dem Fahrzeug installiert wird. Geringe Kosten, ein

hoher Wirkungsgrad und lange Wartungsintervalle runden die Anforderungsspezifikation ab [RBW+09], [DGR09], [ADG+09].

Verfügbare Technologien werden diesen Anforderungen nicht gerecht. Konventionelle Akkumulatoren z.B. auf Basis von Nickel-Metallhydrid (NiMH) speichern zwar große Mengen an Energie, besitzen aber nur eine geringe spezifische Leistung. Zusätzlich ist die Anzahl an Ladezyklen gering. Doppelschichtkondensatoren (DLC: Double Layer Capacitor) können bei hohen Leistungen nahezu unendlich oft geladen und entladen werden, bieten aber nur eine geringe Speicherkapazität.

Um eine hohe Speicherkapazität und Leistung zu erfüllen, kombiniert daher das Energiespeichersystem des RailCabs die beiden genannten Speichertechnologien. Das **hybride Energiespeichersystem** besitzt sowohl Nickel-Metallhybrid-Akkumulatoren, deren Aufgabe die langfristige Energiespeicherung ist, als auch Doppelschichtkondensatoren zur Pufferung von Leistungsspitzen. Bild 5-1 zeigt die Struktur des Systems.

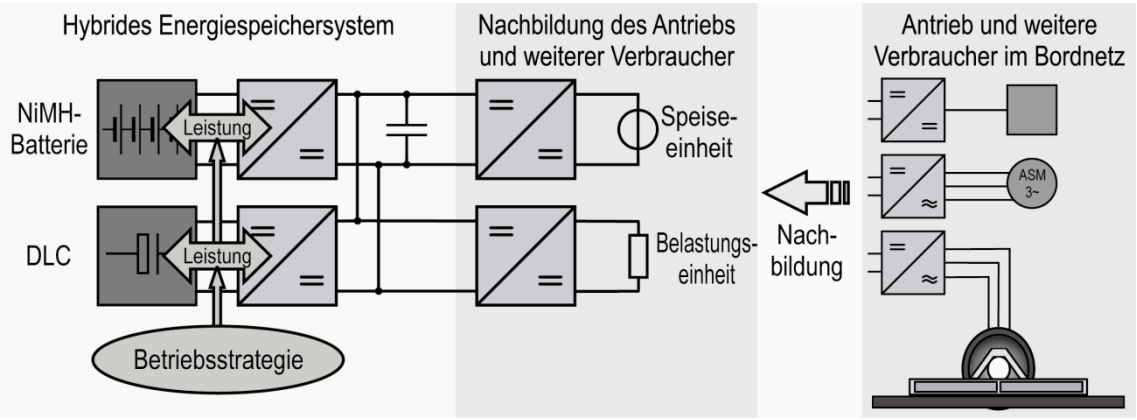


Bild 5-1: Struktur des hybriden Energiespeichersystems [ADG+09, S. 76]

Die im hybriden Energiespeichersystem zu speichernde Energie bzw. zu weitergebende Leistung resultiert aus der Leistungsbilanz des elektrischen Bordnetzes. Diese resultiert aus der durch das Antriebsmodul übertragene Leistung und der benötigten Leistung der Verbraucher. Hierzu ist ein zentrales Energiemanagement für das RailCab notwendig. Dieses vernetzt die verschiedenen Funktionsmodule und errechnet Planungsdaten für die Leistungsverteilung im Gesamtfahrzeug. Im Folgenden werden die vier Phasen und die wesentlichen Ergebnisse der Entwicklungssystematik zur Integration kognitiver Funktionen anhand der Konzipierung des hybriden Energiespeichersystems gezeigt.

5.1.1 Phase 1: Systemanalyse

Die Struktur des hybriden Energiespeichersystems aus Bild 5-1 sowie die bereits beschriebenen Anforderungen bilden den Ausgangspunkt. Zunächst wird die **Methode zur Zielabhängigkeitsanalyse** durchgeführt, die mit der Ermittlung zielrelevanter Einflussausprägungen beginnt. Da die Aufgabe des hybriden Energiespeichersystems eine effiziente Energiespeicherung als auch die optimale Leistungsaufteilung ist, ergeben

sich drei Ziele: *Energieverluste minimieren*, *Batterieschädigung minimieren* und *Leistungsreserve maximieren*. Mittels einer Umfeldanalyse können potentielle Einflüsse auf das System identifiziert werden (Bild 5-2).

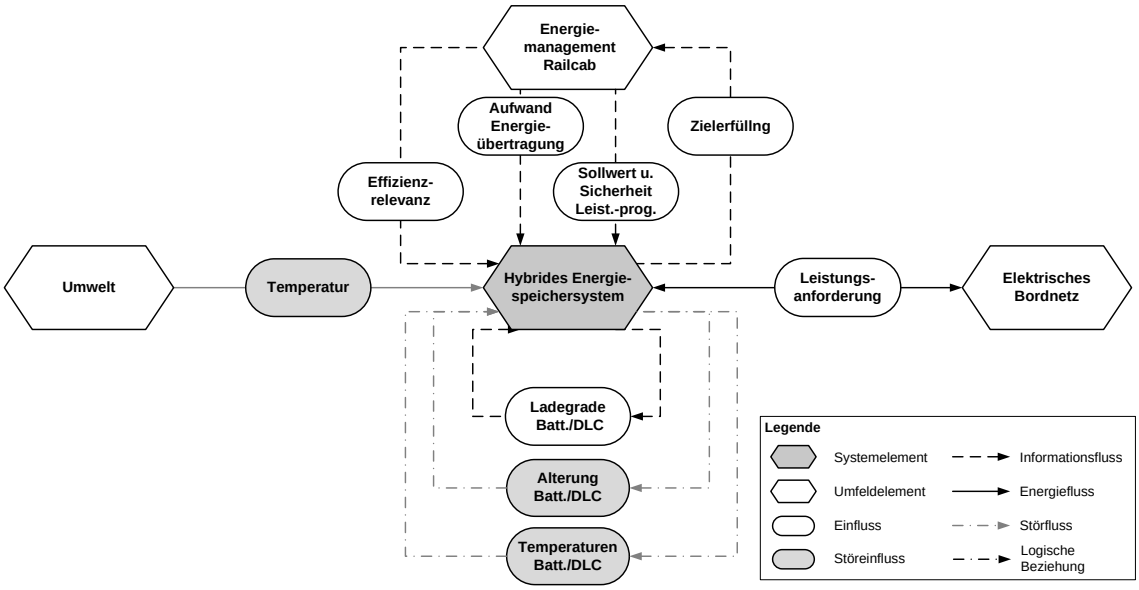


Bild 5-2: Umfeld des hybriden Energiespeichersystems

Sechs der Einflüsse haben einen direkten Einfluss auf die Systemziele, zu denen jeweils drei qualitative Ausprägungen definiert werden. In einem weiteren Schritt wird bewertet, wie sich die Einflussausprägungen auf die jeweilige Priorität der Systemziele auswirken. Eine erhöhte Zielpriorität kann zu einer Erhöhung der entsprechenden Zielgewichtung führen. Die Ergebnisse der Bewertung werden in der Zielprioritätsmatrix zusammengetragen (vgl. Bild 5-3).

Ausschlüsse der erkannten Einflussausprägungen existieren, außer zwischen denen eines Einflusses, nicht. So ergeben sich insgesamt 13 qualitative Einflussausprägungen, die zu 972 möglichen Situationen kombiniert werden können.

Das Zusammenzählen der Bewertung der Zielprioritätsmatrix für die gebildeten Situationen zeigt, dass zwar in 758 Situationen immer genau ein Ziel die höchste Priorität hat, dieses aber je nach Situation stark wechselt. Das Ziel *Energieverluste minimieren* ist in 287 Situationen dominant, das Ziel *Batterieschädigung minimieren* in 302 Situationen und das Ziel *Leistungsreserve maximieren* in 169 Situationen. Zudem ergibt sich für alle drei Ziele ein beachtlicher Anteil geteilter Zielpriorität (Ziel 1: 33%, Ziel 2: 35%, Ziel 3: 46%), was zusätzlich die Relevanz einer situationsgerechten und flexiblen Zielgewichtung zur Laufzeit bestärkt. Insbesondere die Minimierung der Leistungsreserve ist stark abhängig von der aktuellen Situation und den Prioritäten der anderen Ziele, so dass deren Gewichtung nicht fest vordefiniert werden sollte.

Zielprioritätsmatrix

Fragestellung:
"Wie wirkt sich die Einflussausprägung i (Zeile) auf die Zielprorität der Systemziele j (Spalte) aus?"

Bewertungsmaßstab:
-- = starke Verringerung der Zielpriorität
- = Veringerung der Zielpriorität
0 = keine Veränderung der Zielpriorität
+ = Erhöhung der Zielpriorität
++ = starke Erhöhung der Zielpriorität

		Systemziele	Energieverluste minimieren	Batterieschädigung minimieren	Leistungsreserve maximieren
Einflüsse	**Ausprägungen (qualitativ)**	**Nr.**	**1**	**2**	**3**
Sicherheit der Leistungsprognose	hoch	1	0	0	-
	mittel	2	0	0	0
	gering	3	0	0	++
Prognostizierte Leistungsanforderung	gering	4	0	0	-
	mittel	5	0	0	0
	hoch	6	+	0	+
Batterietemperatur	T > 45°C	7	+	++	0
	25°C < T < 45°C	8	0	0	0
	T < 25°C	9	0	+	0
Ladegrad Batterie	SOC > 0,8	10	-	+	+
	0,2 < SOC < 0,8	11	0	0	0
	SOC < 0,2	12	++	+	+
Aufwand der Energieübertragung	gering	13	--	+	0
	normal	14	0	0	0
	hoch	15	++	-	0
Ladegrad Kondensator	SOC > 0,8	16	0	0	+
	0,4 < SOC < 0,8	17	0	0	-
	0,25 < SOC < 0,4	18	+	0	+

Bild 5-3: Zielprioritätsmatrix des hybriden Energiespeichersystems

Aus der Zielabhängigkeitsanalyse ergibt sich ein Potential für den Einsatz einer flexiblen Leistungsverteilung in und aus den Energiespeichern zur Laufzeit. Die Struktur des hybriden Energiespeichersystems führt dabei zu einem Freiheitsgrad für die Aufteilung des Leistungsflusses auf die beiden Energiespeicher. Um von dieser Struktur zu profitieren ist eine anpassungsfähige und situationsabhängige Betriebsstrategie notwendig (vgl. Bild 5-1). Diese soll das System in die Lage versetzen, auf unterschiedliche Einflüsse aus dem Systemumfeld des RailCabs zu reagieren und sein Verhalten entsprechend anzupassen. Konventionelle Betriebsstrategien sind hierbei nicht ausreichend, da diese nur eine Anpassung an statische Regeln ermöglichen und keine Änderungen aufgrund der Systemumgebung bzw. Situation unterstützten [LL07], [Hei06].

Aus diesem Grund wird die Integration kognitiver Funktionen benötigt, die eine selbstoptimierende Berechnung der Leistungsverteilung der beiden Energiespeicher realisiert. Als Optimierungsvariable kann der aktuelle Batteriestrom zu diskreten Zeitpunkten gewählt werden. Bild 5-4 zeigt das vorläufige Zielsystem des hybriden Energiespeichersystems, für das eine kognitive Informationsverarbeitung zu konzipieren ist.

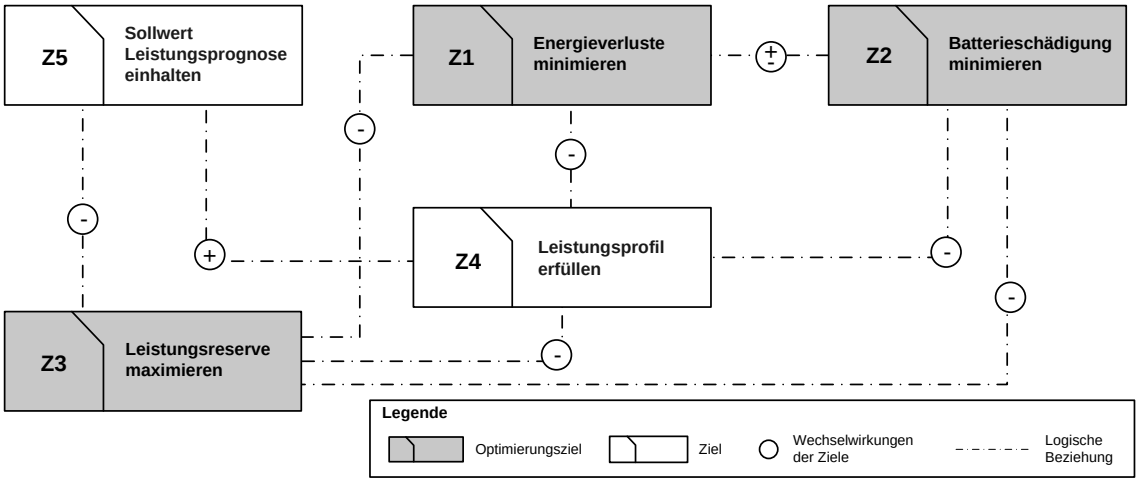


Bild 5-4: Zielsystem des hybriden Energiespeichersystems

5.1.2 Phase 2: Funktionssynthese

Um eine flexible und selbstoptimierende Betriebsstrategie zu realisieren, muss deren Aufbau und Ablauf zunächst weiter geklärt werden. Zu diesem Zweck erfolgt eine funktionale Beschreibung der informationsverarbeitenden Prozesse. Der **Funktionsverbkatalog der Informationsverarbeitung** unterstützt den Entwickler bei der Identifikation und Festlegung der wesentlichen Funktionen, die notwendig sind, um die beschriebenen Anforderungen zu erfüllen und das Nutzenpotential kognitiver Funktionen zu spezifizieren. Ferner kann auf Basis der **Entwurfsschablone für die Funktionshierarchie** des OCM systematisch eine strukturierte, aber weitestgehend lösungsneutrale Spezifikation der Gesamtfunktionalität erfolgen. In Bild 5-5, Bild 5-6 und Bild 5-7 sind die Funktionshierarchien der drei OCM Ebenen dargestellt. Gemeinsam beschreiben sie die Informationsverarbeitung im hybriden Energiespeichersystem. Auf Basis dieser Funktionen kann in der folgenden Phase die Suche nach geeigneten Lösungen beginnen.

Der *Controller* übernimmt die Regelung der Energiespeicher. Hierzu werden notwendige Messwerte erfasst und einige für den reflektorischen Operator aufbereitet. So werden die Ladegrade (SOC: State of Charge) und der Zustand (SOH: State of Health) der beiden Energiespeicher bestimmt. Die physikalischen Funktionen des *Grundsystems* sowie die *Benutzerschnittstelle* sind nur fakultativ aufgeführt (vgl. Bild 5-5).

Der *reflektorische Operator* soll die Kommunikation zum Energiemanagement des RailCabs ermöglichen und entsprechende Daten für den kognitiven Operator bereitstellen. Die wesentliche Aufgabe ist die Sollwertgenerierung für den Controller auf Basis der Optimierung (vgl. Bild 5-6).

Im *kognitiven Operator* soll die Situationsabhängigkeit des hybriden Energiespeichersystems realisiert werden. Hierzu sind detaillierte Analysen der wichtigsten Informationen notwendig. Die Betriebsstrategie soll einen optimierten Batteriestrom bzw. ein entsprechendes Stromprofil vorgeben (vgl. Bild 5-7).

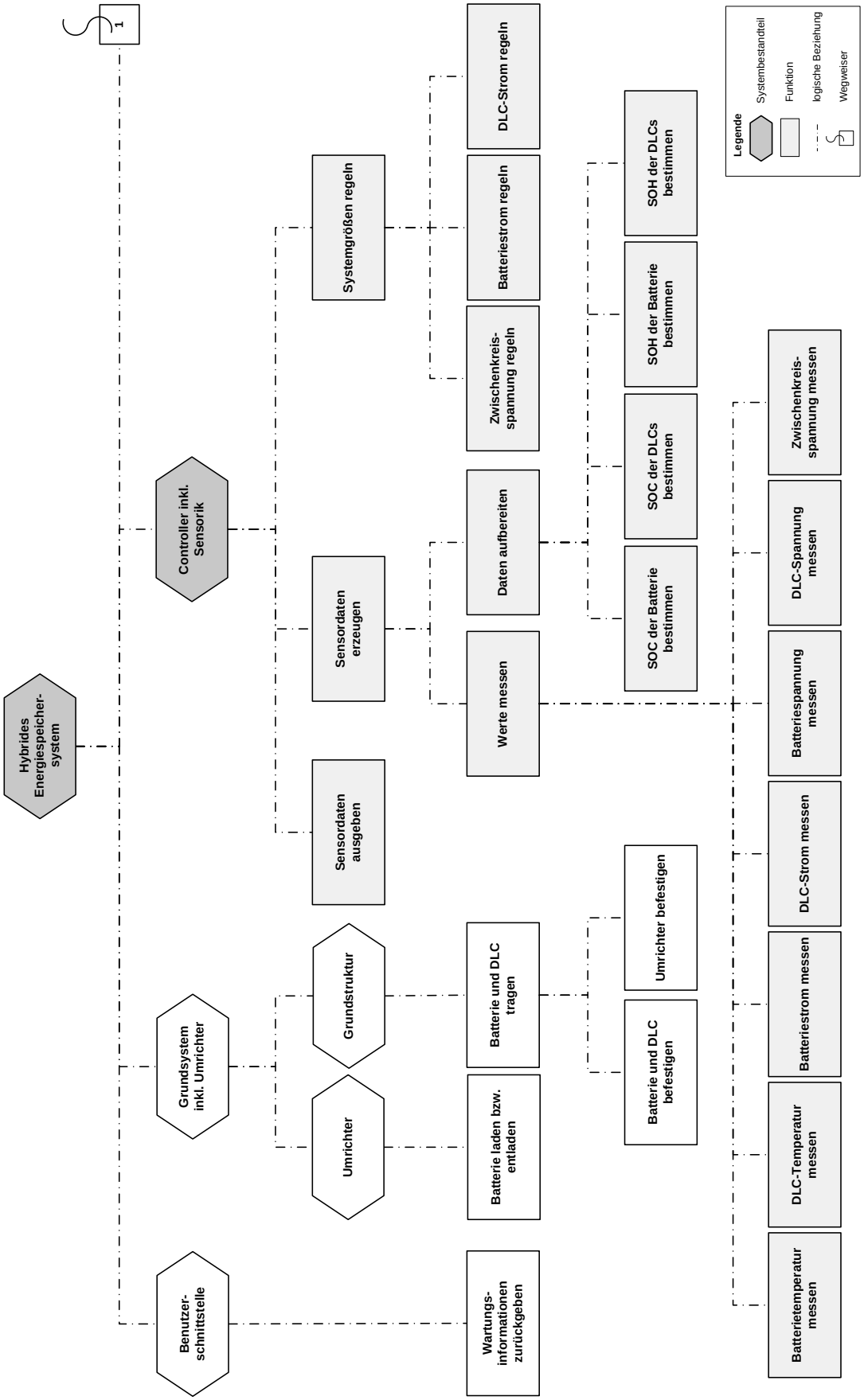


Bild 5-5: Funktionshierarchie des Controllers

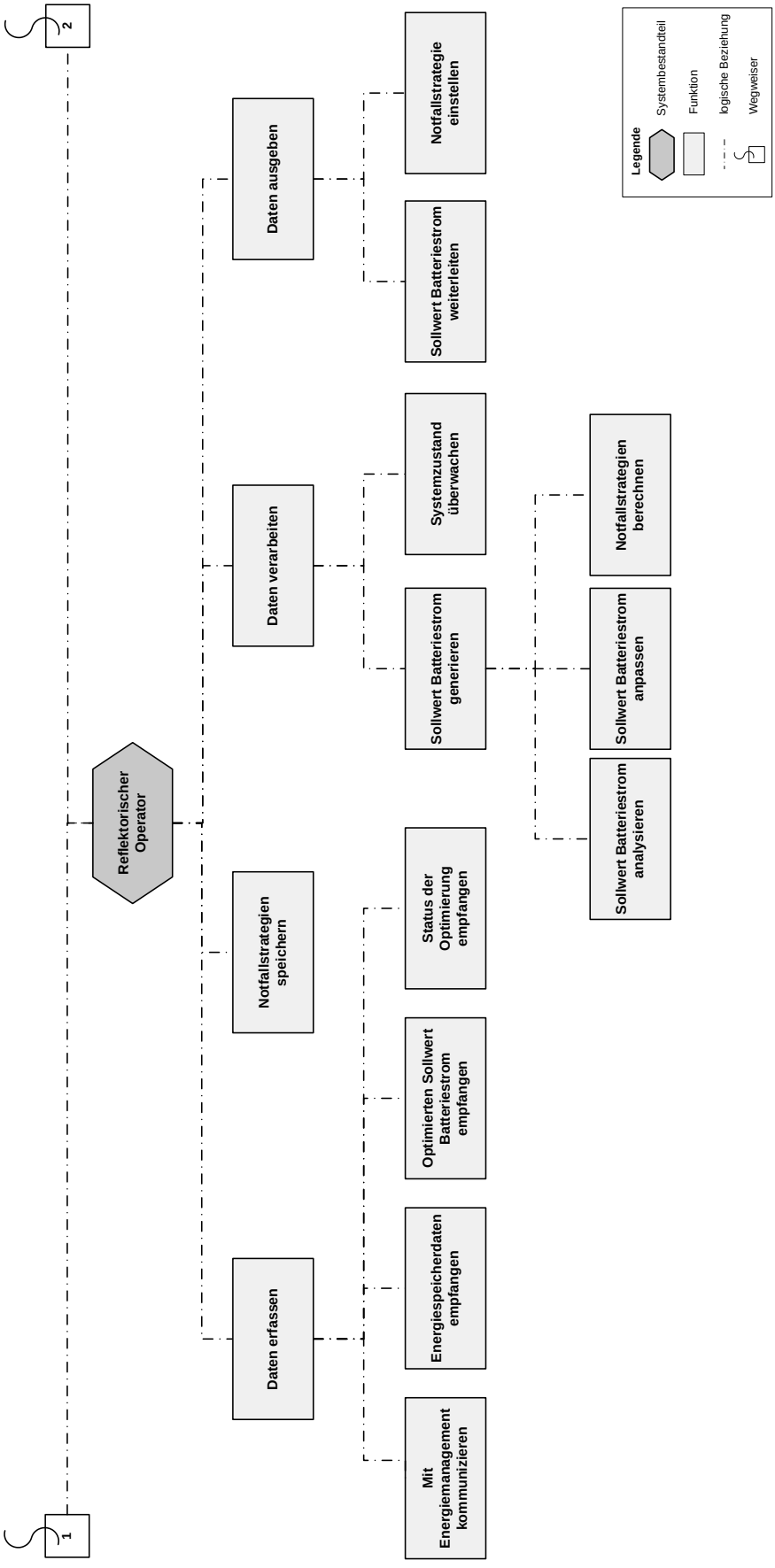


Bild 5-6: Funktionshierarchie des reflektorischen Operators

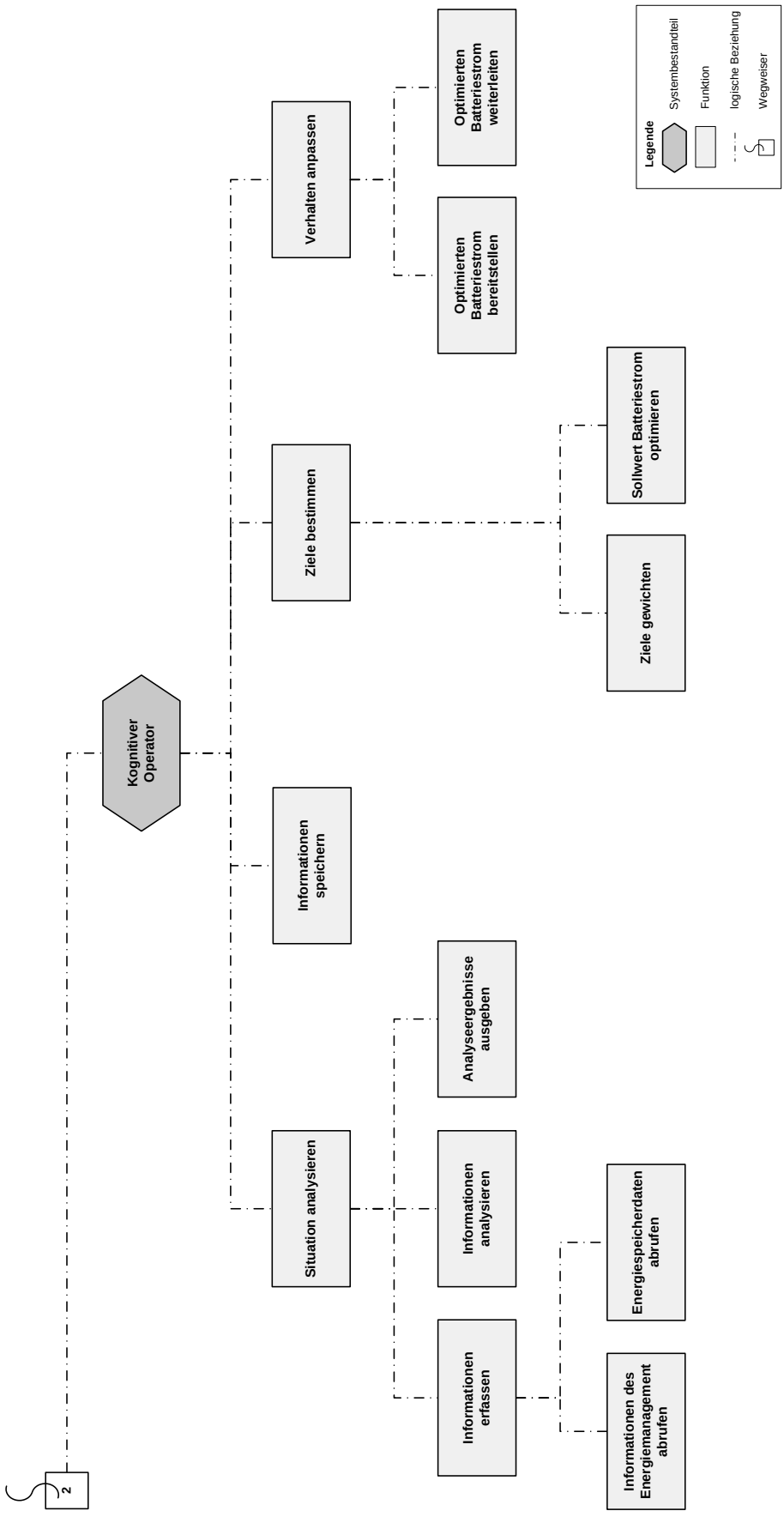


Bild 5-7: Funktionshierarchie des kognitiven Operators

5.1.3 Phase 3: Lösungsauswahl

Das Ergebnis der Funktionssynthese ist eine weitestgehend lösungsneutrale Funktionshierarchie. Für diese sind unter Berücksichtigung der Anforderungen und weiteren Randbedingungen erste Lösungsmöglichkeiten zu suchen. Aufgrund der Komplexität fortgeschrittener mechatronischer Systeme, können neue Lösungen nur mit großem Aufwand und unter Zuhilfenahme teils exklusiven Expertenwissens entdeckt werden. Daher sollten die erarbeiteten **Lösungsmuster für fortgeschrittene mechatronische Systeme** sowie die **Klassifikation kognitionsrelevanter Verfahren** eingesetzt werden, um den Lösungsentwurf initial zu konkretisieren. Beide Hilfsmittel werden durch die Lösungsmusterwissensbasis unterstützt (vgl. Kap. 4.6.2).

Aus der Systemanalyse und der Funktionssynthese ergibt sich, dass eine Optimierung hinsichtlich des Batteriestroms Kern der selbstoptimierenden Betriebsstrategie für das hybride Energiespeichersystem sein sollte. Die Verteilung der Leistungsflüsse auf die Energiespeicher kann prinzipiell durch ein mathematisches Optimierungsmodell gelöst werden, dabei können kontinuierliche als auch diskrete Optimierungsverfahren eingesetzt werden (vgl. Kap. 4.5.3). Hierzu wurden im Rahmen des SFB 614 bislang zwei Lösungsmuster für den kognitiven Operator (LM_{KO}) dokumentiert, die aus vorangegangen Systementwicklungen abgeleitet wurden.

Das LM_{KO} „Mehrzieloptimierung“ basiert auf dem Verfahren der Mehrzieloptimierung bei kontinuierlichen Optimierungsproblemen und der Berechnung einzelner Lösungen. Aufgenommen wurde das Lösungsmuster nach der Implementierung einer selbstoptimierenden Arbeitspunksteuerung (SOAPS) für das Antriebsmodul des RailCabs. Für das SOAPS wurde bereits eine geeignete Mehrzieloptimierung entwickelt [ADG+09], [WSD+08]. Im Anhang befinden sich die wesentlichen Aspekte des Lösungsmusters (vgl. A2.6). Bild 5-8 zeigt die Verfahrensbeschreibung der Mehrzieloptimierung, wie sie als Lösungsprinzip auch in der Lösungsmusterwissensbasis abgelegt ist.

Das LM_{KO} „Intelligente Vorausschau“ unterstützt ebenfalls die Optimierung mehrerer Ziele, aber für diskrete Optimierungsaufgaben. Das wesentliche Lösungsprinzip des Lösungsmusters ist die Suchbaumanalyse zur intelligenten Planung von Systemreaktionen auf noch nicht eingetretene Situationen. Das Lösungsmuster wurde nach der Entwicklung einer selbstoptimierenden Ressourcenplanung für ein echtzeitfähiges Betriebssystem abgeleitet und dokumentiert [GZO+08]. Das LM_{KO} „Intelligente Vorausschau“ wird im Anhang gezeigt (vgl. A2.3).

Beide Lösungsmuster wurden ebenfalls in die Lösungsmusterwissensbasis abgelegt. Diese unterstützt den Entwickler bei der Identifikation und schnellen Einarbeitung in die Lösungsmuster sowie der zugrundeliegenden Verfahren (vgl. Kap. 4.6.2).

Für das hybride Energiespeichersystem wurden beide Lösungsmuster für die Umsetzung der kognitiven Informationsverarbeitung ausgewählt. Für bekannte und oft befahrene Strecken soll eine Offline-Optimierung implementiert werden, die auf einer konti-

nuierlichen Auswertung der Optimierungsvariablen beruht. Für diese Strecken soll das hybride Energiespeichersystem auf bereits errechnete Optimierungsergebnisse in Form von Paretomengen für die Leistungsprofile zurückgreifen, wie es das LM_{KO} „Mehrzieloptimierung“ beschreibt. Die Paretomengen können in einer Datenbank für den Betrieb abgelegt werden [RBW+09].

Da nicht garantiert werden kann, dass ein benötigtes Leistungsprofil für die Leistungsverteilung bereits vorhanden ist, wird eine zweite, online ablaufende Optimierung für die Berechnung der Betriebsstrategie angestrebt. Diese soll mit dem LM_{KO} „Intelligente Vorausschau“ realisiert werden.

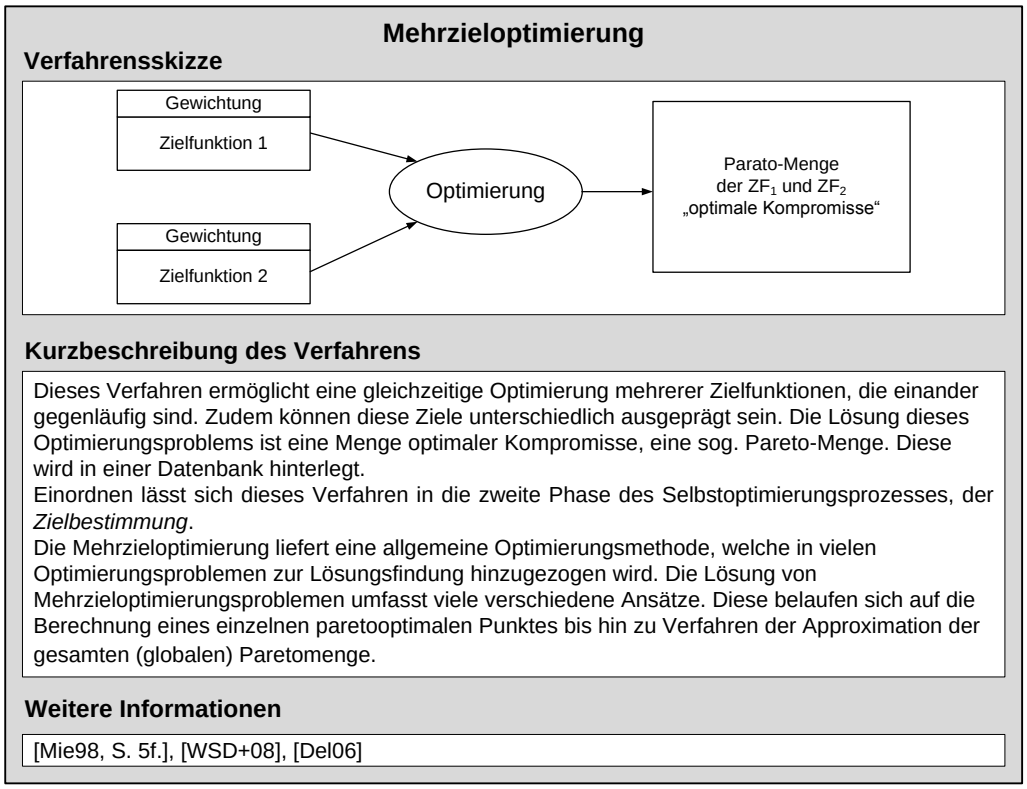


Bild 5-8: Kurzdarstellung des Verfahrens „Mehrzieloptimierung“

5.1.4 Phase 4: Systemspezifikation

In der letzten Phase werden die ausgewählten Lösungsmuster genutzt, um die Spezifikation des OCM im Kontext des Gesamtsystems zu erstellen. Zu diesem Zweck sollte die Funktionshierarchie weiter konkretisiert werden. Bild 5-9 zeigt das Partialmodell Funktionen des kognitiven Operators nach Berücksichtigung der Funktionen der beiden LM_{KO} „Mehrzieloptimierung“ (vgl. Bild A-24) und „Intelligente Vorausschau“ (vgl. Bild A-14). Im Vergleich zur lösungsneutralen Funktionshierarchie sind insbesondere die für die kognitive Informationsverarbeitung wesentlichen Funktionen *Situation analysieren* und *Ziele bestimmen* erweitert worden.

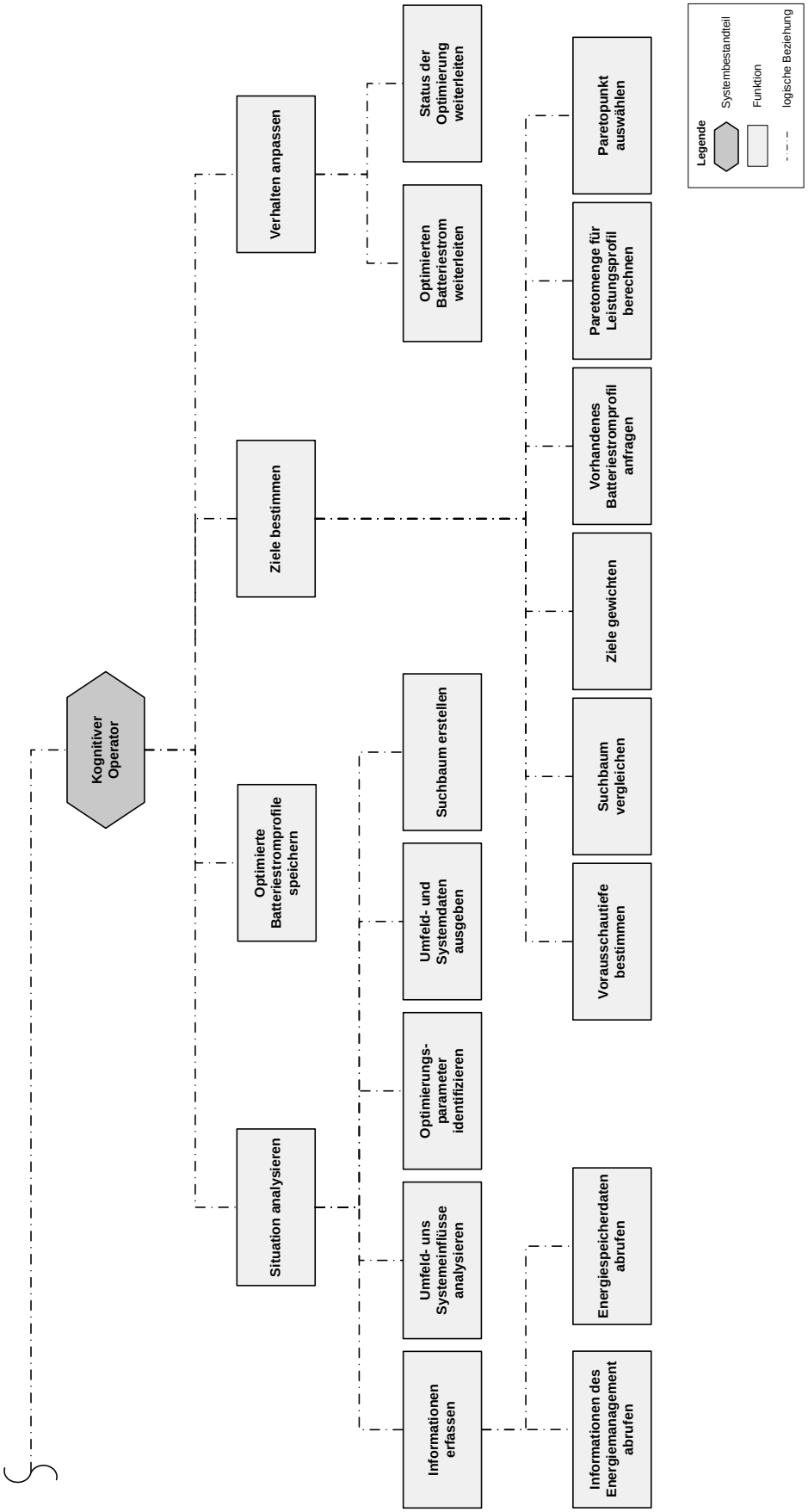


Bild 5-9: Konkretisierte Funktionshierarchie des kognitiven Operators

Im Anschluss wird die Wirkstruktur des OCM sukzessive erstellt. Auf Basis der **Entwurfsschablone für die Wirkstruktur** fortgeschrittener mechatronischer Systeme können die verschiedenen Ebenen des OCM detailliert werden. Dabei sollten die Schnittstellen innerhalb des entsprechenden Aspekts der verwendeten Lösungsmuster beachtet werden, um die Interaktionen der gewählten Systemelemente eindeutig zu spezifizieren. Die in Kapitel 4.4.2 vorgestellten **Systemelemente der Informationsverarbeitung** unterstützen den Entwickler bei der eindeutigen Benennung der Systemelemente. Bild 5-10 zeigt die Wirkstruktur des OCM für das zu entwickelnde hybride Energiespeichersystem, das über kognitive Funktionen für eine flexible Betriebsstrategie verfügt.

Bild 5-10: Wirkstruktur des OCM für das hybride Energiespeichersystem

Das **Grundsystem** besteht im Wesentlichen aus den beiden Energiespeichern. Da eine einzelne Batterie und ein einzelner DLC nicht ausreichend Leistung aufnehmen können, werden diese hier als *Batterie-Pack* bzw. *DLC-Pack* aus mehreren Elementen bestehen.

Der *Zwischenkreis* puffert die Leistung in den Schaltvorgängen zwischen den beiden *bidirektionalen Gleichstromstellern*. Er ist die Schnittstelle zum *elektrischen Bordnetz* des RailCabs und somit zum Antriebsmodul und den Verbrauchern.

Der **Controller** umfasst die *Batteriestromregelung*, die *DLC-Stromregelung* und den *Spannungsregler* für die Zwischenkreisspannung. Die Regelung erfolgt auf Basis der Momentanwerte, dem Sollwert der Zwischenkreisspannung (u^*_{ZK}) und dem vorgegeben Sollwert des Batteriestroms (i^*_{Bat}). Die DLC-Stromregelung erfolgt auf Basis des Spannungsreglers für die Zwischenkreisspannung Die Leistungsbilanz wird auf diese Weise ausgeglichen. Der *Messsignalaufbereiter* erzeugt die Ladegrade und Zustände der Energiespeicher für die spätere Optimierung.

Im **reflektorischen Operator** erfolgt die Generierung des Batteriesollstroms für die Batteriestromregelung. Zu diesem Zweck wird durch ein Simulationsmodell der Energiespeicher der optimierte Batteriestrom aus dem kognitiven Operator hinsichtlich Parameterfehler korrigiert. Das ist die Aufgabe des *Korrekturelements*. Um die Fälle abzusichern, dass die Optimierung nicht rechtzeitig Werte liefert, sich die Bordnetzleistung entgegen der Prognose verhält oder die Optimierung Fehler enthält, wird der optimierte Batteriestrom durch die *Überwachung* kontinuierlich geprüft. Das *Kommunikationsmodul* ist die externe Schnittstelle zum Energiemanagement des RailCabs. Ferner verteilt es die Informationen zwischen den drei Ebenen des OCM.

Der **kognitive Operator** umfasst die beiden Optimierer. Das Systemelement *kontinuierliche MZO* (Mehrzieloptimierung) wird durch die Mustergruppe des LM_{KO} „Mehrzieloptimierung“ realisiert (vgl. Bild A-25). Die kontinuierliche MZO wird der Übersicht halber nicht weiter detailliert dargestellt. Das Systemelement *diskrete MZO* ist Kern des LM_{KO} „Intelligente Vorausschau“. Die *Datenbank* legt Leistungsprofile für den Batteriestrom ab, die durch die kontinuierliche MZO auf Basis eines Abgleichs zwischen Simulationsmodell des Optimierers und den Messwerten berechnet wurden. So können Stromprofile für wiederkehrende Leistungsprofile direkt übernommen werden. Dazu ist eine *Ablaufsteuerung* notwendig, die anfragt, ob ein entsprechendes Leistungsprofil vorhanden ist. Falls nicht, wird die diskrete Optimierung gestartet, da diese im Gegensatz zur kontinuierlichen MZO auch online ablaufen kann. Von entscheidender Bedeutung ist die *Umfeldanalyse*. Sie erhält alle Informationen vom Energiemanagement als auch vom Messsignalaufbereiter, um Umfeld- und Systemdaten zu bilden und zu analysieren. Hierfür werden die Ziele bewertet bzw. ausgewählt, deren Umsetzung dann durch die Optimierer erfolgt.

Das Zusammenspiel der Systemelemente um die selbstoptimierende Betriebsstrategie umzusetzen wird im Partialmodell Verhalten – Aktivitäten beschrieben. Dieses Modell beschreibt, in welcher Reihenfolge die Funktionen zur Laufzeit ausgeführt werden sollen. Die in Bild 5-11 abgebildeten Aktivitäten spezifizieren den gesamten Ablauf der kognitiven Informationsverarbeitung bestehend aus der kontinuierlichen und diskreten Optimierung.

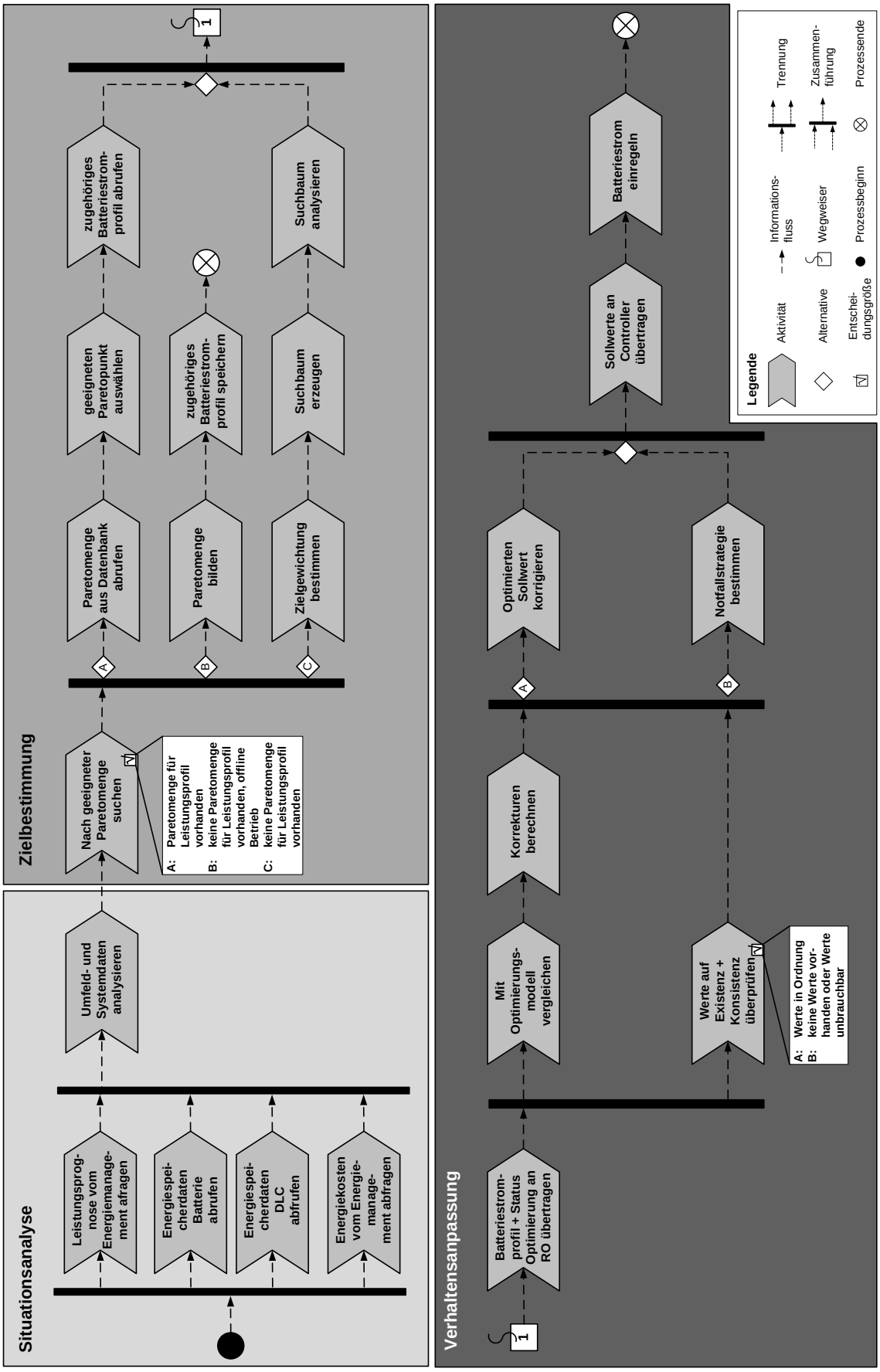


Bild 5-11: Verhalten – Aktivitäten der selbstoptimierenden Betriebsstrategie

Die Aktivitäten der *Situationsanalyse* werden wesentlich durch das Umfeldanalyseelement ausgeführt (vgl. Bild 5-10). Infolge dieser Ergebnisse entscheidet die Ablaufsteuerung, wie die Optimierung und somit die *Zielbestimmung* auszuführen ist. Wurde bereits eine geeignete Paretomenge durch die kontinuierliche MZO berechnet, wird ein Stromprofil aus der Datenbank abgerufen (A). Falls keine geeignete Paretomenge vorhanden ist, wird die Suchbaumanalyse durch die diskrete MZO durchgeführt (C). Befindet sich das System im Stand by oder in einem Offline-Modus, bildet die kontinuierliche MZO auf Basis der vergangen Fahrten neue Paretomengen für zukünftige Fahrten und speichert diese in der Datenbank (B). Die *Verhaltensanpassung* wird nach der Übergabe der Optimierungsergebnisse durch den reflektorischen Operator und den Controller realisiert. Das Korrekturelement vergleicht die Optimierungswerte mit den Simulationsmodellen der Energiespeicher und berechnet entsprechende Korrekturen. Die Überwachung prüft die Werte des kognitiven Operators und entscheidet dann, ob das Korrekturelement (A) oder eine Notfallstrategie (B) die Sollwerte für den Controller vorgibt. Dieser regelt dann die Leistungsaufteilung auf die Energiespeicher über den Batteriestrom.

Wegen der beiden unterschiedlichen Optimierungsverfahren interessiert in erster Linie, wann welche Optimierung aktiv bzw. inaktiv ist und welche Ereignisse Zustandsübergänge auslösen. Insgesamt wurden fünf unterschiedliche Zustände identifiziert, die die Kombination der kontinuierlichen und der diskreten Optimierung charakterisieren. In Bild 5-12 sind die Verhalten – Zustände der Optimierung im kognitiven Operator des hybriden Energiespeichersystems dargestellt.

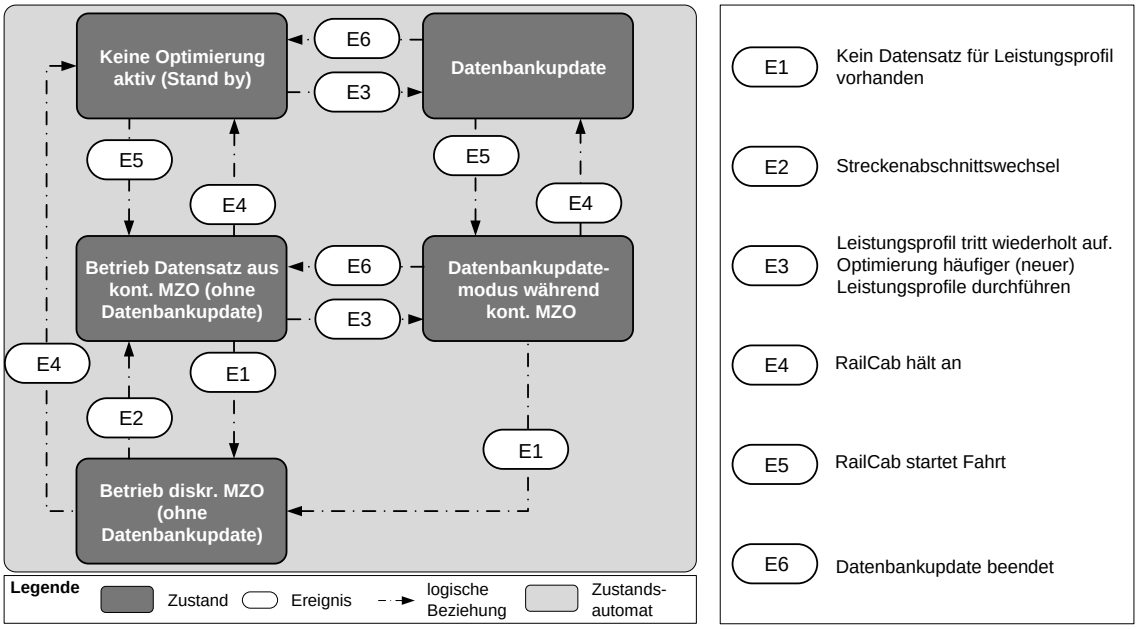


Bild 5-12: Verhalten – Zustände der Optimierung

Die Betriebsstrategie soll nicht nur das Systemverhalten verbessern, sondern auch so ausgelegt sein, dass potentielle Notfälle intelligent abgefangen werden. Diese müssen

im Rahmen der Systemspezifikation durch den Entwickler vorausgedacht werden. Bild 5-13 zeigt mögliche Notzustände und die Ereignisse, die diese auslösen können.

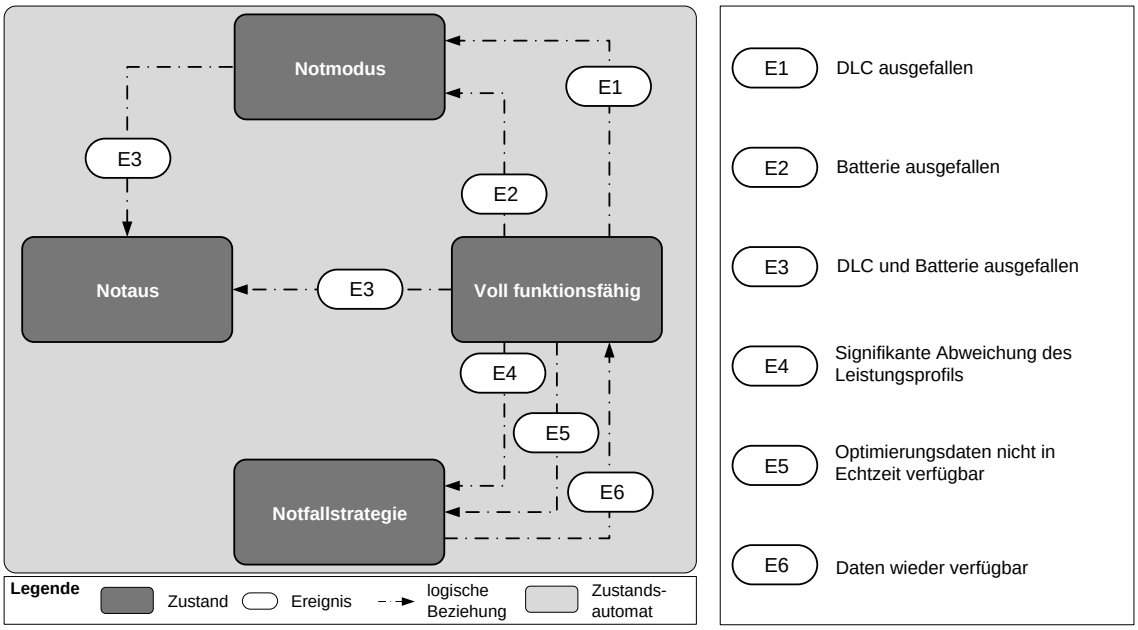


Bild 5-13: Verhalten – Zustände des Gesamtsystems hinsichtlich möglicher Notzustände

5.1.5 Implementierung und Umsetzung

Die Entwicklungssystematik zur Integration kognitiver Funktionen endet mit der Spezifikation des OCM im Rahmen der Prinziplösung. In den weiteren Entwicklungsphasen wird das hybride Energiespeichersystem konkretisiert, der OCM implementiert und der beschriebene Selbstoptimierungsprozess mit detaillierten Systemmodellen realisiert[58]. Zu diesem Zweck wurde im Fachgebiet Leistungselektronik und Elektrische Antriebstechnik (LEA) der Universität Paderborn von ROMAUS et al. ein skalierter Prüfstand aufgebaut. Mit diesem können die Systemmodelle verifiziert und die Betriebsstrategie validiert werden (vgl. Bild 5-14) [RBW+09].

Der Prüfstand besteht neben den beiden Energiespeichern aus einem Netzteil und einer Last, die das Antriebsmodul und die Verbraucher im RailCab-Bordnetz nachbilden. Die Kopplung erfolgt durch bidirektionale Leistungsumrichter wie es in der Prinziplösung festgelegt ist. Der reflektorische Operator und Controller wurden auf einem Echtzeitrechensystem implementiert – der kognitive Operator zeitlich entkoppelt auf einem leistungsstarken PC-System.

Eine erste Evaluierung im Vergleich zu Betriebsstrategien, die mit festen Zielen arbeiten, bestätigt den identifizierten Nutzen durch die Integration kognitiver Funktionen. So

[58] Die an die Prinziplösung anschließenden Entwicklungstätigkeiten waren nicht Bestandteil der Arbeit des Autors.

kann die selbstoptimierende Betriebsstrategie die Energieverluste um bis zu 13% reduzieren. Ferner kann die Verfügbarkeit der maximalen Leistung signifikant gesteigert werden, indem aktuelle Umfeld- und Systemeinflüsse im Zielsystem zur Laufzeit berücksichtigt werden [RBW+09], [DGR09].

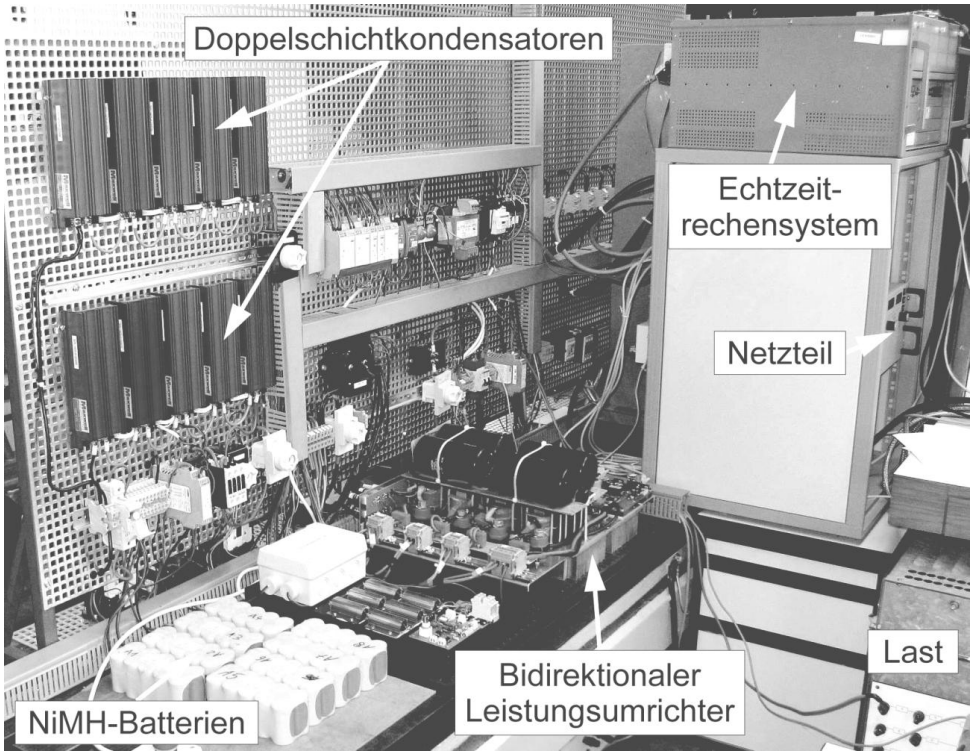


Bild 5-14: Prüfstand für das hybride Energiespeichersystem [ADG+09, S. 77]

5.2 Bewertung der Arbeit an den Anforderungen

Abschließend wird die erarbeitete *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* anhand der neun identifizierten Anforderungen aus Kapitel 2.6 bewertet. Zu diesem Zweck wird für jede Anforderung knapp erläutert, inwiefern sie durch die Entwicklungssystematik erfüllt wird. Bild 5-15 zeigt die Erfüllung der Anforderungen durch die jeweiligen Bestandteile der Entwicklungssystematik im Überblick.

A1) Interdisziplinarität: Die Basis der erarbeiteten Entwicklungssystematik ist eine gemeinsame *Begriffswelt der Mechatronik und der Kognitionswissenschaft* (vgl. Kap. 4.2). Diese wird von allen Bestandteilen der Entwicklungssystematik aufgegriffen und weiter konkretisiert. Folglich werden Techniken und Methoden aller an der Integration kognitiver Funktionen beteiligten Disziplinen integriert. Ferner ist die Entwicklungssystematik derart gestaltet, dass sie zwar hinsichtlich ihrer Anwendbarkeit abgeschlossen ist, aber weiterhin die disziplinunabhängige Integration neuer Ansätze und Verfahren zur Entwicklung fortgeschrittener mechatronischer Systeme erlaubt.

A2) Systematische Vorgehensweise: Das *Vorgehensmodell* der Entwicklungssystematik ermöglicht eine systematische Integration kognitiver Funktionen in den frühen Phasen des Entwurf (vgl. Kap. 4.3). Hierzu unterteilt es sich in vier wesentliche Phasen. Das Vorgehensmodell beschreibt die durchzuführenden Tätigkeiten und empfiehlt notwendige Hilfsmittel. Ferner wurden die Phasen des Vorgehensmodells in einen etablierten Konzipierungsprozess für fortgeschrittene mechatronische Systeme integriert (vgl. Kap. 4.3.2). Die Anwendbarkeit des Vorgehensmodells – und somit der gesamten Entwicklungssystematik – wurde anhand der Konzipierung einer intelligenten Betriebsstrategie für das hybride Energiespeichersystems des RailCabs belegt (vgl. Kap. 5.1).

A3) Orientierung am Systementwurf: In der Produktentwicklung wird ausgehend von den Anforderungen zunächst eine Funktionshierarchie des zu entwickelnden Systems erstellt. In einem nächsten Schritt werden für die Teilfunktionen der untersten Hierarchieebene mögliche Teillösungen gesucht, welche die jeweilige Funktion erfüllen. Diese Vorgehensweise greift das *Vorgehensmodell* der erarbeiteten Entwicklungssystematik mit den Phasen *Funktionssynthese*, *Lösungsauswahl* und *Systemspezifikation* explizit auf (vgl. Kap. 4.3.1). Hierdurch fokussiert die Entwicklungssystematik den im Systementwurf entscheidenden Übergang von der Funktionshierarchie zur sog. Wirkstruktur, die in Verbindung mit einer Verhaltensbeschreibung die prinzipielle Wirkungsweise der Systemlösung beschreibt.

A4) Verständlichkeit: Die verständliche Beschreibung von zu entwickelnden kognitiven Funktionen ist eine Herausforderung. Informationsverarbeitende Systemprozesse werden in der klassischen Konstruktionslehre gänzlich vernachlässigt. Die Entwicklungssystematik definiert hierzu einheitliche *Funktionsverben* und *Systemelemente der Informationsverarbeitung* (vgl. Kap. 4.4.3 und 4.4.2). Diese versetzen den Entwickler in die Lage, eine eindeutige und nachvollziehbare Systemspezifikation eines fortgeschrittenen mechatronischen Systems zu erstellen, das über kognitive Funktionen verfügt. Die erarbeitete *einheitliche Spezifikation von Lösungsmustern* für fortgeschrittene mechatronische Systeme erlaubt die Überführung von fachspezifischem Lösungswissen, das zur Implementierung und Umsetzung kognitiver Funktionen notwendig ist. So werden komplexe Methoden und Verfahren in einer prägnanten und auch für Nicht-Experten verständlichen Form dokumentiert (vgl. Kap. 4.5.1.1).

A5) Ganzheitliche Spezifikation der Informationsverarbeitung: Die Informationsverarbeitung wird mit der *Spezifikationstechnik des SFB 614* beschrieben. Ihre Anwendung wurde dementsprechend angepasst (vgl. Kap. 4.4.1). Die strukturelle Beschreibung greift die OCM-Architektur auf und ergänzt diese um kognitionswissenschaftliche Aspekte. Ausgangspunkt sollte das Dreischichtenmodell nach STRUBE sein. Diese Anforderung wurde erfüllt. Sowohl für die Erstellung der Wirkstruktur als auch der Funktionshierarchie des OCM wurden entsprechende *Entwurfsschablonen* erarbeitet, die eine ganzheitliche Spezifikation der Informationsverarbeitung unterstützen (vgl. Kap. 4.4.2 und 4.4.3)

A6) Technische Relevanz: Die abstrakten kognitiven Funktionen aus der Kognitionswissenschaft wurden schrittweise in kognitive Funktionen zur ingenieurwissenschaftlichen Systembeschreibung überführt. Zu diesem Zweck wurde das aus der Datentechnik anerkannte EVA-Prinzip aufgegriffen und in Anlehnung an die drei kognitiven Funktionen des menschlichen Nervensystems zur Beschreibung der Informationsverarbeitung technischer Systeme erweitert. Ergebnis ist ein angepasstes EVA-Prinzip, das aus *drei allgemeinen Funktionen der Informationsverarbeitung* besteht. Der erarbeitete *Funktionsverbkatalog der Informationsverarbeitung* konkretisiert das angepasste EVA-Prinzip (vgl. Kap. 4.4.3). Auf diese Weise wird die technische Relevanz der Systembeschreibung sichergestellt.

A7) Lösungswissen für den Entwurf der Informationsverarbeitung: Expertenwissen ist ein entscheidender Aspekt für die Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme. Die *einheitliche Spezifikation eines Lösungsmusters* ermöglicht die Dokumentation und Wiederverwendung von Expertenwissen (vgl. Kap. 4.5.1.1). Sie erfüllt die geforderten Kategorien nach ALEXANDER zur Problem- und Lösungsbeschreibung. Im Zuge dieser Arbeit wurden sieben Lösungsmuster für den kognitiven Operator aufgenommen, die die Anwendbarkeit hinsichtlich der kognitiven Informationsverarbeitung validieren und zukünftig als Lösungsmuster für kognitive Funktionen zur Verfügung stehen (vgl. Kap. 4.5.2.1 und Anhang A2). Desweiteren wurden *kognitionsrelevante Verfahren* klassifiziert und aufgenommen (vgl. Kap. 4.5.3 und Anhang A3).

A8) Auswahl und Kombination von Lösungswissen: Die *einheitliche Spezifikation von Lösungsmustern* für fortgeschrittene mechatronische Systeme eignet sich in erster Linie für Lösungen der Informationsverarbeitung, kann aber auch auf physikalische Teillösungen angewendet werden. Die ausgearbeiteten Lösungsmuster werden dieser Anforderung gerecht. Zu diesem Zweck wurden ein Lösungsmuster für den reflektorischen Operator (vgl. Kap. 4.5.2.2), ein Lösungsmuster für den Controller (vgl. Kap. 4.5.2.3) und eines für das Grundsystem aufgenommen (vgl. Anhang A2.1). Da die Lösungsmuster festdefinierte Teilbereiche des OCM erfüllen, wird eine Kombination auf Basis der spezifizierten Schnittstellen grundsätzlich unterstützt. Hinsichtlich der Auswahl umfasst die Lösungsmusterspezifikation die Aspekte Merkmale und Funktionen, die das Problem strukturiert und vergleichbar beschreiben (vgl. Kap. 4.5.1.1)

A9) Rechnerunterstützung durch eine Wissensbasis: Im Rahmen dieser Arbeit wurde ein *Konzept für ein rechnergestütztes Werkzeug* erstellt, das den Umgang mit Lösungswissen auf Basis eines wissensbasierten Systems beschreibt (vgl. Kap. 4.6.1). Das Konzept wurde mit der *Lösungsmusterwissensbasis (LMWB)* prototypisch umgesetzt (vgl. Kap. 4.6.2). Durch die Anbindung einer Funktionsverbenontologie und weiterer Werkzeuge, wie z.B. dem Mechatronic Modeller, ermöglicht die LMWB die rechnerinterne Dokumentation, die gezielte Suche und die systematische Anwendung aufgenommener Lösungsmuster. Zusätzlich wurden kognitionsrelevante Verfahren in ihr abgelegt.

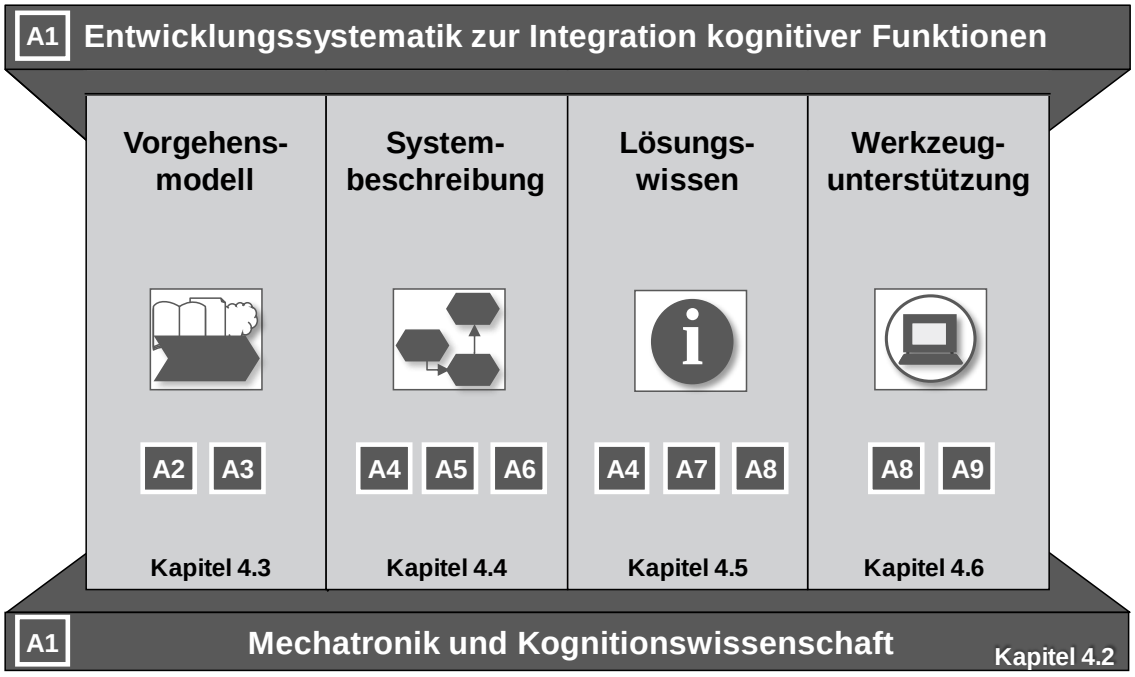


Bild 5-15: Erfüllung der Anforderungen durch die Entwicklungssystematik

Die vorgestellte *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* erfüllt damit alle Anforderungen in vollem Umfang. Sie ist geeignet, kognitive Funktionen im Rahmen der Konzipierung zu beschreiben und auf Basis von Lösungsmustern eine passende Systemspezifikation zu entwickeln. Sie wurde mit Erfolg am Anwendungsbeispiel *Hybrides Energiespeichersystem* validiert.

6 Zusammenfassung und Ausblick

Erfolgsversprechende Produktinnovationen für den modernen Maschinenbau ergeben sich aus dem engen Zusammenwirken von Mechanik, Elektrik/Elektronik, Regelungstechnik und Software. Der Begriff Mechatronik bringt dies zum Ausdruck. Zukünftige **Nutzenpotentiale** im Bereich der Mechatronik werden insbesondere durch die fortschreitende Entwicklung der Informations- und Kommunikationstechnik sowie durch neue Erkenntnisse und Methoden nicht-technischer Disziplinen, wie der Kognitionswissenschaft oder der Neurobiologie ermöglicht. Ziel sind *fortgeschrittene mechatronische Systeme*, die sich durch ihre Funktionalität effektiv in ihre Umgebung einbinden und in bestmöglicher Art und Weise ihren Systemzielen nachgehen. Derartige Systeme zeigen ein intelligentes sowie adaptives Verhalten und verfügen über eine erhöhte Flexibilität sowie Robustheit – auch im Umgang mit dem menschlichen Benutzer. Diese faszinierende Perspektive wird durch das Phänomen der Kognition und die damit einhergehende *Integration kognitiver Funktionen* wie Wahrnehmen, Verstehen, Problemlösen oder Kommunizieren charakterisiert.

Auf dem Weg zu derartigen fortgeschrittenen mechatronischen Systemen fehlt es an einer systematischen Verzahnung der für die Erforschung kognitiver Funktionen relevanten Disziplinen mit der ingenieurwissenschaftlichen Vorgehensweise in der Produktentwicklung. Die beiden wesentlichen **Herausforderungen** sind die verstärkte Interdisziplinarität und die steigende Komplexität der zu entwickelnden Systeme. Beide führen zu einer erhöhten Komplexität der Entwicklung. Aus diesem Grund wurden bislang nur wenige technische Anwendungen realisiert. Bei der Mehrzahl handelt es sich um prototypische Entwicklungen aus dem Bereich der Robotik.

Um diesen Herausforderungen zu begegnen, muss die Integration kognitiver Funktionen bereits in den frühen Phasen der Entwicklung methodisch unterstützt werden. Es werden ein systematisches Vorgehen und dedizierte Hilfsmittel zur Identifikation und Spezifikation kognitiver Funktionen für fortgeschrittene mechatronische Systeme benötigt. Von zentraler Bedeutung ist hierbei die Dokumentation und Wiederverwendung von erfolgreich eingesetztem Lösungs- und Expertenwissen für die Implementierung und Umsetzung kognitiver Funktionen.

Im Rahmen dieser Arbeit wurden existierende Ansätze zur methodischen Entwicklung kognitiver Funktionen, Methoden zur Beschreibung einer kognitiven Informationsverarbeitung und Techniken zur Wiederverwendung von Lösungswissen untersucht. Die betrachteten Ansätze liefern wenn aber nur eine partielle Unterstützung bei der Integration kognitiver Funktionen in technische Systeme. Eine ganzheitliche Systematik existiert nicht. Das *Framework für kognitive Produkte*, der *Entwurf anpassungsfähiger Produkte* oder die *Entwicklung kognitiver technischer Systeme nach* PAETZOLD beschreiben eher generische und abstrakte Ansätze; die Systemspezifikation wird überhaupt nicht berücksichtigt. Bestehende *Methoden der Funktionsbeschreibung* definieren zwar die

einzuhaltenden Rahmenbedingungen für die Beschreibung technischer Funktionen im Systementwurf, informationsverarbeitende oder gar kognitive Funktionen werden nicht unterstützt. Von den analysierten *Strukturkonzepten für eine kognitive Informationsverarbeitung* erfüllt insbesondere das *Operator-Controller-Modul* wichtige Anforderungen. Das gleiche gilt für die *Spezifikationstechnik des SFB 614*. Sie ist als *Sprache zur Beschreibung der Informationsverarbeitung* geeignet. Bestehende *Muster zur Beschreibung wiederkehrender Lösungen* adressieren nicht die Integration kognitiver Funktionen sowie entsprechender Lösungen und bieten größtenteils nur eine textuelle Dokumentation ohne wiederverwendbare Systemmodelle. Die beiden untersuchten *IT-Konzepte für den Umgang mit Lösungswissen* umfassen nur allgemeine Ansätze, die stets eine anwendungsspezifische Ausarbeitung benötigen. Aus diesen Gründen besteht ein **Handlungsbedarf** für eine ganzheitliche *Entwicklungssystematik,* die die methodische Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme in den frühen Phasen des Entwurfs unterstützt.

Die erarbeitete Entwicklungssystematik greift einige der untersuchten Ansätze auf und ergänzt diese um neu entwickelte Hilfsmittel. Grundlage ist die Definition einer gemeinsamen Begriffswelt der Mechatronik und der Kognitionswissenschaft. Das **Ergebnis** ist eine Entwicklungssystematik, die im Kern vier übergeordnete Bestandteile umfasst:

- ein strukturiertes **Vorgehensmodell**, das die durchzuführenden Tätigkeiten in vier zentrale Phasen einteilt und deren Resultate sowie notwendige Hilfsmittel, wie z.B. Methoden oder Werkzeuge, integriert,
- eine Technik zur **Systembeschreibung**, die auf Basis der Spezifikationstechnik des SFB 614 eine dedizierte Sprache und generische Entwurfsschablonen für eine intuitive und disziplinübergreifende Spezifikation der zu entwickelnden kognitiven Informationsverarbeitung bereitstellt,
- wiederverwendbares **Lösungswissen** auf Basis einer einheitlichen Spezifikation von Lösungsmustern für fortgeschrittene mechatronische Systeme um relevante Aspekte einer Aufgabe und ihrer Lösung auch für Dritte bereitzustellen und
- ein Konzept zur **Werkzeugunterstützung** für das rechnergestützte Dokumentieren, Suchen sowie Analysieren und Anwenden von Lösungswissen und dessen prototypische Implementierung in Form der sog. Lösungsmusterwissensbasis.

Die **Validierung** der Entwicklungssystematik erfolgte anhand der Konzipierung eines intelligenten hybriden Energiespeichersystems für das autonome Schienenfahrzeug RailCab. Hierzu wurden das Vorgehensmodell der Entwicklungssystematik vollständig durchlaufen, die neu entwickelten Hilfsmittel angewendet und die so erstellten Resultate vorgestellt. Ferner wurde die Anwendbarkeit der entwickelten Lösungsmusterspezifikation durch die Aufnahme von insgesamt zehn Lösungsmustern belegt. Somit erfüllt die

erarbeitete *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme* die aufgestellten Anforderungen im vollen Umfang.

Für die Integration kognitiver Funktionen in technische Systeme besteht **weiterer Forschungsbedarf**. Zukünftige Arbeiten müssen die Verzahnung der beteiligten Disziplinen weiter vorantreiben. Es werden die Implementierungsphase und somit die kognitionsrelevanten Verfahren stärker in den Vordergrund rücken. Für die Verfahren wurde im Rahmen dieser Arbeit eine erste Klassifikation aufgestellt. Es wurden nur Verfahren aufgenommen, die in den dokumentierten Lösungsmustern benötigt werden. Eine Methode zur Identifikation und Auswahl von Verfahren für die Implementierung der Informationsverarbeitung fortgeschrittener mechatronischer Systeme könnte hier großen Nutzen stiften.

Übergeordnetes Ziel zukünftiger Forschungsarbeiten ist eine neue Schule des Entwurfs intelligenter technischer Systeme. Sie muss dem Trend der zunehmenden Bedeutung von Informationsverarbeitung in mechatronischen Systemen und deren innovativen Lösungen gerecht werden. Diese Schule ist Gegenstand des Sonderforschungsbereichs 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) der Universität Paderborn, in dessen Rahmen die vorliegende Arbeit entstanden ist.

7 Abkürzungsverzeichnis

bzgl.	bezüglich
bspw.	beispielsweise
bzw.	beziehungsweise
CAD	Computer Aided Design
CO	Controller
elektr.	elektrisch
engl.	englisch
ggf.	gegebenenfalls
GoF	Gang of Four
GS	Grundsystem
inkl.	inklusive
IV	Informationsverarbeitung
Kap.	Kapitel
KI	künstliche Intelligenz
KO	Kognitiver Operator
LM	Lösungsmuster
LM_{CO}	Lösungsmuster für den Controller
LM_{GS}	Lösungsmuster für das Grundsystem
LM_{IV}	Lösungsmuster für die Informationsverarbeitung
LM_{KO}	Lösungsmuster für den kognitiven Operator
LM_{RO}	Lösungsmuster für den reflektorischen Operator
mind.	mindestens
OCM	Operator-Controller-Modul
PM	Partialmodell
RO	Reflektorischer Operator
SA	Situationsanalyse
SE	Systemelement

SFB	Sonderforschungsbereich
SO	Selbstoptimierung
sog.	sogenannte
u.a.	unter Anderem
VA	Verhaltensanpassung
vgl.	vergleiche
ZB	Zielbestimmung
z.B.	zum Beispiel

8 Literaturverzeichnis

Publikationen

[ABB+04] ANDERSON, J. R.; BOTHEL, D.; BYRNE, M. D.; DOUGLASS, S.; LEBIERE, C.; QIN, Y.: An Integrated Theory of the Mind. In: Psychological Review, Volume 111, Nr. 4, American Psychological Association, 2004, pp. 1036-1060

[ABM+05] ALBERS, A.; BURKARDT, N.; MEBOLDT, M.; SAAK, M: SPALTEN Problem Solving Methodology in Product Development. In: Proceedings of the 15th International Conference on Engineering Design (ICED2005), August 15-18, Melbourne, Australia, 2005

[ADG+09] ADELT, P.; DONOTH, J.; GAUSEMEIER, J.; GEISLER, J.; HENKLER, S.; KAHL, S.; KLÖPPER, B.; KRUPP, A.; MÜNCH, E.; OBERTHÜR, S.; PAIZ, C.; PORRMANN, M.; RADKOWSKI, R.; ROMAUS, C.; SCHMIDT, A.; SCHULZ, B.; VÖCKING, H.; WITKOWSKI, U.; WITTING, K.; ZNAMENSHCHYKOV, O.: Selbstoptimierende Systeme des Maschinenbaus – Definition, Anwendungen, Konzepte. HNI-Verlagsschriftenreihe, Band 234, 2009

[ADT09] ALBERS, A.; DEIGENDESCH, T.; TURKI, T: Design Patterns in Microtechnology. In: Proceedings of the 17th International Conference on Engineering Design (ICED2009), August 24-27, Stanford, USA, 2009

[AH07] ALCIATORE, D. G.; HILSTAND, M. B.: Introduction to Mechatronics and Measurement Systems. McGraw-Hill, New York, 3. Auflage, 2007

[AIS+77] ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M.; JACOBSON, M.; FIKSDAHLKING, I.; ANGEL, S.: A Pattern Language – Towns, Buildings, Construction. Oxford University Press, 1st Edition, 1977

[AIS+95] ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M.; JACOBSON, M.; FIKSDAHLKING, I.; ANGEL, S.; CZECH, H.(Hrsg.): Eine Muster-Sprache – Städte, Gebäude, Konstruktion. Löcker Verlag, Wien, 1995

[AK09] ADELT, P.; KLÖPPER, B.: Building Blocks und Prototypical Implementation of a Hybrid Planning Architecture. In Klöpper, B.; Dangelmaier, W. (Eds.): Self-x in Engineering. September 15-19, Paderborn, MV-Verlag, Münster, 2009

[AKH+10] ADELT, P.; KLEINJOHANN, B.; HERBRECHTSMEIER, S.; RÜCKERT, U.: Demonstrating self-optimization using a heterogeneous robot group. In: Proceedings of 8th IEEE International Conference on Industrial Informatics (INDIN2010), July 13-16, Osaka, Japan, 2010

[AL98] ANDERSON, J. R.; LEBIERE, C.: Atomic Componets of Thought. Lawrence Erlbaum Associates, Hillsdale, 1998

[And01] ANDERSON, J. R.: Kognitive Psychologie. Spektrum Akademischer Verlag, Heidelberg, 3. Auflage, 2001

[And97] ANDERSON, J. R.: Rules of the Mind. Lawrence Erlbaum Associates, Hillsdale, 1993

[AW95] ASTRÖM, K. J.; WITTENMARK, B.: Adaptive Control. Addison-Wesley, 2. Auflage, 1995

[Bac07] BACH, J.: Principles of Synthetic Intelligence – Building Blocks for an Architecture of Motivated Cognition. Dissertation, Fachbereich Humanwissenschaften, Universität Osnabrück, 2007

[Bal01] BALZERT, H.: Lehrbuch der Software-Technik – Software-Entwicklung (Band 1). Spektrum-Verlag, Heidelberg, 2. Auflage, 2001

[Bal05] BALZERT, H.: Lehrbuch Grundlagen der Informatik – Konzepte und Notationen in UML 2, Java 5, C# und C++ – Algorithmik und Software-Technik – Anwendungen. Spektrum-Verlag, Heidelberg, 2. Auflage, 2005

[BBW08] BEETZ, M.; BUSS, M.; WOLLHERR, D.: Cognitive Technical Systems – What is the Role of Articicial Intelligence? In: KI 2007: Advances in Artificial Intelligence, Lecture Notes in Computer Science 2997, Volume 4667/2007, Springer, Berlin, 2007, pp. 19-42

[BD95] BRÖHL, A.-P.; DRÖSCHEL, W. (Hrsg.): Das V-Modell – Der Standard für die Softwareentwicklung mit Praxisleitfaden . Oldenbourg Verlag, München, 1995

[BE01] BOURNE, L. E.; EKSTRAND, B. R.: Einführung in die Psychologie. Verlag Dietmar Klotz, Eschborn bei Frankfurt am Main, 3. Auflage, 2001

[BF97] BLUM, A. L.; FURST, M.: Fast planning through planning graph analysis. In: Artificial Intelligence, Volume 90, 1997, pp. 279-298

[Bir80] BIRKHOFER, H.: Analyse und Synthese der Funktionen technischer Produkte. VDI Fortschritt-Berichte, Reihe 1, Nr. 70, VDI Verlag, Düsseldorf, 1980

[Bis09] BISCHOF, N.: Psychologie – Ein Grundkurs für Anspruchsvolle. Kohlhammer, Stuttgart, 2. Auflage, 2009

[BK06] BEIERLE, C.; KERN-ISBERNER, G.: Methoden wissensbasierter Systeme: Grundlagen – Algorithmen – Anwendungen. Vieweg Verlag, Berlin, 2006

[BLL04] BENJAMIN, P.; LYONS, D.; LONSDALE, D.: ADAPT – A Cognitive Architecture for Robotics. In: Proceedings of the International Conference of Cognitive Modeling (ICCM2004), July 30-August 1, Pittsburgh, USA, 2004

[BMS+05] BURGHART, C.; MIKUT, R.; STIEFELHAGEN, R.; ASFOUR, T.; HOLZAPFEL, H.; STEINHAUS, P.; DILLMANN, R.; A Cognitive Architecture for a Humanoid Robot – A First Approach. In: Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids2005), December 5-7, Tsukuba, Japan, 2005

[Bor94] BORMANN, S.: Virtuelle Realität: Genese und Evaluation. Addison-Wesley, Bonn, 1994

[Bos94] BOSSEL, H.: Modellbildung und Simulation – Konzepte, Verfahren, und Modelle. Vieweg Verlag, Braunschweig, 1994

[Bra02] BRACHMAN, R. J.: Systems That Know What They`re Doing. In. IEEE Intelligent Systems, Volume 17, Nr. 6, November/December, 2002, pp. 67-71

[Bro02] BROOKS, R.: Menschmaschinen – Wie uns die Zukunftstechnologien neu erschaffen. Campus Verlag, Frankfurt am Main, 2002

[Bro03] BROCKHAUS: Brockhaus – Naturwissenschaft und Technik. Bibliographisches Institut & F.A. Brockhaus AG, Mannheim, 2003

[BS06] BIRBAUMER, N.; SCHMIDT, R.F.: Kognitive Funktionen und Denken. In Schmidt, F.; Schaible, H.-G. (Hrsg.): Neuro- und Sinnesphysiologie, Springer Medizin Verlag, Heidelberg, 5. Auflage, 2007, S. 449-465

[BSK+06] BÖCKER. J.; SCHULZ, B.; KNOKE, T.; FRÖHLEKE, N.: Self-Optimization as a Framework for Advanced Control Systems. In: 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON06), November 7-10 2006, Paris, France, 2006

[BW10] BIUNDO, S.; WENDEMUTH, A.: Von kognitiven technischen Systemen zu Companion-Systemen. In: KI – Künstliche Intelligenz, Volume 24, Springer Verlag, Berlin/Heidelberg, 2010

[CAT08] CHMARRA, M. K.; ARTS, L.; TOMIYAMA, T.: Towards Adaptable Architecture. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2008), August 3-6, New York, USA, 2008

[Chr99] CHRISTALLER, T.: Cognitive Robotics: a new approach to artificial intelligence. In: Artificial Life and Robotics, Volume 1, Nr. 3, 1999

[Com94] COMERFORD, R.: Mecha … what?. IEEE Spectrum, 31(8), 46-49, IEEE Press, Piscataway, USA, 1994

[CR98] CLAUSSEN, U.; RODENACKER, W. G.: Maschinensystematik und Konstruktionsmethodik – Grundlagen und Entwicklung moderner Methoden. Springer Verlag, Berlin/Heidelberg, 1998

[CS04] CHUNG, L.; SUBRAMANIAN, N.: Adaptable architecture generation for embedded systems. Journal of Systems and Software, Vol. 71, 2004, pp. 271-295

[Czi08] CZICHOS, H.: Mechatronik – Grundlagen und Anwendungen technischer Systeme. Vieweg+Teubner Verlag, Wiesbaden, 2. Auflage, 2008

[DB09] DANGELMAIER, W.; KLÖPPER, B.: Verhaltensplanung für mechatronische Systeme - Planung als Funktion in selbstoptimierenden Systemen. In: wt Werkstattstechnik online, März, 2009, S. 123-129

[Dei09] DEIGENDESCH, T.: Kreativität in der Produktentwicklung und Muster als methodisches Hilfsmittel. Dissertation, Fakultät für Maschinenbau, Karlsruher Institut für Technologie (KIT), Forschungsbericht Band 41, Institut für Produktentwicklung (IPEK), Karlsruhe, 2009

[Del06] DELL'AERE, A.: Multi-Objective Optimization in Self-Optimizing Systems. In: Proceedings of the IEEE 32nd Annual Conference on Industrial Electronics (IECON2006), Paris, 2006, pp. 4755-4760

[DGD+09] DUMITRESCU, R.; GAUSEMEIER, J.; DANGELMAIER, W.; KLÖPPER, B.: Solution Patterns for the Development of Self-Optimizing Systems. In Klöpper, B.; Dangelmaier, W. (Eds.): Self-x in Engineering. September 15-19, Paderborn, MV-Verlag, Münster, 2009

[DGR09] DUMITRESCU, R.; GAUSEMEIER, J; ROMAUS, C.: Towards the design of cognitive functions in self-optimizing Systems exemplified by a Hybrid Energy Storage System. In: Proceedings of the 10th International Workshop on Research and Education in Mechatronics (REM2009), September 10-11, Glasgow, UK, 2009

[DH02] DAENZER, W. F.; HUBER, F. (Hrsg.): Systems Engineering – Methodik und Praxis, Verlag Industrielle Organisation, Zürich, 11. Auflage, 2002

[DK10] DUMITRESCU, R.; KAHL, S.: DeeP View (Development Process Viewer) – a Tool for the Interactive Visualization of Product Development Processes. In: Proceedings of the 1st International Conference on Modelling and Management of Engineering Processes (MMEP2010), July 19-20, Cambridge, UK, 2010

[Dör01] DÖRNER, D.: Bauplan für eine Seele. Rowohlt Verlag, Reinbek bei Hamburg, 2001

[DSD01] DÖRNER, D.; SCHAUB. H., DETJE, F.: Das Leben von PSI – Über das Zusammenspiel von Kognition, Emotion und Motivation – oder: Eine einfache Theorie für komplizierte Verhaltensweisen. In: Sozionikaktuell, Nr. 2/2001, Hamburg, 2001

[Dud01] DUDEN: Informatik – ein Sachlexikon für Studium und Praxis. Dudenverlag, Mannheim, 3. Auflage, 2001

[Dud07a] DUDEN: Das große Fremdwörterbuch. Band 5. Dudenverlag, Mannheim, 2007

[Dud07b] DUDEN: Das Synonymwörterbuch – Ein Wörterbuch sinnverwandter Wörter. Band 8. Dudenverlag, Mannheim, 2007

[Dwo89] DWORATSCHEK, S.: Grundlagen der Datenverarbeitung. De Gruyter Verlag, Berlin, 8. Auflage, 1989

[EAS+08] ESAU, N.; ADELT, P.; SCHMIDT, A.; ROSE, M.: Interval Optimization in a Hybrid Planner Context for an Air Gap Adaptation System. In: Proceedings of 3rd Asia International Symposium on Mechatronics (AISM2008), August 27-31, Sapporo, Japan, 2008

[Ede00] EDELMANN, W.: Lernpsychologie. Beltz PVU, Weinheim, 5. Auflage, 2000

[EGH+08] EBERSBACH, A.; GLASER, M.; HEIGL, R.; WARTA, A.: Wiki – Kooperation im Web. Springer Verlag, Berlin, 2008

[Ehr07] EHRLENSPIEL, K.: Integrierte Produktentwicklung – Denkabläufe Methodeneinsatz Zusammenarbeit. Carl Hanser Verlag, München, 3. Auflage, 2007

[EM98] ELMQVIST, J. MATTSON, S. E.: An Overview of the Modeling Language Modelica. Eurosim`98, Simulation Congress, April 14-15, Helsinki, Finnland, 1998

[EM099] ELMQVIST, H.; MATTSON, S. E.; OTTER, M.: Modelica – A Language for Physical System Modeling, Visualization and Interaction. In: The 1999 IEEE Symposium on Computer-Aided Control System Design (CACSD99), August 22-29, Hawaii, IEEE Customer Service, Piscataway, 1999

[Eur09] EUROPEAN COMMISSION: FP7 Updated Work Programme 2009 and Work Programme 2010. Cooperation, Theme 3, ICT – Information and Communications Technologies, 2009

[FGK+04] FRANK, U.; GIESE, B.; KLEIN, F.; OBERSCHELP, O.; SCHMIDT, A.; SCHULZ, B.; VÖCKING, H.; WITTING, K.; GAUSEMEIER, J. (Hrsg.): Selbstoptimierende Systeme des Maschinenbaus – Definitionen und Konzepte. HNI-Verlagsschriftenreihe, Band 155, Paderborn, 2004

[FHV92] FRITZON, P.; HERBER, J.; VIKLUND, L.: The implementation of ObjectMath - a high-level programming environment for scientific computing. In Kastens, U.; Pfahler, P.: Compiler Construction – 4th International Conference (CC92), Volume 641 of Lecture Notes in Computer Science, Springer Verlag, Berlin, 1992, pp. 312-318

[For07] FORBIG, P.: Objektorientierte Softwareentwicklung mit UML. Carl Hanser Verlag, München, 3. Auflage, 2007

[Föl08] FÖLLINGER, O.: Regelungstechnik – Einführung in die Methoden und ihre Anwendung. Hüthig Verlag, Heidelberg, 2008

[Fra06] FRANK, U.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 175, Paderborn, 2006

[GBI09] GAUSEMEIER, J.; BRINK, V.; IHMELS, S.: Technologieorientiertes Innovationsmanagement mit der Innovations-Datenbank. In: Industrie-Management, Ausgabe 1/2009, GITO-Verlag, 2009, S. 40-44

[Gau10] GAUSEMEIER, J. (Hrsg.): Frühzeitige Zuverlässigkeitsanalyse mechatronischer Systeme. Carl Hanser Verlag, München, 2010

[GDK10] GAUSEMEIER, J.; DOROCIAK, R.; KAISER, L.: Computer-Aided Modeling of the Principle Solution of Mechatronic Systems – A Domain-Spanning Methodology for the Conceptual Design of Mechatronic Systems. In: Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2010), August 15-18, Montreal, Canada, 2010

[GEK01] GAUSEMEIER, J.; EBBESMEYER, P.; KALLMEYER, F.: Produktinnovation – Strategische Planung und Entwicklung der Produkte von morgen. Carl Hanser Verlag, München, 2001

[GF06] GAUSEMEIER, J.; FELDMANN, K.: Integrative Entwicklung räumlicher elektronischer Baugruppen. Carl Hanser Verlag, München, 2006

[GFD+08a] GAUSEMEIER, J.; FRANK, U.; DONOTH, J.; KAHL, S.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus (Teil1). Konstruktion, 7/8 – 2008, Fachaufsatz Mechatronik, VDI-Verlag, Berlin, 2008, S. 59-66

[GFD+08b] GAUSEMEIER, J.; FRANK, U.; DONOTH, J.; KAHL, S.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus (Teil2). Konstruktion, 9 – 2008, Fachaufsatz Mechatronik, VDI-Verlag, Berlin, 2008, S. 91-99

[GHH+08] GIESE, H.; HENKLER, S.; HIRSCH, M.; ROUBIN, V.; TICHY, M.: Modeling Techniques for Software-Intensive Systems. In: Tiako, P. F. (Ed.): Designing Software-Intensive Systems - Methods and Principles. IGI Global, Hershey, USA, 2008

[GHJ+04] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. RIEHLE, D. (Übersetzer): Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software. Addison-Wesley Verlag, München, 2004

[GHJ+94] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley Verlag, München, 1994

[GHK+06] GAUSEMEIER, J.; HAHN, A.; KESPOHL, H.-D.; SEIFERT, L.: Vernetzte Produktentwicklung – Der erfolgreiche Weg zum Global Engineering Networking. Carl Hanser Verlag, München, 2006

[GL00] GRABOWSKI, H.; LOSSACK, R.: The Axiomatic Approach in the Universal Design Theory. In: Proceedings of the First International Conference on Axiomatic Design (ICAD2000), June 21-23, Cambridge, USA, 2000

[GN03] GÖRZ, G.; NEBEL, B.: Künstliche Intelligenz. Fischer Taschenbuch Verlag, Frankfurt am Main, 2003

[GPW09] GAUSEMEIER, J.; PLASS, C.; WENZELMANN, C.: Zukunftsorientierte Unternehmensgestaltung Strategien, Geschäftsprozesse und IT-Systeme für die Produktion von morgen. Carl Hanser Verlag, München, 2009

[GRD+09] GEIGER, C.; RECKTER, H.; DUMITRESCU, R.; KAHL, S.; BERSSENBRÜGGE, J.: A Zoomable User Interface for Presenting Hierachical Diagrams on Large Screens. In Jacko, J. A. (Ed.): Lecture Notes in Computer Science, Volume 5611/2009, Human-Computer Interaction – Novel Interaction Methods and Techniques, Springer, Berlin/Heidelberg, 2009

[GRS03] GÖRZ, G.; ROLLINGER, C.-R.; SCHNEEBERGER, J. (Hrsg.): Handbuch der Künstlichen Intelligenz. Oldenbourg Wissenschaftsverlag, München, 4. Auflage, 2003

[Gru93] GRUBER, T. R.: A Translation Approach to Portable Ontology Specifications. In: Knowledge acquisition, Volume 5, Nr. 2, 1993, pp. 199-220

[GSD+09] GAUSEMEIER, J.; STEFFEN, D.; DONOTH, J.; KAHL, S.: Conceptual Design of Modularized Advanced Mechatronic Systems. In: Proceedings of the 17th International Conference on Engineering Design (ICED2009), August 24-27, Stanford, USA, 2009

[GZF+07] GAUSEMEIER, J.; ZIMMER, D.; FRANK, U.; POOK, S.; SCHMIDT, A.: Conceptual Design of Self-optimizing Mechatronic Systems. In: Proceedings of the 16th International Conference on Engineering Design (ICED2007), August 28-31, Paris, France, 2007

[GZO+08] GAUSEMEIER, J.; ZNAMENSCHYKOV, O.; OBERTHÜR, S.; PODLOGAR, H.: AN Approach for Achieving Self-Optimization in Mechatronic Systems by Active Pattern. In: Proceedings of the 8th International Conference on Intelligent Systems and Applications, November 26-28, Kaoshing, Taiwan, 2008

[Hae04] HAECKEL, E.: Kunstformen der Natur. Marix Verlag, Wiesbaden, 2004

[Han55] HANSEN, F.: Konstruktionssystematik – Eine Arbeitsweise für fortschrittliche Konstrukteure. VEB Verlag Technik, Berlin, 2. Auflage, 1955

[Hei06] HEINEMANN, D.: Strukturen von Batterie- und Energiemanagementsystemen mit Bleibatterien und Ultracaps. Dissertation, Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, 2007

[Hes06] HESTERMEYER, T.: Strukturierte Entwicklung der Informationsverarbeitung für die aktive Federung eines Schienenfahrzeugs. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, 2006

[Hey04-ol] HEY, J.: The Data, Information, Knowledge, Wisdom Chain – The Metphorical link. International Oceanographic Commission – OceanTeacher: a training system for ocean data and information management, 2004. Unter: http://web.archive.org/web/20070206032947/ioc.unesco.org/oceanteacher/OceanTeacher2/02_InfTchSciCmm/DIKWchain.pdf, 25. August 2010

[Höh02] HÖHNE, G.: 50 Jahre Konstruktionssystematik – ein Stück Thüringer Technik- und Wirtschaftsgeschichte. In: Thüringer Mitteilungen des VDI und VDE - Das Magazin für Technik, Wissenschaft und Wirtschaft, Ausgabe September-Dezember 2002, Erfurt, 2002

[HS09] HERZOG, O.; SCHILDHAUER, T. (Hrsg.): acatech DISKUTIERT. Intelligente Objekte: Technische Gestaltung - Wirtschaftliche Verwertung - Gesellschaftliche Wirkung. Springer Verlag, Berlin, 2009

[HTF96] HARASHIMA, F.; TOMIZUKA, M.; FUKUDA, T.: Mechatronics – „What Is It, Why and How?" An Editorial. In: IEEE/ASME Transactions on Mechatronics, Volume 1, Nr. 1, 1996

[Hua02] HUANG, M.: Funktionsmodellierung und Lösungsfindung mechatronischer Produkte. Dissertation, Fakultät für Maschinenbau, Universität Karlsruhe, Forschungsberichte aus dem Institut für Rechneranwendung in Planung und Konstruktion, Band 2/2002, Shaker Verlag, Aachen 2002

[ILM02] ISERMANN, R.; LACHMANN, K.-H.; MATKO, D.: Adaptive Control Systems, Prentice-Hall, Hemel Hempstead, 1992

[IS96] IOANNOU, P. A.; SUN, J.: Robust Adaptive Control. Prentice-Hall, 1995

[Ise08] ISERMANN, R.: Mechatronische Systeme – Grundlagen. Springer Verlag, Berlin, 2008

[Jan07] JANOCHA, H. (Ed.): Adaptronics and Smart Structures – Basics, Materials, Design and Applications. Springer Verlag, Berlin, 2. Auflage, 2007

[JK02] JOHNSON, E. N.; KANNAN, S. K.: Adaptive Flight Control for an Autonomous Unmanned Helicopter. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit 2002, August 5-8, Monterey, California, 2002

[Jör89] JÖRGER, K.: Einführung in die Lernpsychologie – Mit Anwendungsbeispielen, Kontrollaufgaben und weiterführenden Literaturhinweisen Herder Verlag, Freiburg, 13. Auflage, 1989

[Kal98] KALLMEYER, F.: Eine Methode zur Modellierung prinzipieller Lösungen mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 42, Paderborn, 1998

[Kas03] KASTOR, M.: Psychologie der Individualität. Verlag Königshausen & Neumann GmbH, Würzburg, 2003

[KC03] KNOLL, A.; CHRISTALLER, T.: Robotik. Fischer Taschenbuch Verlag, Frankfurt am Main, 2003

[Kel00] KELLER, H. B.: Maschinelle Intelligenz – Grundlagen, Lernverfahren, Bausteine intelligenter Systeme. Vieweg Verlag, Braunschweig, 2000

[KGD10] KAHL, S.; GAUSEMEIER, J.; DUMITRESCU, R.: Interactive Visualization of Development Processes in Mechatronic Engineering. In: Proceedings of the 1st International Conference on Modelling and Management of Engineering Processes (MMEP2010), July 19-20, Cambridge, UK, 2010

[KK98] KOLLER, R.; KASTRUP, N.: Prinziplösungen zur Konstruktion technischer Produkte. Springer Verlag, Berlin/Heidelberg, 2. Auflage, 1998

[Klö09] KLÖPPER, B.: Ein Beitrag zur Verhaltensplanung für interagierende intelligente mechatronische Systeme in nicht-deterministischen Umgebungen. Dissertation, Fakultät für Wirtschaftswissenschaften, Universität Paderborn, 2009

[Kno03] KNOLL, A.: Roboter für Menschen – Zielvorstellungen und Ansätze für autonome smarte Serviceroboter. In: Mattern, F. (Hrsg.): Total vernetzt – Szenarien einer informatisierten Welt. Springer Verlag, Berlin, 2003, pp. 187-208

[Kol98] KOLLER, R.: Konstruktionslehre für den Maschinenbau – Grundlagen zur Neu- und Weiterentwicklung technischer Produkte mit Beispielen. Springer Verlag, Berlin/Heidelberg, 1998

[Kop97] KOPETZ, H.: Real-Time Systems – Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, Dordrecht, 1997

[Köc04] KÖCKERLING, M.: Methodische Entwicklung und Optimierung der Wirkstruktur mechatronischer Produkte. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 143, Paderborn, 2004

[KTM+08] KOPP, B.; TABELING, S.; MOSCHNER, C.;WESSEL, K.: Kognitive Hirnleistungen des präfrontalen Kortex. Der Nervenarzt, Springer Medizin Verlag, 2008

[KW08] KOMUS A.; WAUCH, F.: Wikimanagement – Was Unternehmen von Social Software und Web 2.0 lernen können. Wissenschaftsverlag, München, 2008

[Lai08] LAIRD, J. E.: Extending the Soar Cognitive Architecture. In: Proceedings of 1st Conference on Artificial General Intelligence (AIG08), March 1-3, Memphis, USA, 2008

[Lan00] LANGLOTZ, G.: Ein Beitrag zur Funktionsstrukturentwicklung innovativer Produkte. Dissertation. Shaker Verlag, Aachen, 2000

[Lan91] LANGENSCHEIDT: Langenscheidts Großes Schulwörterbuch Lateinisch - Deutsch. Langenscheidt KG, Berlin, 8. Auflage, 1991

[LC01] LEUF, B.; CUNNINGHAM, W.: The Wiki Way – Quick Collaboration on the Web. Addison-Wesley, Harlow, 2001

[Len02] LENZEN, M.: Natürliche und künstliche Intelligenz – Einführung in die Kognitionswissenschaft. Campus Verlag, Frankfurt am Main, 2002

[LHL01] LÜCKEL, J.; HESTERMEYER, T.; LIU-HENKE, X.: Generalization of the Cascade Principle in View of Structured Form of Mechatronic Sysems. In: International Conference on Advanced Intelligent Mechatronics (AIM2001), IEEE/ASME, Como, Italy, 2001

[Lin09] LINDEMANN, U.: Methodische Entwicklung technischer Produkte – Methoden flexible und situationsgerecht anwenden. Springer Verlag, Berlin, 3. Auflage, 2009

[LL06] LECOINTRE, G; LE GUYADER, H.: Biosystematik: Alle Organismen im Überblick. Springer Verlag, Berlin 2006

[LL07] LI; H.; LIU, D.: Power Distribution Strategy of Fuel Cell Vehicle System with Hybrid Energy Storage Elements Using Triple Half Bridge (THB) Bidirectional DC-DC converter. In: Proceedings of 42nd Industry Applications Conference, September 23-27, New Orleans, USA, 2007, pp. 636-642

[LLR08] LANGLEY, R.; LAIRD, J. E.; ROGERS, S.: Cognitive architectures – Research issues and challenges. In press: Cognitive Systems Research (doi:10.1016/j.cogsys.2006.07.004), Elsevier B.V., 2008

[LMP99] LENGELER, J. W.; MÜLLER, B. S.; DI PRIMIO, D.: Kognitive Leistungen und einzellige Lebewesen. GMD Report 57. GMD-Forschungszentrum Informationstechnik GmbH, Sankt Augustin, 1999

[LNR87] LAIRD, J. E.; NEWELL, A.; ROSENBLOOM, P. S.: Soar: an architecture for general intelligence. In: Artificial Intelligence, Volume 33, 1987, pp. 1-64

[Los06] LOSSACK, R.-S.: Wissenschaftstheoretische Grundlagen für die rechnerunterstützte Konstruktion. Springer Verlag, Berlin, 2006

[Low09] LOW, C. Y.: A Methodology to Manage the Transition from the Principle Solution towards the Controller Design of Advanced Mechatronic Systems. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 254, Paderborn, 2009

[Lun07] LUNZE, J.: Regelungstechnik 1 – Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Reglungen. Springer Verlag, Berlin/Heidelberg, 6. Auflage, 2007

[May08] MAY, K. A.: Identifikation von Koordinationsmustern in autonomen mechatronischen Echtzeitsystemen. Bachelorarbeit. Fakultät für Elektrotechnik, Informatik und Mathematik. Universität Paderborn, Fachgebiet Softwaretechnik, 2008

[Mca06] MCAFEE, A. P.: Enterprise 2.0. The Dawn of Emergent Collaboration. In: MITSloan Management Review, Volume 47, Nr. 3, 2006, pp. 25-28

[Met00] METZINGER, T.: Die Selbstmodell-Theorie der Subjektivität – Eine Kurzdarstellung für Nicht-Philosophen in fünf Schritten. In Greve, W. (Hrsg.): Psychologie des Selbst, Beltz PVU, Weinheim, 2010, S. 317-336

[Met05] METZINGER, T.: Precis of Being No One. In: PSYCHE – An Interdisciplinary Journal of Research of Consciousness, June 11 (5), 2005, pp. 1-35

[Mie98] MIETTINEN, K.: Nonlinear multiobjective optimization. Kluwer Academic Publishers, Second Printing, Norwell, 2002

[Mod10] MODELICA ASSOCIATION: Modelica® A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification – Version 3.2. Linköping, Sweden, 2010

[Mor69] MORI, T.: Yaskawa Internal Trademark Application Memo, 21.131.01. July, 1969

[Möh04] MÖHRINGER, S.: Entwicklungsmethodik für mechatronische Systeme. Habilitation. HNI-Verlagsschriftenreihe, Band 156, Paderborn, 2004

[MS10] METZLER, T.; SHEA, K.: Cognitive Products – Definition and Framework. In: Proceedings of International Design Conference (DESIGN2010), May 17-20, Dubrovnik, Croatia, 2010, pp. 865-874

[Mül98] MÜLLER, B. S.: Identifikation elementarer kognitiver Leistungen. GMD Report 17, Sankt Augustin, 1998

[MV80] MATURANA, H. R.; VARELA, F. J.: Autopoiesis and Cognition – The Realization of the Living. Goldmann, D. Reidel Publishing Company, Dordrecht, Holland, 1980

[MV90] MATURANA, H. R.; VARELA, F. J.: Der Baum der Erkenntnis. Die biologischen Wurzeln des Erkennens. Goldmann, 1990

[Nau00] NAUMANN, R.: Modellierung und Verarbeitung vernetzter intelligenter mechatronischer Systeme. Fortschrittsbericht VDI Reihe 20, Nr. 318, VDI Verlag, Düsseldorf, 2000

[New90] NEWELL, A.: Unified theories of cognition. Harvard University Press, Cambridge, 1990.

[NJT08] NACHTIGAL, V.; JÄKER, K.-P.; TRÄCHTLER, A.: Development and Control of a Quarter-Vehicle Testbed for a Fully Active X-by-Wire Demonstrator. In: 9th International Symposium on Advanced Vehicle Control (AVEC2008), October 6-9, Kobe, Japan, 2008.

[NT97] NONAKA, I.; TAKEUCHI, H.: Die Organisation des Wissens – Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen. Campus Verlag, Frankfurt am Main, 1997

[Nor05] NORTH, K.: Wissensorientierte Unternehmensführung – Wertschöpfung durch Wissen. Gabler Verlag, Wiesbaden, 4. Auflage, 2005

[Oes05] OESTERREICH, B.: Objektorientierte Softwareentwicklung. Analyse und Design mit UML 2.0. Oldenbourg Verlag, München, 7. Auflage, 2005

[OZK+08] OBERTHÜR, S; ZNAMENSCHYKOV, O.; KLÖPPER, B.; VÖCKING, H.: Improved Flixible Resource Management by Means of Look-Ahead Scheduling and Baysian Forecasting. In: Gausemeier, J.; Rammig, F. J.; Schäfer, W. (Hrsg.): 7. Internationales Heinz Nixdorf Symposium "Self-Optimizing Mechatronic Systems: Design the Future", 20-21 Februar 2008, HNI-Verlagsschriftenreihe, Band 223, Paderborn, 2008, S. 361-376

[Pae06] PAETZOLD, K.: On the Importance of a Functional Description for the Development f Cognitive Technical Systems. In: Proceedings of the International Design Conference (DESIGN2006), May 15-18, Dubrovnik, Croatia, 2006

[PBF+07] PAHL, G.; BEITZ, W.; FELDHUSEN, J.; GROTE, K.-H.: Konstruktionslehre – Grundlagen erfolgreicher Produktentwicklung – Methoden und Anwendung. Springer Verlag, Berlin, 7. Auflage, 2007

[Pol85] POLANYI, M.: Implizites Wissen. Suhrkamp Verlag, Frankfurt am Main, 1985

[Pot05] POTTHARST, A.: Energieversorgung und Leittechnik einer Anlage mit Linearmotor getriebenen Bahnfahrzeugen. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, 2005

[Pri98] DI PRIMIO, F. Roboterkognition – Aufsetzpunkte und Probleme für eine biologisch-evolutionär orientierte, synchronisch und diachronisch minimierende Forschung. GMD Report 16, Sankt Augustin, 1998

[PS05] PAETZOLD, K.; STUPPY, J.: Multidisziplinäre Ansätze zur Entwicklung kognitiver technischer Systeme. In: 50. Internationales Wissenschaftliches Kolloquium der TU Illmenau, 19.-23. September, Illmenau, 2005

[Pup86] PUPPE, F.: Expertensysteme. In: Informatik Spektrum, Band 9, Heft 1, Springer Verlag, Berlin, 1986, S. 1-13

[Ram06] RAMMIG, F. J.: Towards Self-coordinating Ubiquitous Computing Environments. In: Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC2006), August 1-4, Seoul, Korea, 2006, pp. 2-13

[RBW+09] ROMAUS, C.; BÖCKER, J.; WITTING, K.; SEIFRIED, A.; ZNAMENSHCHYKOV, O.: Optimal Energy Management for a Hybrid Energy Storage System Combining Batteries and Double Layer Capacitors. In: Proceedings of the IEEE Energy Conversion Congress and Exposition (ECCE2009), September 20-24, San Jose, USA, 2009

[RFK08] RODRIGUEZ, L. V.; FELDER, M.; KNOLL, A.: A cognitive architecture framework for CoTeSys. In: Proceedings of 1st International Workshop on Cognition for Technical Systems, October 8-9, Munich, 2008

[Rie07] RIEGLER, A.: The Goose, the Fly, and the Submarine Navigator – The Case for Interdisciplinarity in Artificial Cognition Research. In Loula, A; Gudwin, R, Quieroz, J. (Eds.): Artificial cognition systems. Idea Group Publishing, Hershey, 2007

[RNJ+09] REINOLD, P.; NACHTIGAL, V.; JÄKER, K.-P.; TRÄCHTLER, A:. Control Strategy for the Lateral and Longitudinal Dynamics of a Fully Active X-by-wire Test Vehicle. In: Proceedings of the European Control Conference (ECC2009), August 23-26, Budapest, Hungary, 2009

[Rod91] RODENACKER, W. G.: Methodisches Konstruieren – Grundlagen, Methodik, praktische Beispiele. Springer Verlag, Berlin, 4. Auflage, 1991

[Rot01] ROTH, K.: Konstruieren mit Konstruktionskatalogen: Band 2 – Kataloge. Springer Verlag, Berlin, 3. Auflage, 2001

[Rot94] ROTH, K.: Konstruieren mit Konstruktionskatalogen: Band 1 – Konstruktionslehre. Springer Verlag, Berlin, 2. Auflage, 1994

[Roh08-ol] ROHDE, T: The Straight on Mechatronics – Yaskawa, the company that invented the term, describes its evolution. Unter: http://yaskawa.com/site/products.nsf/staticPagesNewWindow/mechatronics.html, 10. April 2010

[RM96] ROTH, G.; MENZEL, R.: Kognitive Funktionen, sprachliche und nichtsprachliche Kommunikation. In Dudel, J.; Menzel, R.; Schmidt, F. R. (Hrsg.).: Neurowissenschaft. Vom Molekül zur Kognition. Springer Verlag, Heidelberg-Berlin, 1996

[RN07] RUSSEL, S.; NORVIG, P.: Künstliche Intelligenz – Ein moderner Ansatz. Pearson Studium, München, 2. Auflage, Nachdruck, 2007

[RRV+02] REICHERT, M.; RUF, W.-D.; VOGT, A.; SCHIESSLE, E. (Hrsg.): Mechatronik 2, Vogel Buchverlag, 2002

[RS98] RYSDYK, R. T.; CALISE, A. J.: Nonlinear Adaptive Flight Control Using Neural Networks. In: IEEE Controls Systems Magazine. Volume 18, Nr. 6, 1998, pp. 14-25

[Rup07] RUPP, C.: Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis. Carl Hanser Verlag, München, 4. Auflage, 2007

[Rze03] RZEVSKI, G.: On conceptual design of intelligent mechatronic systems. In: Mechatronics, Volume 13, 2003, pp. 1029-1044

[Sal01] SALUSTRI, F. A.: Using Design Patterns to Promote Multidisciplinary Design. In: Proceedings of CSME International Conference on Multidisciplinary Design Engineering, November, Montreal, Canada, 2001

[Sal05] SALUSTRI, F. A.: Using Design Pattern Languages in Design Engineering. In: Proceedings of 15th International Conference Engineering Design (ICED2005), August 15-18, Melbourne, Australia, 2005

[Sau06] SAUER, T.: Ein Konzept zur Nutzung von Lösungsobjekten für die Produktentwicklung in Lern- und Assistenzsystemen. VDI-Fortschritt-Berichte, VDI Reihe 1, Nr. 390, VDI Verlag, Düsseldorf, 2006

[SB89] SASTRY, S.; BODSON, M.: Adaptive Control – Stability, Convergence, and Robustness. Prentice-Hall, New Jersey, 1989

[Sch00] SCHMIDT, B.: Modellierung menschlichen Verhaltens – Das PECS-Refernzmodell. Gruner Druck, 2000

[Sch05] SCHIEBLER, T. H. (Hrsg.): Anatomie – Histologie, Entwicklungsgeschichte, makroskopische und mikroskopische Anatomie, Topographie. Springer Medizin Verlag, Heidelberg, 9. Auflage, 2005

[Sch06a] SCHMID, U.: Computermodelle des Denkens und Problemlösens. In Funke, J. (Hrsg.), Enzyklopädie der Psychologie, Themenbereich C: Theorie und Forschung, Serie 2: Kognition, Band 8, Denken und Problemlösen, Hogrefe, 2006, S. 483-547

[Sch06b] SCHMIDT, A.: Wirkmuster zur Selbstoptimierung – Konstrukte für den Entwurf selbstoptimierender Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 175, Paderborn, 2006

[See03] SEEL, N. M.: Psychologie des Lernens – Lehrbuch für Psychologen und Pädagogen. Reinhardt Verlag, München, 2003

[SEH+10] SCHÄFER, W.; ECKHARDT, T.; HENKE, C.; KAISER, L.; RIEKE, J.; TICHY, M.: Der Softwareentwurf im Entwicklungsprozess mechatronischer Systeme. In: Tagungsband des 7. Paderborner Workshop Entwurf mechatronischer Systeme, 18.-19. März, HNI-Verlagsschriftenreihe, Band 272, Paderborn, 2010

[SFB08] SFB 614 – Finanzierungsantrag für den Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“, 2009/2 bis 2013/1. Universität Paderborn, 2008

[SNC87] SCHNUPP, P.; NGUYEN, H.; CHAU, T.: Expertensysteme-Praktikum. Springer Verlag, Berlin, 1987

[SP05] STUPPY, J.; PAETZOLD, K.: Integration der Kognition in technische Systeme. In: Tagungsband des 16. Symposiums „Design for X“, 13.-14. Oktober, Neukirchen, 2005, S. 130-150

[SR88] SCHULZE, K.-P.; REHBERG, K.-J.: Entwurf von adaptiven Systemen – Eine Darstellung für Ingenieure. VEB Verlag Technik, Berlin, 1988

[Ste07] STEFFEN, D.: Ein Verfahren zur Produktstrukturierung für fortgeschrittene mechatronische Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Band 207, Paderborn, 2007

[Ste12] STERN, W.: The Psychological Methods of Intelligence Testing. Warwick & York, Baltimore, 1912

[Str06] STRASSER, A.: Kognition künstlicher Systeme. Ontos Verlag, 2006

[Str95] STROHNER, H.: Kognitive Systeme – Eine Einführung in die Kognitionswissenschaft. Westdeutscher Verlag, Opladen, 1995

[Str96] STRUBE, G.: Wörterbuch der Kognitionswissenschaft. Klett-Cotta, 1996

[Str98] STRUBE, G.: Modelling Motivation and Action Control in Cognitive Systems. In: Schmid, U.; Krems, J. F.; Wysocki, F.: Mind Modelling. Pabst, Berlin, 1998

[Stu09] STUCKENSCHMIDT, H.: Ontologien – Konzepte, Technologien und Anwendungen. Springer Verlag, Berlin/Heidelberg, 2009

[Suh90] SUH, N. P.: The Principles of Design. Oxford University Press, New York, 1990

[Suh93] SUHM, A.: Produktmodellierung in wissensbasierten Konstruktionssystemen auf Basis von Lösungsmustern. Dissertation, Fakultät für Maschinenbau, Universität Karlsruhe, Reihe Konstruktionstechnik, Verlag Shaker, Aachen, 1993

[Szy02] SZYPERSKI, C.: Component Software – Beyond Object-Oriented Programming. Addison-Wesley Longman, Amsterdam, 2. Auflage, 2002

[TMV06] TRÄCHTLER, A.; MÜNCH, E.; VÖCKING, H.: Iterative Learning and Self-Optimization Techniques for the Innovative RailCab-System. In: 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON2006), November 7-12, Paris, France, 2006

[Trä09] TRÄCHTLER, A.: Entwurf intelligenter mechatronischer Systeme – Regelungstechnische Konzepte für selbstoptimierendes Verhalten. In: Tagungsband des 6. Paderborner Workshop Entwurf mechatronischer Systeme, 2.-3. April, HNI-Verlagsschriftenreihe, Band 250, Paderborn, 2009

[Urb04] URBAN, C.: Das Referenzmodell PECS – Agentenbasierte Modellierung menschlichen Handelns, Entscheidens und Verhaltens. Dissertation, Fakultät für Mathematik und Informatik, Universität Passau, 2004

[Ver10] VERNON, D.: Enaction as a Conceptual Framework for Developmental Cognitive Robotics. In: Paladyn Journal of Behavioral Robotics, Volume 1, Nr. 2, 2010, pp. 89-98

[VMS07] VERNON, D.; METTA, G.; SANDINI, G.: A Survey of Artificial Cognitive Systems – Implications for the Autonomous Development of Mental Capabilities in Computational Agents. In: IEEE Transactions on Evolutionary Computation – Special issue on Autonomous Mental Development, Volume 11, Nr. 2, 2007, pp. 151-180

[Web10] WEBER, H.: Erstellung nutzerindividueller Dokumente für die Vermittlung von Produktentwicklungswissen durch den Einsatz von Topic Maps. VDI Fortschritt-Berichte, Reihe 1, Nr. 408, VDI Verlag, Düsseldorf, 2010

[WEH+08] WITKOWSKI, U.; EL-HABBAL, M.; HERBRECHTSMEIER, S.; TANOTO, A.; PENDERS, J.; ALBOUL, L.; GAZI, V.: Ad-hoc network communication infrastructure for multi-robot systems in disaster scenarios. In: IARP/EURN Workshop in Robotics for Risky Interventions and Environmental Surveillance, January, Spain, 2008

[Wei06] WEILKIENS, T.: Systems Engineering mit SysML/UML – Modellierung, Analyse, Design. dpunkt Verlag, Heidelberg, Germany, 2008

[WSD+08] WITTING, K.; SCHULZ, B.; DELLNITZ, M.; BÖCKER; J.; FRÖHLEKE, N.: A new approach for online multiobjective optimization of mechatronical systems. In: International Journal on Software Tools for Technology Transfer (STTT), Volume 10, Nr. 3, 2008, pp. 223-231

[ZG04] ZIMBARDO, P. G.; GERRIG, R. J.: Psychologie. Pearson Studium, München, 2004

[Zwi71] ZWICKY, F.: Entdecken, Erfinden, Forschen im morphologischen Weltbild. Droemer-Knaur Verlag, München, 1971

Normen und Richtlinien

[DIN1421] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (DIN): Gliederung und Benennung in Texten – Abschnitte, Absätze, Aufzählungen. DIN 1421, Beuth-Verlag, Berlin, 1983

[DIN19225] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (DIN): Benennung und Einteilung von Reglern. DIN 19 225, Beuth-Verlag, Berlin, 1981

[DIN19226] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (DIN): Leittechnik – Regelungstechnik und Steuerungstechnik – Allgemeine Grundbegriffe. DIN 19 226 Teil 1, Beuth-Verlag, Berlin, 1995

[IEEE1076.1] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE): IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes). IEEE 1076.1-1999, The Institute of Electrical and Electronics Engineers, New York, USA, 1999

[ISO19501] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO); INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC): Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2. ISO/IEC 19501:2005(E), ISO copyright office, Geneva, 2005

[ISO19759] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO); INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC): Software Engineering – Guide to the Software Engineering Body of Knowledge (SWEBOK). Technical Report ISO/IEC TR 19759:2005(E), ISO copyright office, Geneva, 2005

[ISO9000] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO): Quality management systems – Fundamentals and vocabulary. ISO 9000:2005, ISO copyright office, Geneva, 2005

[VDI2206] VEREIN DEUTSCHER INGENIEURE (VDI): Entwicklungsmethodik für mechatronische Systeme. VDI-Richtlinie 2206, Beuth-Verlag, Berlin, 2004

[VDI2221] VEREIN DEUTSCHER INGENIEURE (VDI): Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. VDI-Richtlinie 2221, Beuth-Verlag, Berlin, 1993

[VDI2222] VEREIN DEUTSCHER INGENIEURE (VDI): Konstruktionsmethodik – Methodisches Entwickeln von Lösungsprinzipien. VDI-Richtlinie 2222, Blatt 1, Beuth-Verlag, Berlin, 1997

[VDI2422] VEREIN DEUTSCHER INGENIEURE (VDI); VEREIN DEUTSCHER ELEKTROTECHNIKER(VDE) : Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik. VDI/VDE-Richtlinie 2422; Beuth-Verlag, Berlin, 1994

[VDI2727] VEREIN DEUTSCHER INGENIEURE (VDI): Konstruktionskataloge – Lösung von Bewegungsaufgaben mit Getrieben – Grundlagen. VDI-Richtlinie 2727, Blatt 1, Beuth-Verlag, Berlin, 2007

[VDI2803] VEREIN DEUTSCHER INGENIEURE (VDI): Funktionenanalyse – Grundlagen und Methode. VDI-Richtlinie 2803, Blatt 1, Beuth-Verlag, Berlin, 1996

[VDW02.02] VEREIN DEUTSCHER WERKZEUGMASCHINENFABRIKEN (VDW): Funktionsbeschreibung. VDW-Richtlinie 02.2002, VDW, Frankfurt, 2002

Anhang

Inhaltsverzeichnis

A1 Ergänzungen zum Stand der Technik

A1.1 Lösungsmuster nach GAUSEMEIER et al.

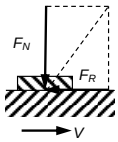
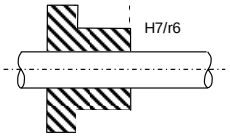
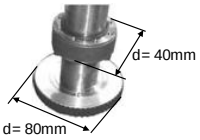
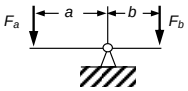
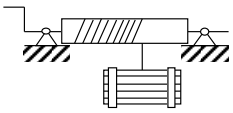
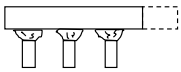
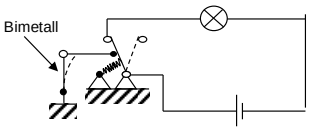
Teilfunktionen	**Physikal. Effekte** (lösungsneutral)	**Wirkprinzipien für eine Teilfunktion** (Phys. Effekte sowie geometrische und stoffliche Merkmale)	**Lösungselemente** (Basieren auf Wirkprinzipien, aber die Merkmale sind mit konkreten Werten belegt)
T_l **Drehmoment übertragen** T_l	**Reibungseffekt**  $F_R = \mu \cdot F_N$		Werkstoff: St 37-2 
F_a **Handkraft vergrößern** F_b	**Hebeleffekt**  $F_a \cdot a = F_b \cdot b$		 Seillänge: 10 m max. Zugbelastung: 360 kg kleinstes Hakenmass: 50mm
ϑ ϑ **Signal geben wenn** $\vartheta \geq \vartheta_a$ ϑ	**Ausdehnungseffekt**  $\Delta l = \alpha \cdot l \cdot \Delta \vartheta$		 Überstrom-bimetallrelais mit Nennstrom 300 A

Bild A-1: Beispiele für Wirkprinzipien der Domäne Maschinenbau [GHK+06, S. 374] (in Anlehnung an [PBF+07, S. 53])

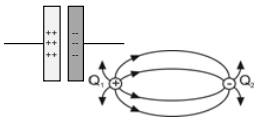
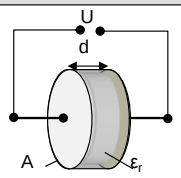
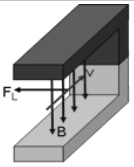
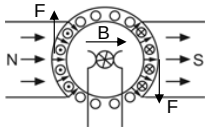
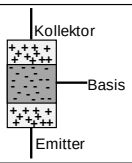
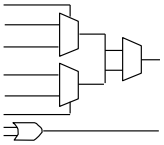
Teilfunktionen	**Physikal. Effekte** (lösungsneutral)	**Wirkprinzipien für eine Teilfunktion** (Phys. Effekte sowie geometrische und stoffliche Merkmale)	**Lösungselemente** (Basieren auf Wirkprizipien, aber die Merkmale sind mit konkreten Werten belegt)
W_a **Energie speichern** W_b	**Elektrisches Feld** 	$W_{Kond} = \frac{1}{2} C \cdot U^2$ $C = \varepsilon_r \cdot \varepsilon_0 \cdot \frac{A}{d}$ 	Elna-Kondensator ROA 851626 C=100µF U=25V 
W_e **Energie wandeln** W_m	**Lorentzkraft** 	 $\vec{F} = I \cdot (\vec{l} \times \vec{B})$	Elektromotor C40 7000U/min. 
S_1 S_2 **Signal schalten** S_{out}	**pn-Übergang**  (Transistor →Gatter →FPGA)	ACT Logikmodul 	XILINX XCS10XL (FPGA) 10K Gates PLCC Package 18$ 

Bild A-2: Beispiele für Wirkprinzipien der Domäne Elektrotechnik [GHK+06, S. 374]

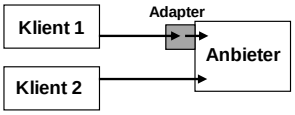
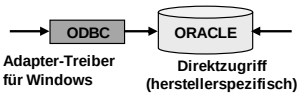
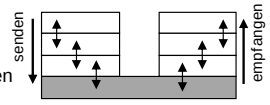
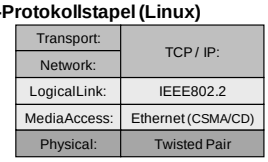
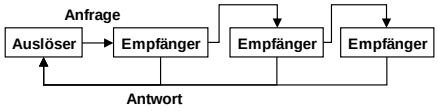
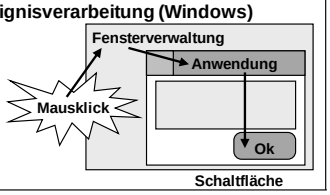
Teilfunktionen	Softwaremuster / Pattern für eine Teilfunktion	Lösungselemente / Softwarekomponenten (Basieren auf Mustern, aber die Merkmale sind konkret ausgeprägt)
Passe die Schnittstelle einer Klasse an eine andere an, die von einem Klienten erwartet wird, ohne die Klasse selbst zu verändern	**Adapter** • Delegation • Datenkapselung 	**Datenbankzugriff** 
Übertrage Anwendungsdaten zwischen zwei Kommunikationsteilnehmern, unabhängig von der Art der physikalischen Netzwerkverbindung	**Abstraktion über Schichten** • Kommuniziere nur zwischen benachbarten Schichten • Entkopple nicht direkt benachbartes Verhalten 	**TCP/IP-Protokollstapel (Linux)** 
Vermeide die Kopplung des Auslösers einer Anfrage an seinen Empfänger, in dem mehr als ein Objekt die Möglichkeit zur Erledigung erhält	**Zuständigkeitskette** • Verkette die Empfänger; leite Anfragen weiter 	**Ereignisverarbeitung (Windows)** 

Bild A-3: Beispiele für Muster der Softwaretechnik [GHK+06, S. 375]

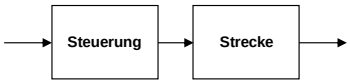
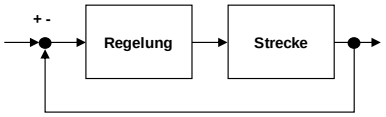
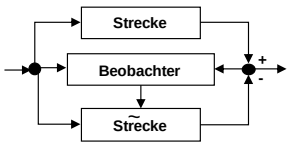
Teilfunktionen	Muster			Lösungselemente
Größe beeinflussen	**Steuerung** 			• Inverse Strecke • Optimale Steuerungsfunktion • Kennfeld
	Regelung 	Eingrößenregler		• P-Regler • PI-Regler (para.) • PD-Regler (n. para.) • Riccatti-Regler • Allg. Zustandsregler
		Mehrgrößenregler	• Kaskadenregelung • Zustandsvektorrückführung • ...	
Größe bestimmen	**Beobachter** 			• Luenberg-Beobachter • Kalman-Filter
	Nachbilden			• Modell der Strecke, z.B. PT1,
	Messen/Erfassen			...

Bild A-4: Beispiele für Muster der Regelungstechnik [GHK+06, S. 375]

A1.2 Beispielmuster nach SALUSTRI

Im Folgenden wird das Beispielmuster „variable fluid mixer“ nach SALUSTRI vorgestellt [Sal05]:

Problem: *design a system that mixes fluid (including powder) ingredients, controlling production rate and mix ratios.*

Sometimes, the same equipment may be used to mix different ingredients at different mix ratios to produce different products (EG: mixers for paint, fertilisers, or dry ingredients of baked foods). Although variability in mix ratio rarely affects production rate in such cases, even small mix ratio variations can lower product quality.

Modular design suggests there are advantages to duplicating the ingredient delivery system identically for each ingredient (for example, having a separately driven pump for each ingredient) to get economies of scale in the subassemblies. However, variation in the relative rates of the motors that drive the pumps can cause unacceptable variations in the mix ratio. The variation arises because the motors, though structurally separate from one another, are functionally coupled: the mix ratio is determined by all motor (and hence pump) speeds and their associated variations, which are additive.

Furthermore, many environments in which this situation can occur are “dirty;” i.e. there is an assortment of contaminants in the operating environment that can hinder relatively delicate control machinery or electronics.

Key drivers are:

- Fine control of mix ratio is required
- Delicate electronics should not be used because of dirty environment
- The mix ratio must be adjustable
- The mix ratio variability must be low
- Reliability must be high
- Capital, operating, and maintenance costs must be kept low

Therefore: design a *functionally* modular system.

Assign one pump per ingredient, but use a single motor to drive all the pumps, and variable ratio gearboxes to vary the speeds of the pumps (and therefore the mix ratio). The variability arising from a single motor and a single gearbox will be transmitted proportionally to all pumps, effectively cancelling it out or at least dramatically reducing it.

This solution is consistent with Axiomatic Design [Suh90]. The functional requirements of this product are (a) it must produce the right amount of product and (b) it must mix the ingredients correctly. If separate motors drive each pump, then the design parameters are the speed of each pump. This induces a fully coupled design, which is undesirable. Setting the speeds of the motors to achieve prescribed productivity and mix ratio values becomes an iterative process. Variation over time in the speeds of the motors requires active, dynamic control of the system.

However, if using a single motor and variable ratio gearboxes, the design parameters are the speed of the motor and the gearbox ratio; this design is decoupled, which is preferred. Setting the motor speed and gear ratios is not an iterative process and does not require active, dynamic control.

But:

Selection of gears and calculation of tolerances must be done carefully, to minimize variability of the gearboxes between gears. Functional modularity introduces structural coupling here. One must pay attention to ensuring this coupling does not lower reliability in specific cases. Reliability Analysis and Failure Mode And Effect Analysis are recommended methods to assess this.

Structural modularity can be salvaged to a degree by designing the shafts and structural elements to facilitate replacement of gearboxes, pumps, the motor, and other major subsystems.

Furthermore, particular attention must be given to the gearbox design or specification. Backlash and other effects must be accounted for to ensure the gearbox is properly specified to prevent a quality loss similar using multiple motors.

A1.3 Beispielmuster nach DEIGENDESCH

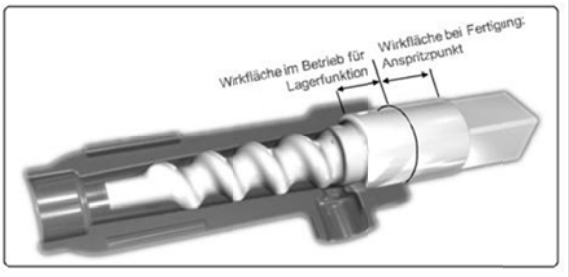
Name	**Anspritzflächen**
Situation	Mikrobauteile werden durch Spritzgießen hergestellt.
Problem	Beim Spritzgießprozess wird die Formmasse in die Kavität gepresst. Nach Erkalten und Erstarren der Formmasse wird das Bauteil entformt und vereinzelt. Dabei wird der noch mit dem Bauteil verbundene Angusskanal gewaltsam vom Bauteil getrennt. Die Geometrie der Trennstelle ist nicht eindeutig definiert und im Betrieb unter Umständen als funktionsrelevante Wirkfläche ungeeignet.
Lösung	Trennung von funktionsrelevanten Flächen während des Betriebs von Anspritzflächen (Wirkflächen während der Fertigung).
Tragweite	Chancen: keine Funktionseinschränkung im Betrieb durch Anspritzpunkte Risiken: zusätzliche Flächen (Fertigungswirkflächen), reduziertes Miniaturisierungspotential
Konsequenz	Bauteil mit definierten Wirkflächen für den Betrieb und separaten Wirkflächen für die Produktion (Anspritzpunkte).
Beispiel	Der Anspritzpunkt wurde bei der Schnecke als separate Wirkfläche gestaltet. Die Wirkfläche für das Gleitlager bleibt unberührt. Erkennbar ist auch der notwenige zusätzliche Bauraum. 
Verwandte Muster	• Auswerferflächen müssen gezielt eingebracht und von während des • Markerflächen müssen gezielt eingebracht und von während des Betriebs
Quellen	Albers, A., Deigendesch, T. & Turki, T.: Design Patterns in Microtechnology. In: Design Society (Veranst.): Int. Conf. on Engineering Design ICED (Stanford 2009)
Metadaten	Schlagworte: Anspritzzapfen Kategorien: Objektmuster Signifikanz: A Ersteller: T. Deigendesch

Bild A-5: Beispielmuster aus der Mikrotechnik nach [Dei09, S. 193ff.]

A2 Lösungsmuster für fortgeschrittene mechatronische Systeme

Im Folgenden werden weitere Lösungsmuster vorgestellt, die im Verlauf der Arbeit sowohl zur Validierung der einheitlichen Spezifikation der Lösungsmuster für fortgeschrittene mechatronische Systeme als auch zur Wiederverwendung von Lösungswissen im SFB 614 erarbeitet wurden. Aufgrund des Umfangs der Lösungsmuster werden nur die wesentlichen Aspekte gezeigt. So können keine Anwendungsbeispiele des Aspekts Kontext sowie alle möglichen Lösungsprinzipien im Detail erläutert werden. Für die Lösungsmuster für den kognitiven Operator (LM_{KO}) werden in A3 die Verfahren weitestgehend aufgelistet und vermerkt, in welchen LM_{KO} die Verfahren prinzipiell eingesetzt werden können. Die vollständige Aufnahme der Lösungsmuster erfolgte in der Lösungsmusterwissensbasis (vgl. Kap. 4.6.2).

A2.1 LM_{GS} „Gleichstrommaschine“

Prinzipiell ist die einheitliche Spezifikation eines Lösungsmusters für fortgeschrittene mechatronische Systeme auch auf Bestandteile des Grundsystems anwendbar. Das Lösungsmuster für das Grundsystem (LM_{GS}) „Gleichstrommaschine“ soll dies beispielhaft zeigen. Eine Gleichstrommaschine lässt sich als Antriebsmotor oder als Generator nutzen. Im Gegensatz zu anderen elektrisch betriebenen Motoren ist die Drehzahl in weiten Bereichen beliebig einstellbar. Darüber hinaus ergibt sich durch die Verbindung mit einem steuerbaren Gleichrichter ein sehr geringer regelungstechnischer Aufwand, bei hervorragenden dynamischen Eigenschaften. Das Einsatzgebiet von Gleichstrommotoren reicht von Schwerantrieben mit einer Leistung von etwa 10 MW bis hin zu batteriebetriebenen Geräten in einem Leistungsbereich von unter einem Watt. Wichtige charakteristische Merkmale sind:

- *Einsatzbedingungen:* Da es sich um elektronische Bauteile handelt, sind sämtliche Kontakte von einem leitenden Medium wie Wasser freizuhalten. Der gesamte Arbeitsraum ist zudem gegenüber feinen Feststoffen abzudichten. Dies gilt für metallische sowie nicht metallische Stoffe gleichermaßen.
- *Geometrie:* Die Abmaße von Radiallagern variieren je nach Stärke der Belastung und Größe der abzustützenden Welle. Sie bestimmen die Gesamtabmaße.
- *Stoffliche Merkmale:* Das Gehäuse einer Gleichstrommaschine besteht je nach Ausführung aus Baustahl oder legiertem Stahl. Die Wicklungen auf der Ankerwelle sind Kupferdrähte. Um den Strom von den Anschlussklemmen auf den Kommutator zu übertragen, werden Kohlebürsten eingesetzt**.**
- *Kinematik:* Das Gehäuse der Maschine ist in der Regel fest mit der Umgebung verbunden. Die Ankerwelle erfährt hingegen eine rotatorische Bewegung.

- *Montage:* Bezüglich der Montage ist darauf zu achten, dass das Gehäuse aufgrund des vorliegenden Drehmoments in geeigneter Weise mit der Umgebung verbunden werden muss.

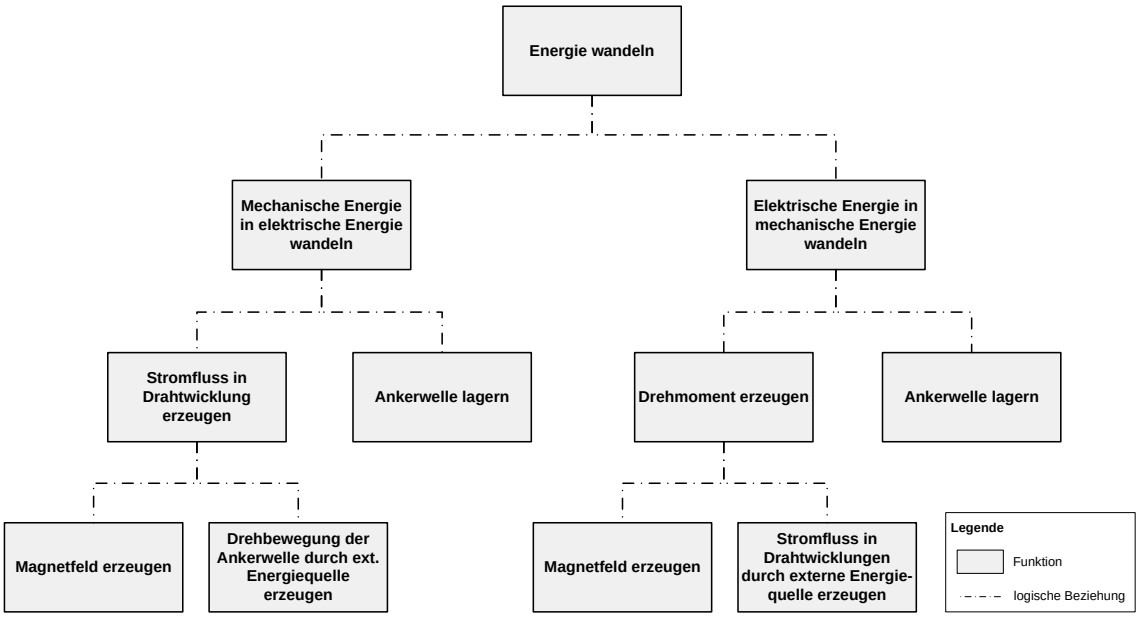


Bild A-6: Funktionen des LM_{GS} „Gleichstrommaschine“

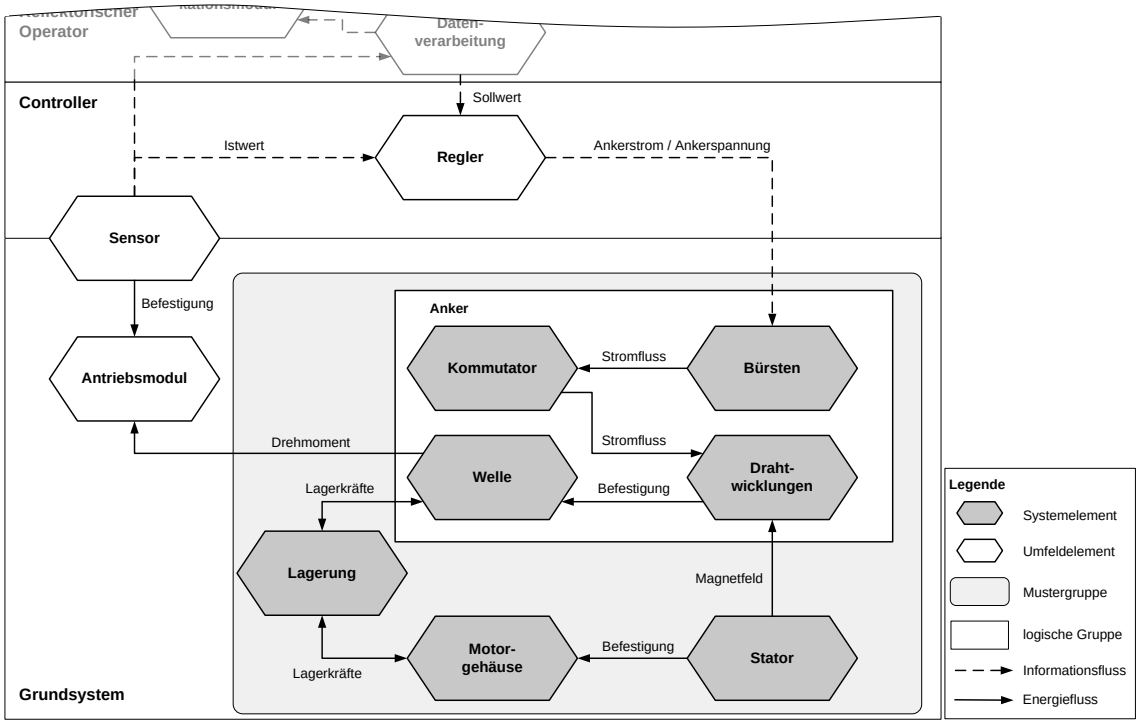


Bild A-7: Wirkstruktur des LM_{GS} „Gleichstrommaschine“

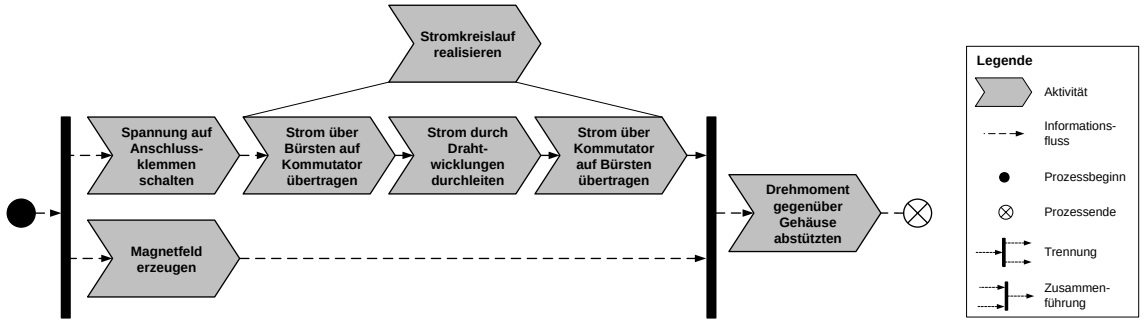


Bild A-8: Verhalten – Aktivitäten des LM_{GS} „Gleichstrommaschine“

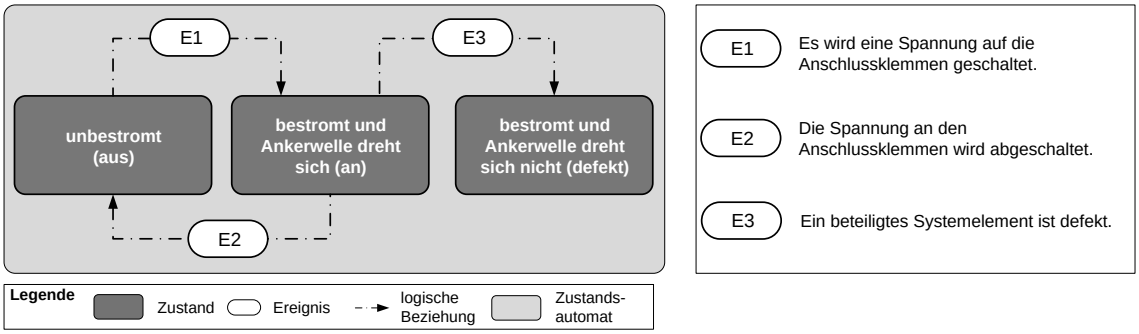


Bild A-9: Verhalten – Zustände des LM_{GS} „Gleichstrommaschine“

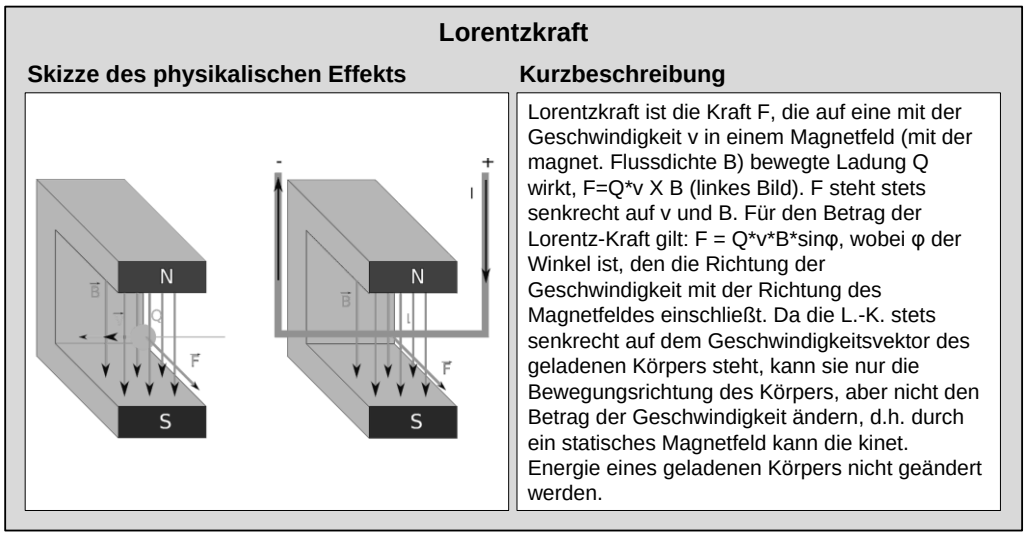


Bild A-10: Lösungsprinzipien des LM_{GS} „Gleichstrommaschine“

A2.2 LM_{KO} „Hybride Planung“

Gegenstand des LM_{KO} „Hybride Planung“ ist die Kombination klassischer Planungsverfahren mit Methoden zur Berücksichtigung kontinuierlicher Prozesse. In der Regel erfolgt dies durch eine Approximation der kontinuierlichen Prozesse bzw. durch ein Simulationsmodell und einer entsprechenden Planaktualisierung. Die Planung selbst kann dabei aus unterschiedlichen Planern bestehen. Ein diskreter Planer erzeugt einen Off-

line-Plan – also vor der Systemausführung. Je nach Einsatzbedingungen können noch weitere Planer, z.B. zur Kooperation mit anderen (Teil-)Systemen, eingesetzt werden.

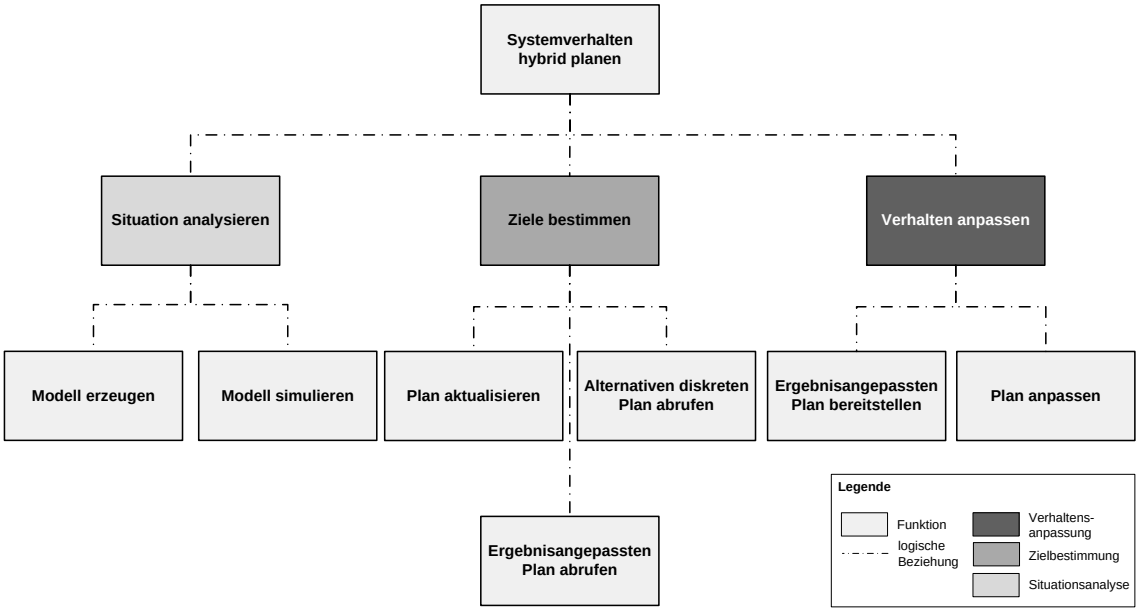


Bild A-11: Funktionen des LM_{KO} „Hybride Planung“

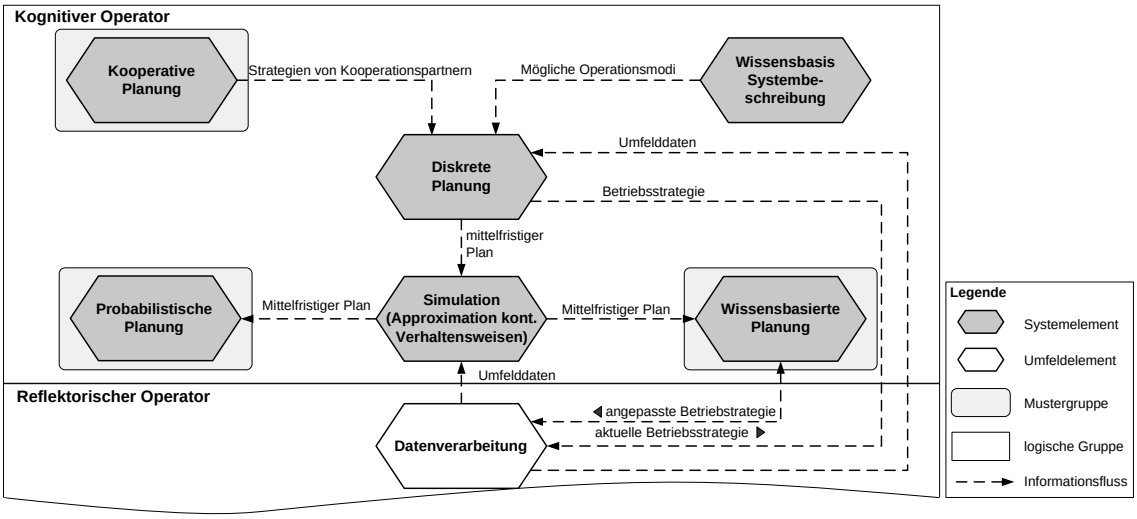


Bild A-12: Wirkstruktur des LM_{KO} „Hybride Planung“

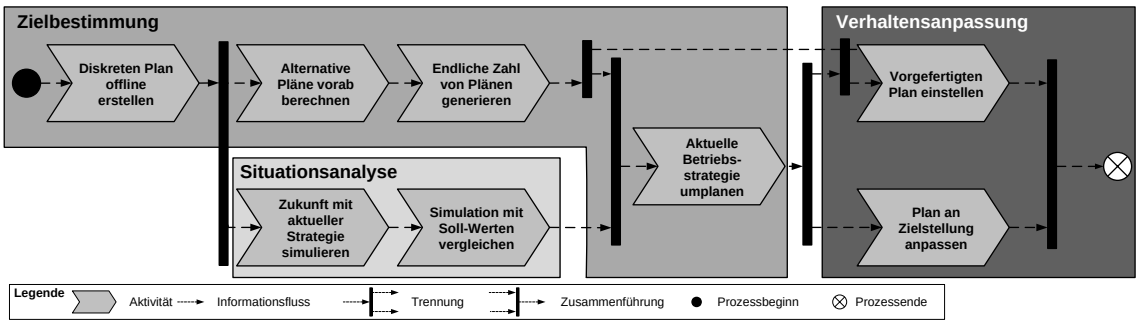


Bild A-13: Verhalten – Aktivitäten des LM_{KO} „Hybride Planung“

A2.3 LM_{KO} „Intelligente Vorausschau“

Gegenstand dieses Lösungsmusters ist eine Entscheidungsstrategie, auf deren Basis ein fortgeschrittenes mechatronisches System zukünftige, potentiell auftretende Einflüsse berücksichtigt. Ziel ist dabei die Optimierung des Einsatzes der Ressourcen und die Verbesserung des Systemverhaltens. Die Lösung des Entscheidungsproblems erfolgt mit Hilfe der Spieltheorie. Dabei handelt es sich um ein diskretes Optimierungsproblem. In einem Such- bzw. Spielbaum werden die Zielfunktionen des Systems optimiert, dabei stellen Umfeld und das System zwei Gegenspieler dar.

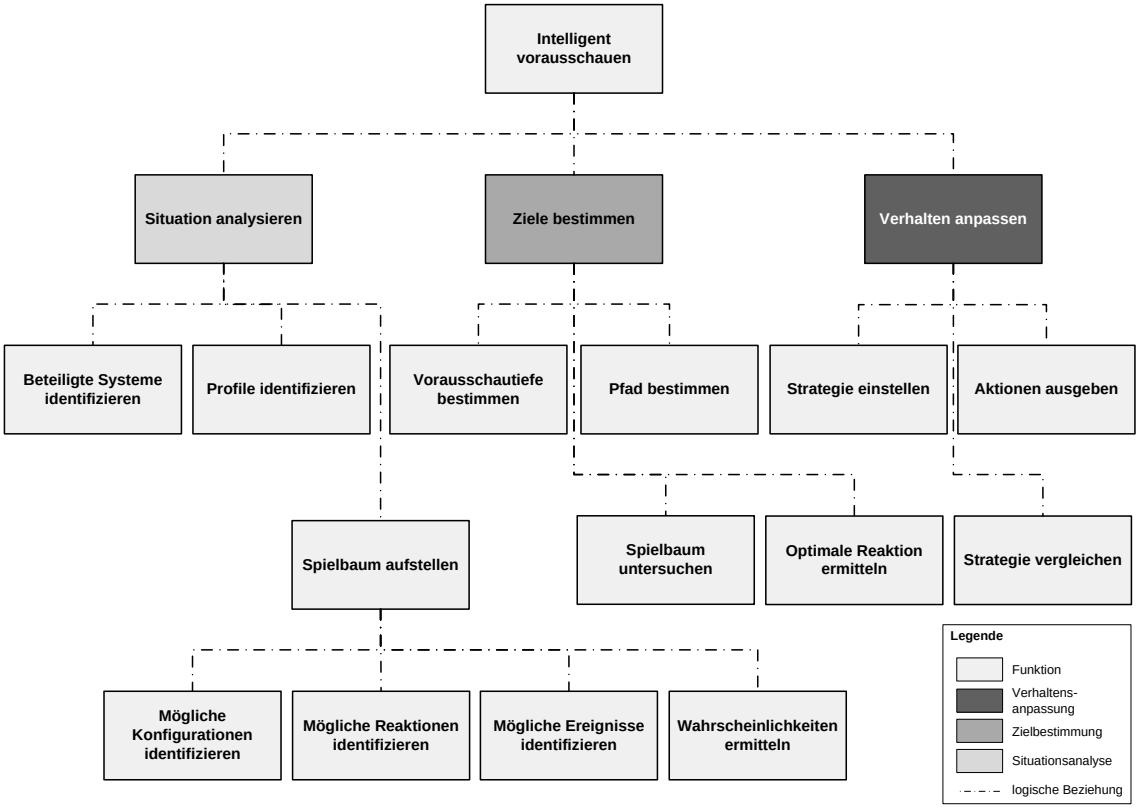


Bild A-14: Funktionen des LM_{KO} „Intelligente Vorausschau“

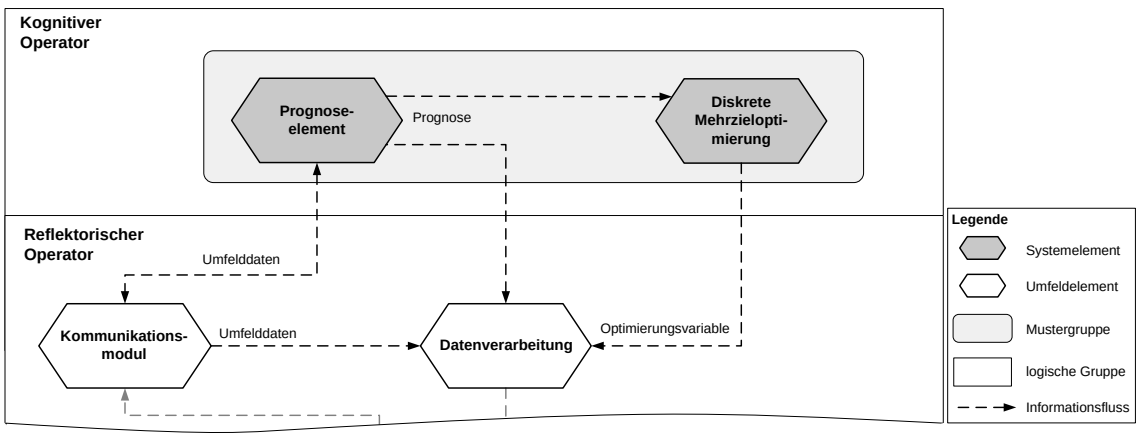


Bild A-15: Wirkstruktur des LM_{KO} „Intelligente Vorausschau“

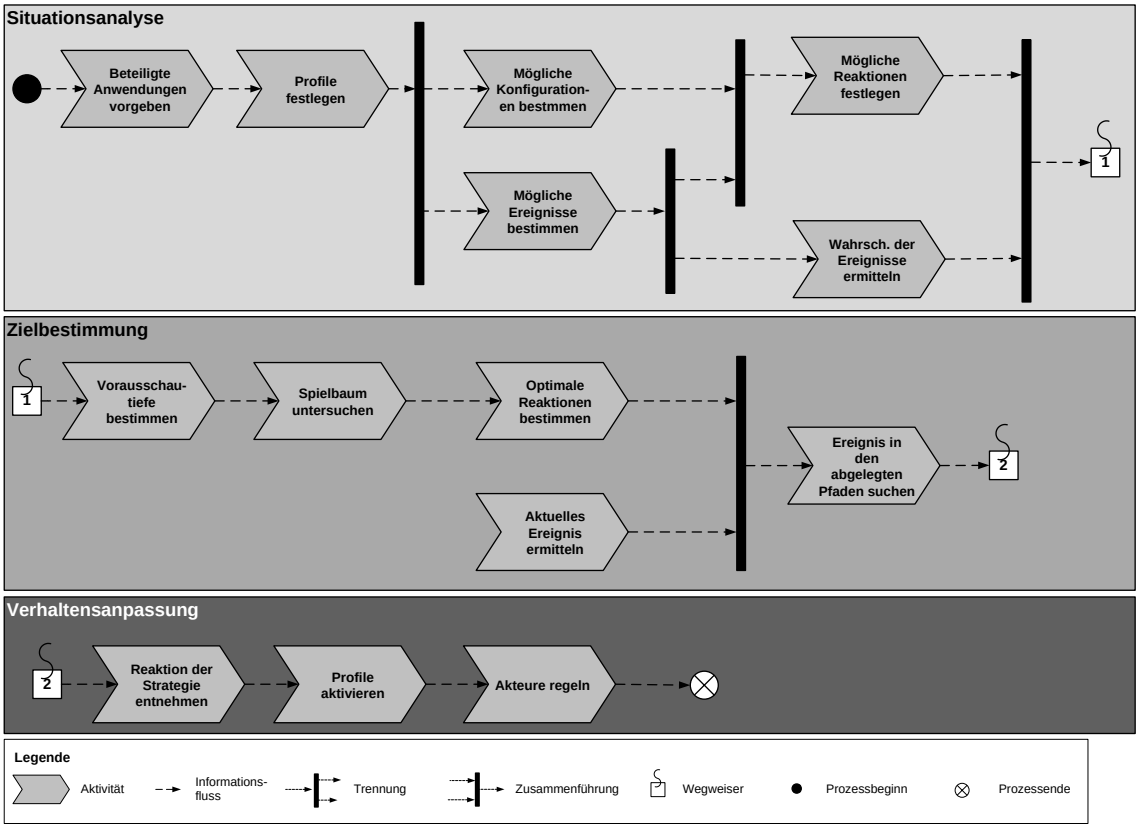


Bild A-16: Verhalten – Aktivitäten des LM_{KO} „Intelligente Vorausschau“

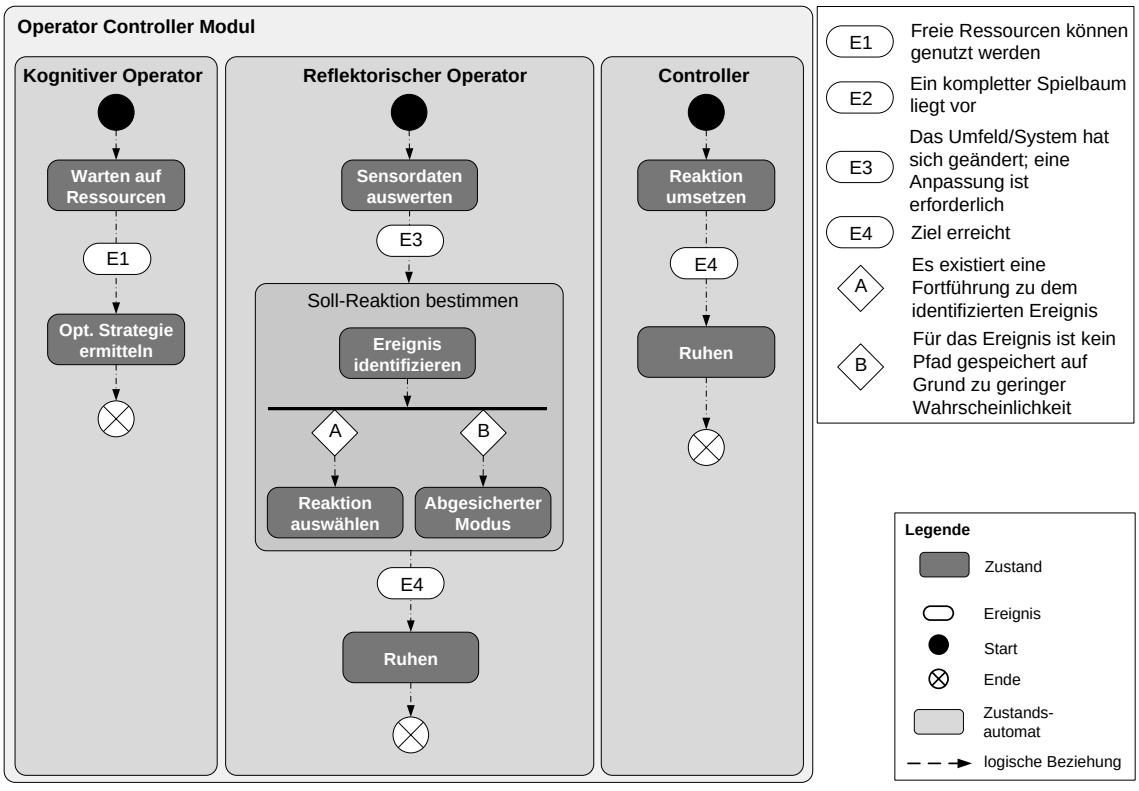


Bild A-17: Verhalten – Zustände des LM_{KO} „Intelligente Vorausschau“

A2.4 LM_{KO} „Kooperative Planung AMS“

Dieses Lösungsmuster kann eingesetzt werden, um Kooperationen zwischen mehreren fortgeschrittenen mechatronischen Systemen (AMS) abzustimmen. Dabei werden drei Kooperationsinteraktionen berücksichtigt: Abgabe von vollständigen Aufträgen bzw. Aufgaben, Abgabe von Teilfunktionen aus einer Aufgabe und Abstimmung der Aktivitäten zur Erreichung eines globalen Optimums.

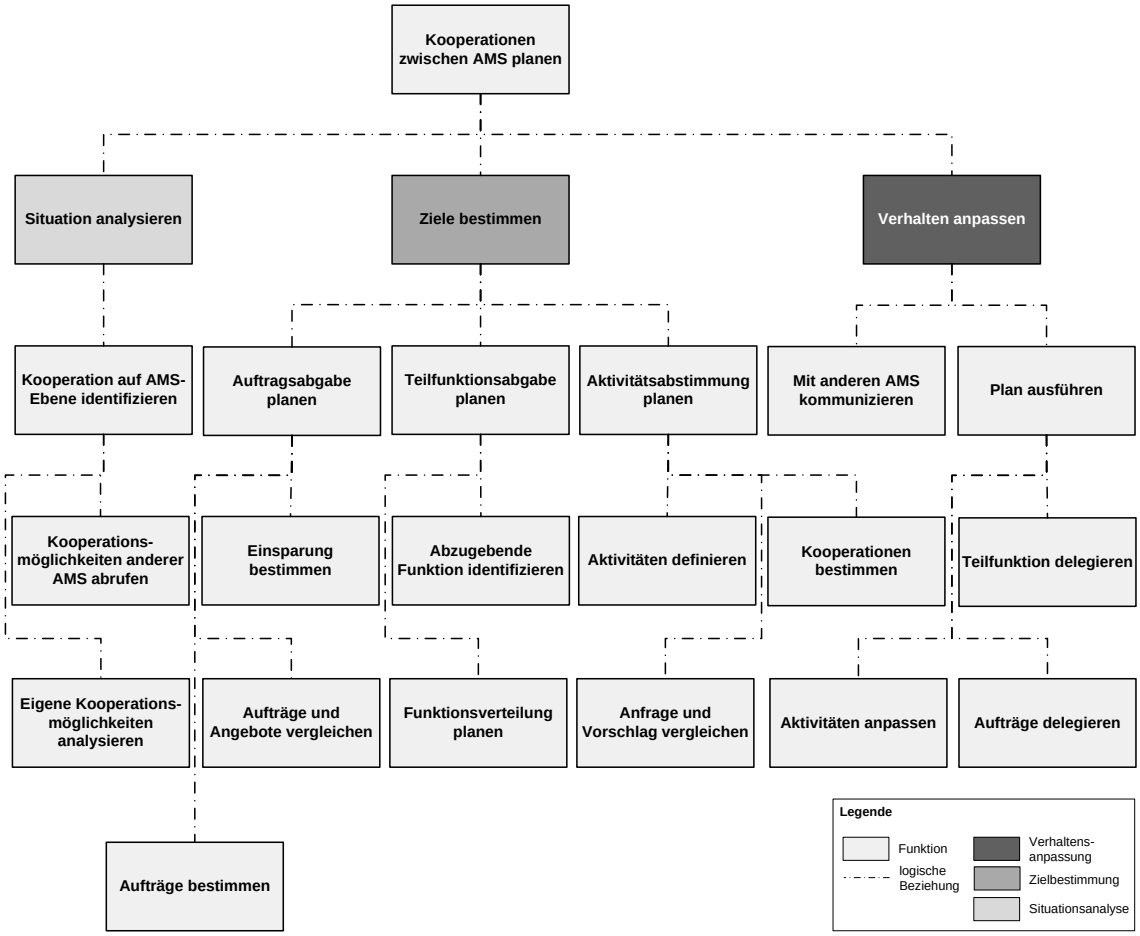


Bild A-18: Funktionen des LM_{KO} „Kooperative Planung AMS“

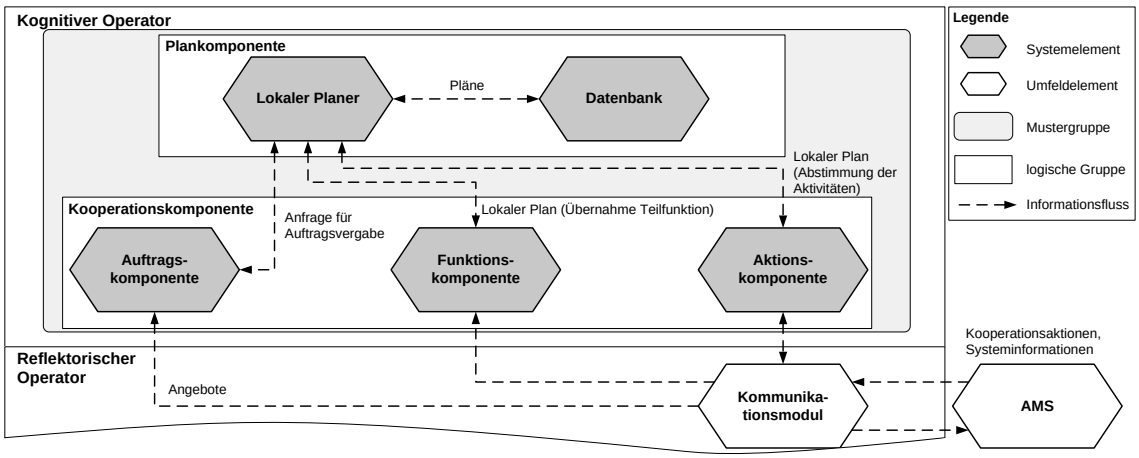


Bild A-19: Wirkstruktur des LM_{KO} „Kooperative Planung AMS“

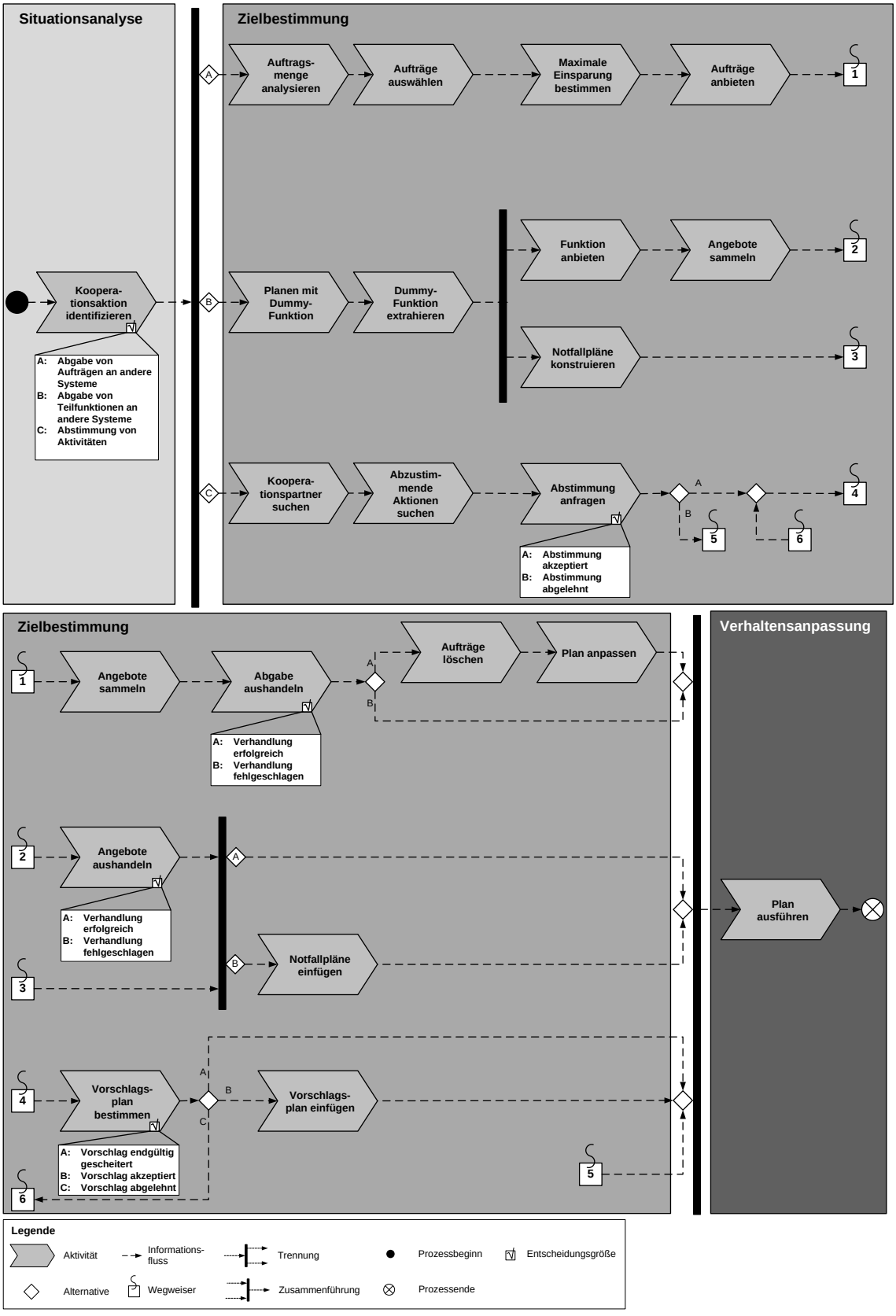


Bild A-20: Verhalten – Aktivitäten des LM_{KO} „Kooperative Planung AMS“

A2.5 LM_{KO} „Kooperative Planung MFM“

Gegenstand des LM_{KO} „Kooperative Planung“ ist die gemeinsame Interaktionen von mehreren mechatronischen Teilsystemen (MFM) innerhalb eines Gesamtsystems. Prinzipiell muss das Gesamtsystem nicht räumlich oder baulich in einer engen Beziehung stehen. Hierzu besitzt jedes MFM einen eigenen lokalen Planer. Dabei werden aber Abhängigkeiten modelliert, so dass eine Koordination der lokalen Pläne erfolgen kann.

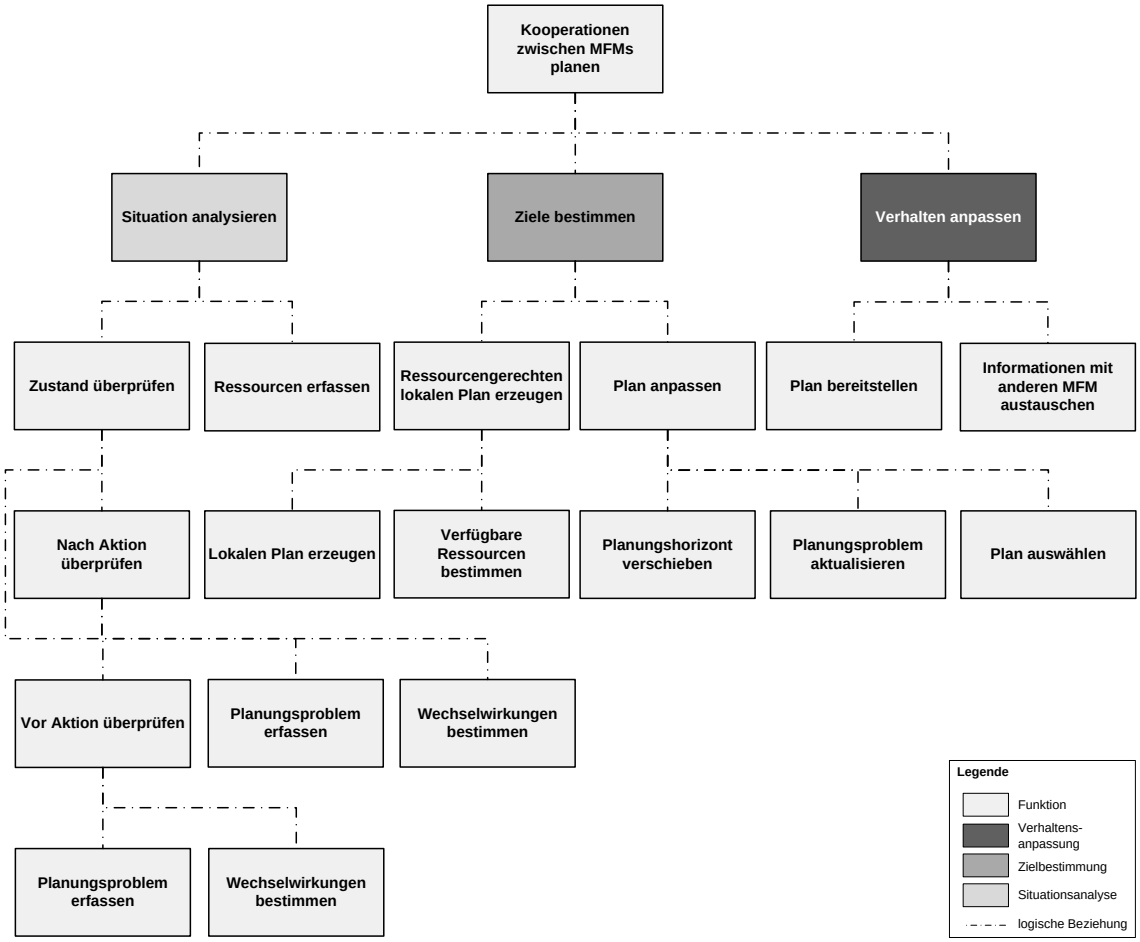


Bild A-21: Funktionen des LM_{KO} „Kooperative Planung MFM“

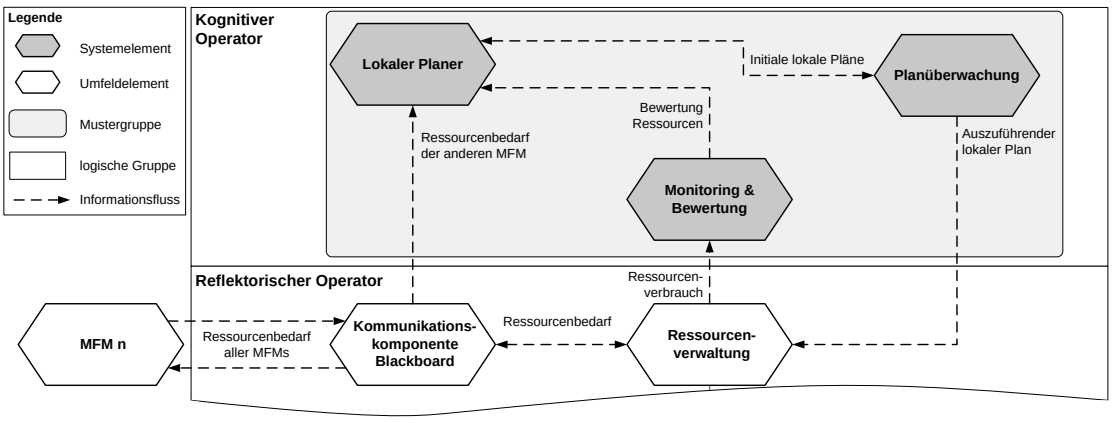


Bild A-22: Wirkstruktur des LM_{KO} „Kooperative Planung MFM“

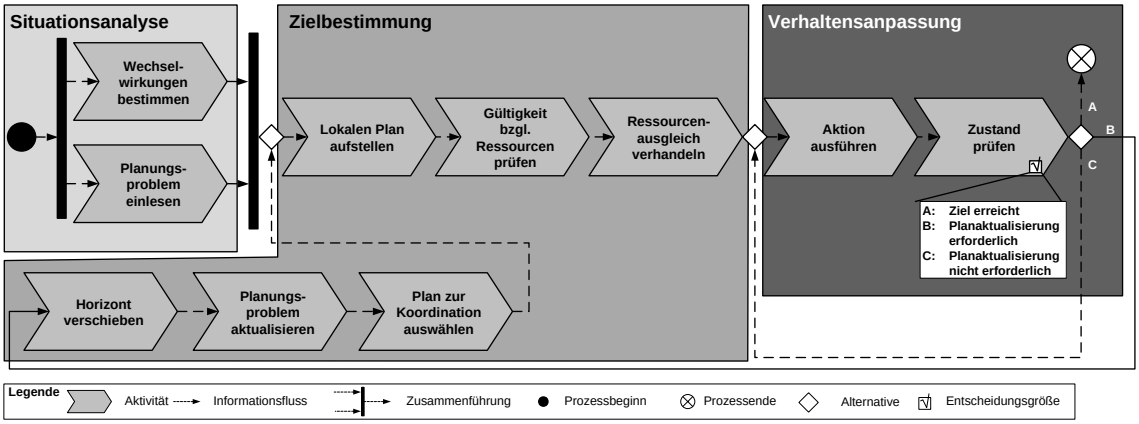


Bild A-23: Verhalten – Aktivitäten des LM_{KO} „Kooperative Planung MFM“

A2.6 LM_{KO} „Mehrzieloptimierung“

Dieses Lösungsmuster beschreibt die Optimierung mehrerer Zielfunktionen für kontinuierliche Optimierungsprobleme. Dabei handelt es sich um ein für mechatronische Systeme typisches Optimierungsproblem, das nicht durch ein globales Optimum gelöst werden kann. Die Lösung führt dabei zu einer Menge optimaler Kompromisse (Paretomenge). Aus dieser ist ein Paretopunkt mittels einer Entscheidungsheuristik auszuwählen. Dabei sind die Struktur und die bislang gewählten Paretopunkte zu berücksichtigen. Nachdem die Pareto-Parameter für die Zielfunktionen übergeben wurden, optimiert die numerische Pfadverfolgung die zeitabhängige, gewichtete Summe der Zielfunktionen nach der Optimierungsvariablen. Das Ergebnis ist eine Sollwertevorgabe für die Regelung.

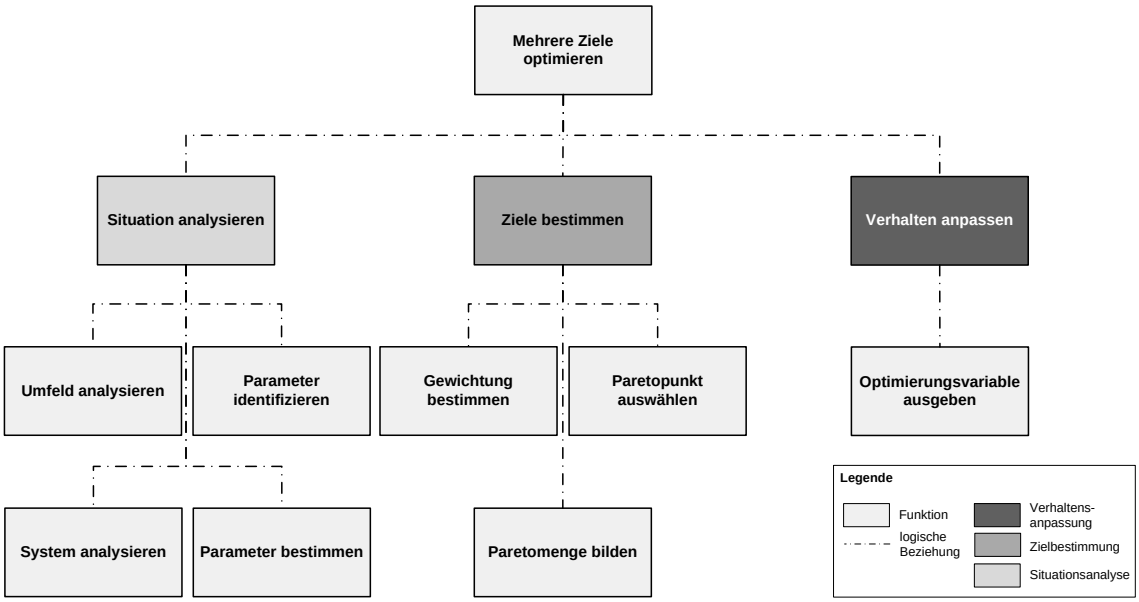


Bild A-24: Funktionen des LM_{KO} „Mehrzieloptimierung“

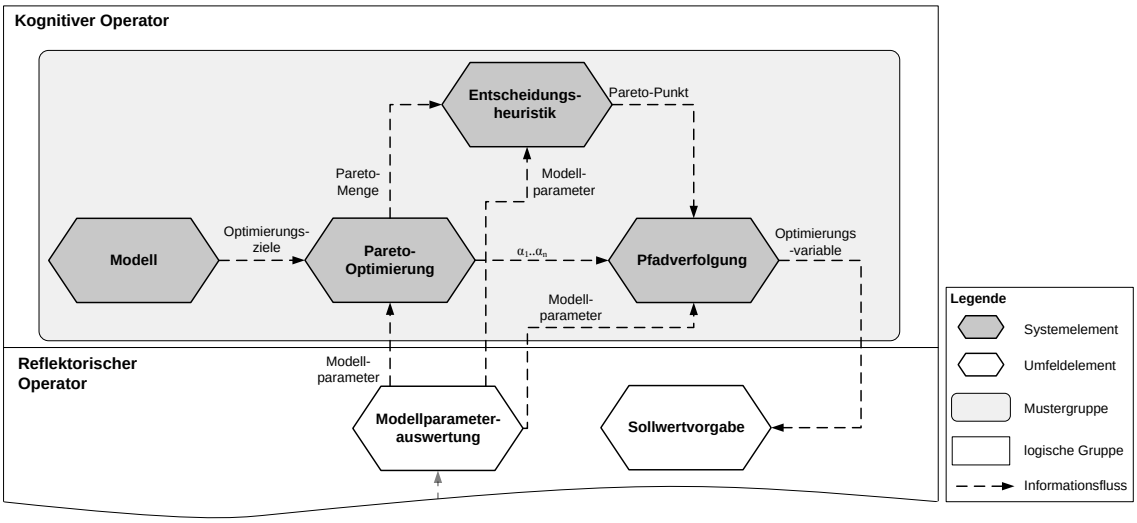


Bild A-25: Wirkstruktur des LM_{KO} „Mehrzieloptimierung“

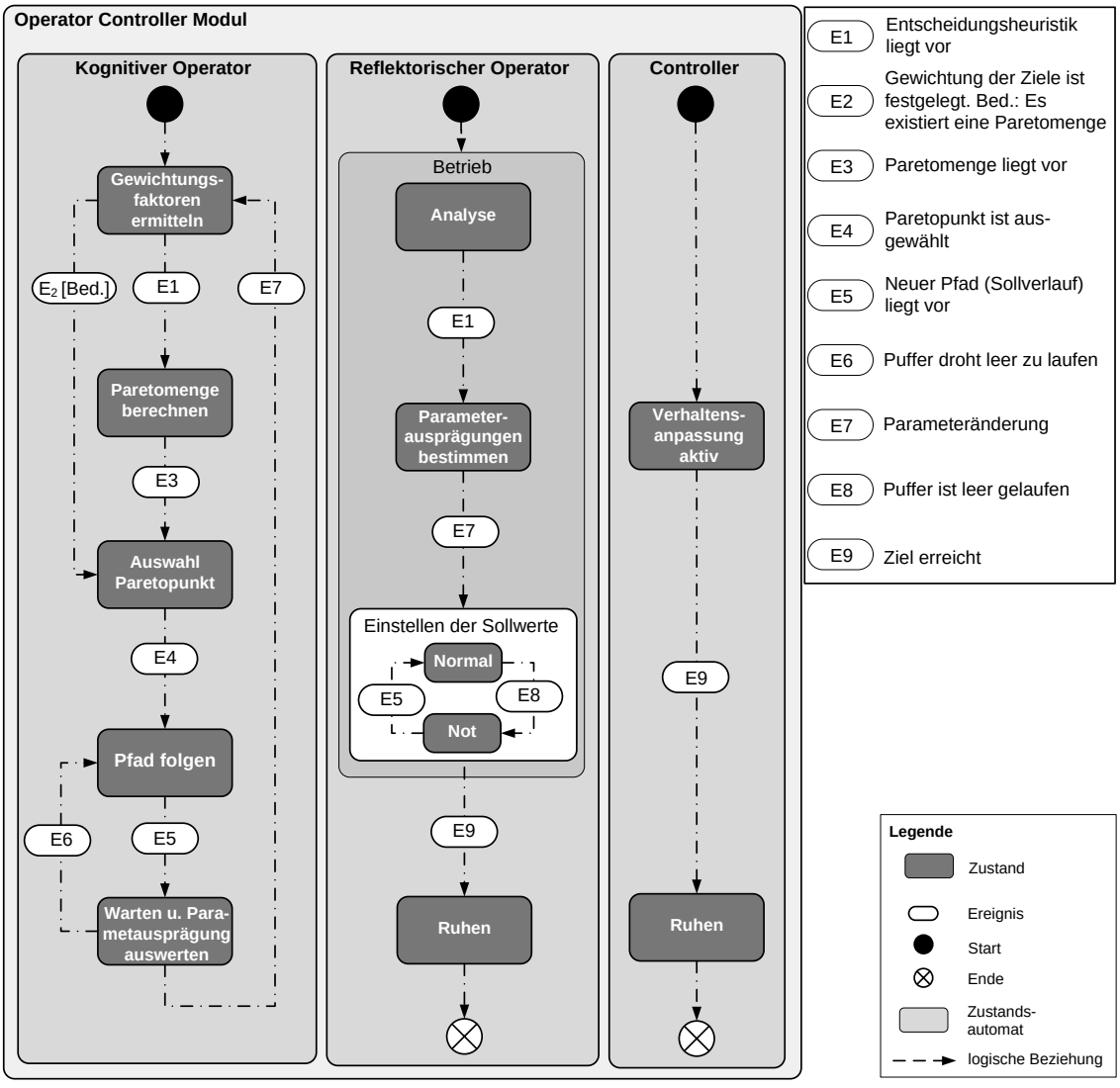


Bild A-26: Verhalten – Zustände des LM_{KO} „Mehrzieloptimierung“

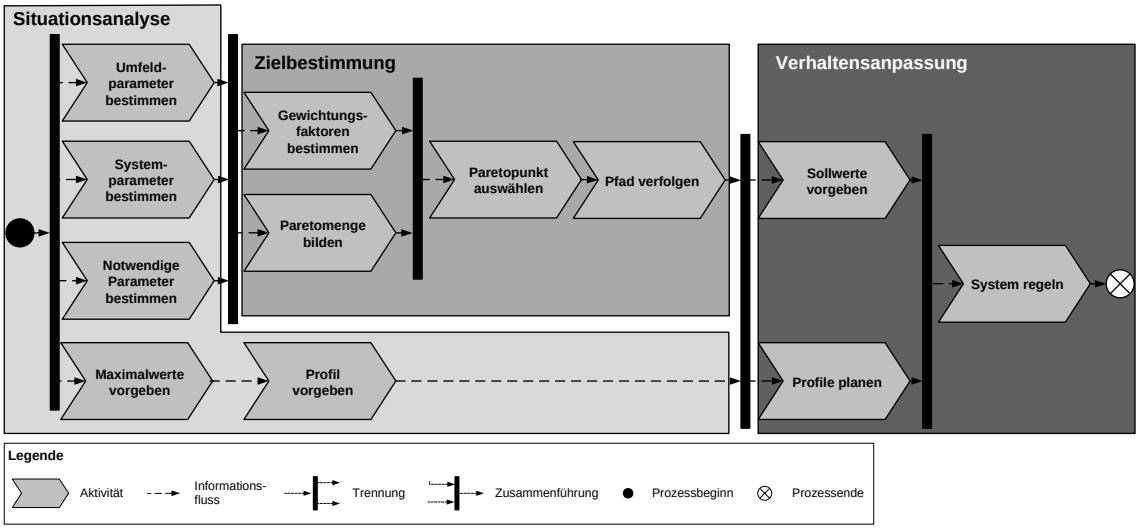


Bild A-27: Verhalten – Aktivitäten des LM_{KO} „Mehrzieloptimierung“

A2.7 LM_{KO} „Wissensbasierte Planung“

Die wissensbasierte Planung umfasst im Kern eine anwendungsbezogene initiale Planung und eine Lern-/Klassifikationskomponente. Ziel ist es, eine Rückfallebene auf Basis von gelernten Trainingsdaten zu erreichen. So ist zwar kein optimales Ergebnis möglich, aber zumindest ein sicheres Ergebnis. Hierzu werden aus Offline-Plänen Verhaltensregeln abgeleitet, um so einen wissensbasierten Plan zu erstellen.

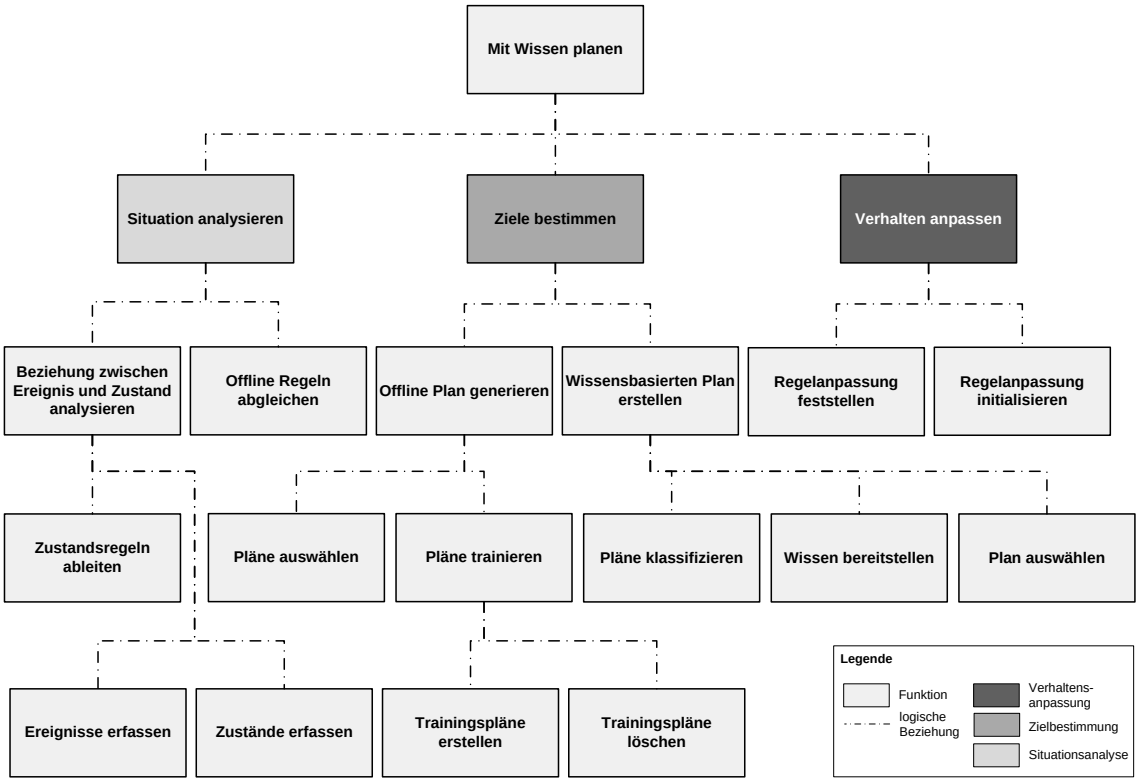


Bild A-28: Funktionen des LM_{KO} „Wissensbasierte Planung“

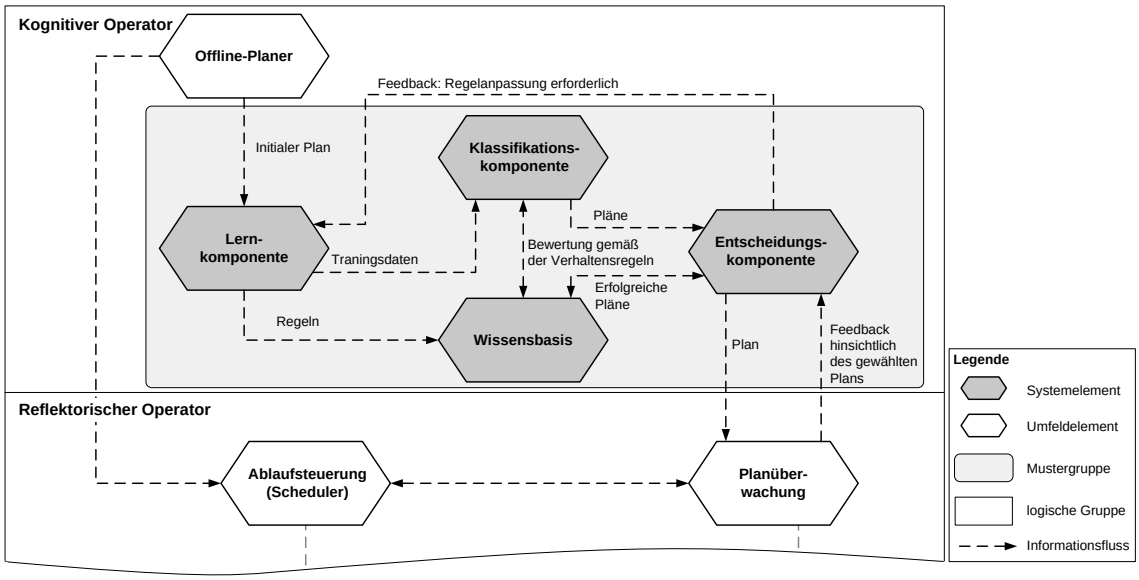


Bild A-29: Wirkstruktur des LM_{KO} „Wissensbasierte Planung“

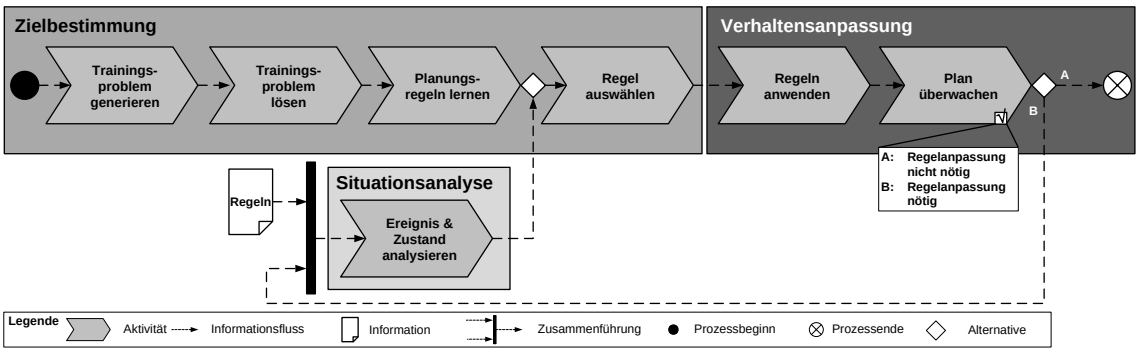


Bild A-30: Verhalten – Aktivitäten des LM_{KO} „Wissensbasierte Planung“

A3 Eingesetzte Verfahren in den LM_{KO}

Die folgende Tabelle zeigt die wesentlichen kognitionsrelevanten Verfahren im Überblick, die in den im Zuge dieser Arbeit dokumentierten Lösungsmustern für den kognitiven Operator aufgenommen wurden. Die Tabellen unterstützen die Spezifikation des Aspekts *Lösungsprinzipien*. Im Rahmen der Lösungsmusterwissensbasis wurden die verwendeten Verfahren in einer Kurzdarstellung abgelegt. Diese besteht aus einer Verfahrensskizze, einer Kurzbeschreibung, weiteren Informationen – im Wesentlichen Quellenangaben – und angehängten Dokumenten mit detaillierten Informationen. Ferner wird angezeigt, in welchen Lösungsmustern das Verfahren bereits verwendet wurde. Es werden mathematische Optimierungsverfahren (*kontinuierliche* und *diskrete*) sowie Verfahren der künstlichen Intelligenz (*Planverfahren* und *Lern-/ Klassifikationsverfahren*) unterschieden (vgl. Kap. 4.5.3).

Tabelle A-1: Eingesetzte Verfahren (Teil 1 von 2)

Verfahren	Verfahrensklasse	SO-Prozess	Eingesetzt in LM_{KO}
Auktionen	Planverfahren	Zielbestimmung	Kooperative Planung MFM
Branch-and-Bound	Diskrete Optimierungsverfahren	Zielbestimmung	Probabilistische Planung
Clusteranalyse	Lern-/ Klassifikationsverfahren	Situationsanalyse	Kooperative Planung AMS
Contract NetProtocol	Sonstige Verfahren	Zielbestimmung	Kooperative Planung AMS
Distributed Matchmaking	Sonstige Verfahren	Zielbestimmung	Kooperative Planung AMS
Entscheidungsbaum (Decision Tree)	Lern-/ Klassifikationsverfahren	Zielbestimmung	Probabilistische Planung Wissensbasierte Planung
Fuzzy-Approximation	Lern-/ Klassifikationsverfahren	Situationsanalyse	Hybride Planung
Graphbasiertes Planen	Planverfahren	Zielbestimmung	Hybride Planung Kooperative Planung AMS Kooperative Planung MFM Probabilistische Planung
Kuhn-Tucker-Verfahren	Kontinuierliche Optimierung	Verhaltensanpassung	Mehrzieloptimierung
Middle-Agents	Sonstige Verfahren	Zielbestimmung	Kooperative Planung AMS

Tabelle A-2: Eingesetzte Verfahren (Teil 2 von 2)

Verfahren	Verfahrensklasse	SO-Prozess	Eingesetzt in LM_{KO}
Naive-Bayes Klassifikator	Lern-/ Klassifikationsverfahren	Situationsanalyse	Kooperative Planung MFM Probabilistische Planung Wissensbasierte Planung
Numerische mengenorientierte Verfahren zur Approximation gesamter Pareto-Mengen	Kontinuierliche Optimierung	Zielbestimmung	Mehrzieloptimierung
Numerische Simulation	Kontinuierliche Optimierung	Situationsanalyse	Hybride Planung
Onlineplanen bzw. Echtzeitplanen	Planverfahren	Zielbestimmung	Hybride Planung
Parameterabhängige Verfolgung von paretooptimalen Lösungen	Kontinuierliche Optimierung	Zielbestimmung	Mehrzieloptimierung
Parameter-Lernen mit der Dirichlet-Verteilung	Lern-/ Klassifikationsverfahren	Situationsanalyse	Kooperative Planung MFM
Partiell ordnendes Planen	Planverfahren	Zielbestimmung	Hybride Planung Kooperative Planung AMS Kooperative Planung MFM Probabilistische Planung
Probabilistisches bedingtes Planen	Planverfahren	Zielbestimmung	Probabilistische Planung
Probabilistisches sensorloses Planen	Planverfahren	Zielbestimmung	Probabilistische Planung
Rückwärts-Zustandsraumsuche (Regressionsplanen)	Planverfahren	Zielbestimmung	Hybride Planung Kooperative Planung AMS Kooperative Planung MFM Probabilistische Planung
Schaeffer-Games	Diskrete Optimierung	Zielbestimmung	Intelligente Vorausschau
Spielbaum- bzw. Suchbaumanalyse	Diskrete Optimierung	Zielbestimmung	Intelligente Vorausschau
Strategic planning for imperfect-information games	Diskrete Optimierung	Zielbestimmung	Intelligente Vorausschau
Verfahren zur Bestimmung robuster Punkte	Kontinuierliche Optimierung	Zielbestimmung	Mehrzieloptimierung
Vorwärts-Zustandsraumsuche (Progressionsplanen)	Planverfahren	Zielbestimmung	Hybride Planung Kooperative Planung AMS Kooperative Planung MFM Probabilistische Planung