Weiterentwicklung der Autogenetischen Konstruktionstheorie (AKT)

Abschlussbericht zum Forschungsvorhaben VA 134/10-2

Prof. Dr.-Ing. Dr. h. c. Sándor Vajna
Dr.-Ing. André Jordan
Dipl.-Ing. Konstantin Kittel

30. Juli 2013

1 Allgemeines

Dieser Abschlussbericht enthält die Ergebnisse des oben genannten Forschungsvorhabens, welches von der Deutschen Forschungsgemeinschaft von Juni 2008 bis Dezember 2009 (VA 134/10-1) sowie als Folgeantrag von Juli 2010 bis Juli 2012 gefördert wurde (VA 134/10-2). Eine kostenneutrale Verlängerung des Vorhabens erfolgte bis zum Februar 2013. Der Antragssteller war Univ.-Prof. Dr.-Ing. Dr. h. c. Sándor Vajna, Inhaber des Lehrstuhl für Maschinenbauinformatik, Institut für Maschinenkonstruktion, Fakultät für Maschinenbau der Otto-von-Guericke-Universität Magdeburg. Als kostenneutraler Projektpartner arbeitete das Institut für Rechnergestützte Methoden im Maschinenbau an der Johannes-Kepler-Universität Linz (Österreich), Leitung Univ.-Prof. Dipl.-Ing. Dr. Klaus Zeman, im Vorhaben mit.

Der Abschlussbericht beschreibt in Kapitel 2 zunächst die Motivation und die Problemstellung. Im Kapitel 3 werden die Forschungsziele sowie die Forschungsmethodik dargestellt. Das Kapitel 4 enthält die Forschungsergebnissen und beantwortet die im Antrag formulierten Forschungsfragen. Eine Zusammenfassung und ein Ausblick auf weiterführende Forschungsarbeiten werden im Kapitel 5 gegeben.

2 Motivation und Problemstellung

Multidisziplinäre Produkte, wie sie heute z. B. in Form mechatronischer Produkte entwickelt werden, sind – verglichen mit „klassischen“ Produkten – durch eine deutlich größere Menge an produktbeschreibenden Informationen sowie durch eine domänenübergreifende Zusammenarbeit verschiedener Akteure gekennzeichnet. Die Mechatronik verbindet die Bereiche (mechanischer) Maschinenbau, Hydraulik/Pneumatik, Elektrik/Elektronik sowie Informationsverarbeitung. Mechatronische Produkte müssen im Hinblick auf eine

Vielzahl von Gerechtheiten hin entwickelt werden, so dass das Prinzip des *Design for X (DfX)* [MK05] für diese Produkte besonders gilt.

Die Entwicklung multidisziplinärer Produkte und die damit verbundenen Anforderungen werden durch die derzeit zur Verfügung stehende Produktentwicklungsmethodik nur unzureichend abgebildet. Ziel des aktuellen Forschungsvorhabens war es daher, die Entwicklung multidisziplinärer Produkte und Systeme zu unterstützen. Den Ausgangspunkt für die Untersuchungen bildet dabei die Autogenetische Konstruktionstheorie (AKT), die durch Bercsey und Vajna [BV94] mit der Feststellung der Analogie zwischen biologischer Evolution und dem Produktentwicklungsprozess begründet wurde. Die Entwicklung der AKT erfolgt mit dem Ziel, Erkenntnisse und Vorgehensweisen der biologischen Evolution auf die Entwicklung von Produkten zu übertragen, dabei Aktivitäten in der Produktentwicklung aus evolutionärer Sicht zu beschreiben und zu realisieren und daraus resultierende Tätigkeiten mit Hilfe geeigneter Rechnersysteme zu unterstützen. Ziel des aktuellen Forschungsvorhabens war es, die AKT so zu erweitern, dass sie auch die Entwicklung multidisziplinärer Produkte und Systeme und die damit verbundenen Anforderungen abbilden kann. Dazu sollte zum einen das bisher erarbeitete Produktbeschreibungsmodell der AKT weiterentwickelt und zum anderen der Prozess der Produktentwicklung durch eine von fixen, linearen Abläufen losgelöste und flexiblere Vorgehensweise beschrieben werden.

3 Forschungsziele und Forschungsmethodik

Wie oben dargestellt, bildete die von Bercsey und Vajna [BV94] festgestellte Analogie zwischen biologischer Evolution und der Produktentwicklung den Ausgangspunkt für die Untersuchungen. Analogievergleiche haben in der Wissenschaft eine lange Tradition. Der Analogieschluss steht nach Holz [Hol73, S. 61] (zitiert in [KBW73]) in der Mitte zwischen dem logisch strengen Vernunftschluss und einer Wahrscheinlichkeitsüberlegung. Eine Beweiskraft wird ihm jedoch nicht zuerkannt. Analogien bilden die loseste Form der Einheit und vageste Form der Erkenntnis [Kun98, S. 50].

Generell kann festgestellt werden, dass Analogien in der Produktentwicklung vielfältig Verwendung finden. Sie dienen als Kommunikationsmittel, fördern das Verständnis einer Problemstellung und zeigen neue Lösungswege auf. Analogie können helfen, Dinge besser zu verstehen, und sie sind für die Modellbildung relevant [Jor08, S. 60]. Aus diesen Gründen wurde für das aktuelle Vorhaben die Analogiebildung als Forschungsmethodik gewählt.

Die Vergangenheit zeigt, dass die Nutzung von Analogien häufig mit der Gefahr verbunden ist, Schlussfolgerungen, die aufgrund der Analogie gezogen werden, unbewusst auf Aspekte auszuweiten, die außerhalb der Gültigkeit der Analogie liegen. Dieser „Missbrauch“ von Analogien führt zu Verwirrung und Fehlinterpretationen. Er leitet letztlich das Denken in die Irre und mündet in falschen Schlussfolgerungen. Tiemann [Tie93, 61 ff.] benennt hierzu Beispiele aus der Entstehungsgeschichte bekannter physikalischer Modelle und Theorien.

Den Ausgangspunkt für die Bildung von Analogien stellen Assoziationen dar. Meist ist

es die zufällige Konfrontation mit einem äußeren Reiz, die eine Assoziation hervorruft. Wenn ein Modell oder eine Theorie auf einer Analogie aufbaut, ist es hilfreich, sich ein Gesamtbild von der gewählten Analogie zu verschaffen, das über die erste Assoziation hinausgeht. Dies ist einerseits notwendig, um weiterführende Ansätze aufzuzeigen und sich andererseits auch der Grenzen der Analogie bewusst zu werden. Letzteres ist wichtig, um unzulässige Analogieschlüsse zu vermeiden.

Der Aufbau eines vollständiges Analogiebildes für die AKT erfolgte in Anlehnung an das in [Jor08] beschriebene Vorgehen zum Aufbau eines Bionischen Analogiemodells in drei Schritten:

Schritt 1: Unabhängig für Natur und Produktentwicklung wurden Aspekte, die für diese Bereiche bedeutsam sind, zusammentragen. Da im Forschungsvorhaben sowohl das Produktmodell der AKT als auch das Vorgehensmodell im Fokus standen, wurde zwischen produkt- und prozessrelevanten Aspekten unterschieden.

Schritt 2: Die gefundenen Aspekte wurden einander gegenübergestellt. Die Zusammenstellungen sind in den Tabellen 1 und 2 auf den Seiten 4 und 5 dargestellt. Aspekte, die für den Bereich „Natur“ als typisch gefunden wurden, wurden auf der Produktentwicklungsseite ergänzt und umgekehrt.

Schritt 3: Die gefundenen Aspekte wurden analysiert, um Gemeinsamkeiten und Unterschiede herauszustellen.

Die Zusammenstellungen verdeutlichen, dass es sowohl bei den produkt- als auch bei den prozessrelevanten Aspekten deutliche Unterschiede zwischen Natur und Produktentwicklung gibt (vgl. hierzu auch [KT02]; [MR98]; [Nac98]). Der direkte Vergleich wirft die Frage auf, warum es diese Unterschiede gibt und welche Ansätze der Natur sich für die Produktentwicklung übernehmen lassen. Zur Beantwortung dieser Frage ist folgendes zu klären:

- Welche Bedeutung haben die Aspekte in den Bereichen Natur und Produktentwicklung und welche Wechselwirkungen bestehen zwischen ihnen?
- Welche Vor- und Nachteile haben die gefundenen Lösungsansätze?
- Welche Voraussetzungen müssen erfüllt sein, um die Lösungsansätze der Natur der Produktentwicklung zugänglich zu machen?

Die Zusammenstellungen in den oben genannten Tabellen zeigen, dass die Unterschiede zum Teil so gravierend sind, dass eine Übernahme der Ansätze in die Produktentwicklung dort zu einem Paradigmenwechsel führt. Die unterschiedlichen Wechselwirkungen von Material und Struktur in Natur und Produktentwicklung sind ein Beispiel hierfür. Technische Werkstoffe, wie Metalle und Kunststoffe, sind in der Regel homogener Natur und in ihren Eigenschaften eindeutig definiert. Dem Produktentwickler ist es daher möglich, Struktur und Material getrennt zu betrachten. Er entwirft z. B. eine Struktur, „füllt“ diese mit einem Material und überprüft in einem dritten Schritt, ob die Struktur

Tabelle 1: Gegenüberstellung produktrelevanter Aspekte

Aspekt	**Natur**	**Produktentwicklung**
Elemente		
Farbe	entstanden	vorherbestimmt
Form	entstanden	vorherbestimmt
Struktur und Material	oft nicht unterscheidbar	klar getrennt
System		
Struktur	vernetzt	linear, hierarchisch
Abgrenzbarkeit	fließend, kontinuierlich	monolithisch oder modular
Verhalten		
Funktion	entstanden	vorherbestimmt
Komplexität	aktiv genutzt	vermeiden, umgehen
Anpassbarkeit	multifunktional	auf einen Verwendungszweck hin optimiert
Ökologie		
Instandhaltbarkeit	Selbstheilung	Reparatur / Austausch
Nachhaltigkeit	nachhaltig	Gestaltungsziel
Recyclebarkeit	recyclebar	Gestaltungsziel
Ergonomie		
Handhabbarkeit	Nutzer passt sich evolutionär an Produkt an	Produkt wird zielgerichtet an Nutzer angepasst
Zuverlässigkeit	nicht relevant	Gestaltungsziel
Sicherheit	nicht relevant	Gestaltungsziel
Haltbarkeit	zeitlich begrenzt	(vage) vorherbestimmt
Wirtschaftlichkeit		
Mehrwert	Nebeneffekt, entsteht evolutionär durch Interaktion vernetzter Lösungen	Gestaltungsziel
Rentabilität	Nebeneffekt, entsteht evolutionär durch Interaktion vernetzter Lösungen	Gestaltungsziel

den Belastungen standhält. Ist dies nicht der Fall, hat er die Möglichkeit, die Struktur zu verändern oder ein anderes Material einzusetzen. In der Natur hingegen sind Struktur und Material oft nicht exakt voneinander zu trennen. Ein Beispiel hierfür sind Hölzer, Muschelschalen oder Knochen. Sie sind – im Gegensatz zu technischen Produkten – auf mehreren Ebenen strukturiert. Neben der äußeren Struktur (der Gestalt) weisen sie hierarchisch gegliederte Mikrostrukturen auf, die das Verhalten maßgeblich beeinflus-

Tabelle 2: Gegenüberstellung prozessrelevanter Aspekte

Aspekt	Natur	Produktentwicklung
Strukturbildung	selbstorganisiert	fremdorganisiert
Entwicklungsrichtung	zukunftsblind	zielorientiert
Komplexität	aktiv genutzt	vermeiden, umgehen
Ablauf	vernetzt	(weitgehend) linear
Steuerung	negativ rückgekoppelt	geregelt
Organisationsform	dezentral	zentral

sen und die sich von der Materialstruktur nicht abgrenzen lassen. Das Bild 1 zeigt die innere Struktur eines Oberschenkelhalsknochens. Der Oberschenkelknochen ist ein Röhrenknochen, der im Bereich des Hüftgelenkes mit einer schaumartigen Struktur gefüllt ist.

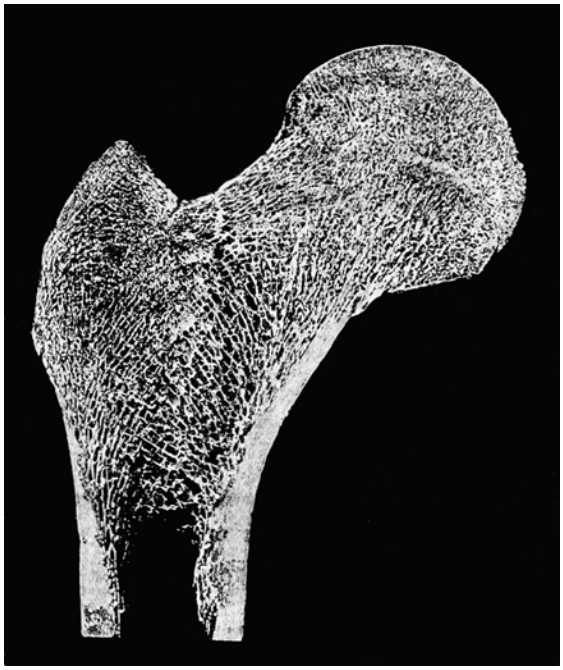

Bild 1: Innere Struktur des Oberschenkelhalsknochen [You93, S. 753]

Die Ausrichtung der inneren Struktur eines Knochens orientiert sich an den Belastungen, denen er ausgesetzt ist. Sie unterliegt einem permanenten Umbau [RM81]. Dies unterscheidet Knochen von technischen Produkten, deren Struktur während der Produktentwicklung festgelegt wird und die bis zum Ende des Produktlebens weitgehend unverändert bleibt. Die Substanz eines Knochens, seine Struktur und sein Entstehungsprozess bilden eine Einheit. Die Übertragung von einzelnen Aspekten, wie z. B. der inneren Struktur, ist daher in der Regel ohne eine Betrachtung der Wechselwirkungen

zwischen den Aspekten nicht möglich. Das oben beschriebene Vorgehen zur Bildung von Analogien soll helfen, die gegenübergestellten Sachverhalte in geeigneter Weise voneinander abzugrenzen und die Auswirkungen auf die einzelnen Unternehmensbereiche sichtbar zu machen.

Eine Schwierigkeit beim Aufstellen des ganzheitlichen Analogiebildes bestand zum Teil darin, Aspekte der Natur mit sinnvollen Inhalten der Produktentwicklung zu füllen und umgekehrt typisch technische Aspekte aus Sicht eines biologisches Systems zu interpretieren (siehe oben Schritt 2 der Analogiebildung). So sind z. B. die Aspekte *Sicherheit* und *Handhabbarkeit* für die Produktentwicklung sehr wichtig. Sicherheit beschreibt einen Zustand, der frei von unvertretbaren Risiken und Gefahrenquellen ist oder der als gefahrenfrei angesehen wird. Handhabbarkeit ist ein Maß für die Leistungsfähigkeit, die Gebrauchstauglichkeit und die Güte der Benutzungsschnittstellen in Bezug auf Produktziele, Anwendungskontext und Nutzungsmerkmale. Beides sind in der Produktentwicklung Gestaltungsziele, in deren Zentrum der Nutzer steht.

Für den Aufbau eines ganzheitlichen Analogiebildes ergibt sich hieraus ein Problem, weil es in der Natur keinen Nutzer im technischen Sinne gibt. Interpretiert man die genannten Aspekte jedoch etwas freier, so lässt sich durchaus ein Vergleichsmaßstab finden: Die Handhabbarkeit beschreibt im weiteren Sinn, wie gut Produkt und Nutzer aufeinander abgestimmt sind. Eine solche Feinabstimmung ist auch oft zwischen biologischen Systemen zu finden. Im Unterschied zur Technik wird hier aber nicht das Produkt an den Nutzer angepasst, vielmehr entwickelt sich der „Nutzer“ evolutionär derart, dass er das „Produkt“ optimal nutzen kann.

Analoges gilt für die Sicherheit. Ein technisches Produkt so zu gestalten, dass von ihm keine Gefährdung für den Nutzer ausgeht, ist Aufgabe der Produktentwicklung. Ein biologisches System hingegen wird evolutionär selbst einen Schutz gegen häufig auftretende Gefahren entwickeln, so dass sich das Risiko eines Schadens verringert. Wiederum ist es hier der Betroffene, der sich anpasst.

Die oben genannten Beispiele zeigen, dass nicht alle Aspekte des Analogiebildes übertragbar sind. Sie verdeutlichen die Grenzen der gewählten Analogie. Die Zusammenstellungen in den Tabellen 1 und 2 eignen sich jedoch, um Strategien und Gegebenheiten der Produktentwicklung zu hinterfragen und sie regen an, Alternativen zu entwickeln. Dabei ist für jeden Aspekt zu prüfen, ob eine analoge Übertragung des Lösungsansatzes der Natur in die Produktentwicklung sinnvoll ist, ob die etablierten Lösungen der Technik beibehalten oder alternative (nicht natur-analoge) Lösungen entwickelt werden müssen.

4 Forschungsergebnisse

Ein Teil der Forschungsergebnisse wurde bereits im Zwischenbericht [VK10] veröffentlicht. Aus diesem Grund werden an dieser Stelle nur die Ergebnisse dargestellt, die seit September 2009 hinzugekommen sind oder die eine Änderung erfahren haben.

Zunächst werden in Abschnitt 4.1 einige Begriffe erläutert. Dies ist notwendig, da im Zuge der Forschungsaktivitäten deutlich wurde, dass die Begriffe *Sichtweise*, *Aspekt* und *Attribut*, die in bisherigen Arbeiten zum Teil synonym gebraucht wurden, unterschiedli-

che Bedeutungen haben und daher voneinander abgegrenzt werden müssen.

Zum besseren Verständnis der Forschungsergebnisse ist es zudem hilfreich, den Zusammenhang von Produktentwicklung und Komplexität zu beleuchten. Im Abschnitt 4.2 wird daher die Bedeutung der Produktentwicklung als komplexes System diskutiert. Darauf aufbauend werden in den Abschnitten 4.3 und 4.4 die Forschungsergebnisse zur Erweiterung des Produktmodells dargestellt und das Vorgehensmodell beschrieben. Die Ergebnisse zum Stand der Forschung im Bereich *Design Spell Checker* werden im Abschnitt 4.5 gezeigt.

4.1 Begriffe

Ein Modell ist ein unvollständiges Abbild der Wirklichkeit, das die wesentlichen Eigenschaften seines Vorbildes repräsentiert [Sta73]. Modelle helfen einerseits Sachverhalte oder Objekte zu erklären und andererseits Theorien zu testen. Ein Produktmodell ist der ersten Kategorie zu zuordnen. Es beschreibt und repräsentiert ein Produkt zu einem Zeitpunkt, zu dem das Produkt selbst noch nicht existiert. Es ermöglicht Vorhersagen und Simulationen des späteren Produktverhaltens. Die Beurteilung und Bewertung des Produktverhaltens erfolgt auf der Basis von *Attributen*. Attribute sind folglich im Sinn der Modelltheorie die Eigenschaften, die sich das Produktmodell und das Produkt selbst teilen.

Im Zwischenbericht [VK10] wurden die Attribute als *Sichtweisen* bezeichnet. Das Sichtweisenkonzept entstand im Forschungsgebiet *Integrated Design Engineering (IDE)*, wo die Sichtweisen die Betrachtung des Produktes unter einem bestimmten Gesichtspunkt beschrieben. Die Sichtweisen wurden hier zeitweise auch als *Aspekte* bezeichnet. Durch neue Erkenntnisse in der IDE wurde der Begriff der *Sichtweise* durch den Begriff *Attribut* ersetzt. Um eine Konsistenz der Begriffe zwischen den einzelnen Forschungsgebieten zu gewährleisten, wird der Attributsbegriff auch in der AKT für die Beschreibung des Produktmodells übernommen. Der Begriff der Sichtweise wird hingegen, wie nachfolgend dargestellt werden wird, für die Beschreibung der Partialmodelle genutzt.

Das Produktmodell besteht aus einer Vielzahl von Parametern. Die Parameter sind nicht nur geometrischer Natur, sondern nehmen darüber hinaus ein Vielzahl weiterer Daten und Informationen auf. So könnten im Modell z. B. die folgenden Parameter existieren:

- `Letzte_Aenderung_der_Anforderungsliste = 2013-06-13`
- `Motor_Leistung = 5 kW`
- `Design_Entwurf_Skizze = gehaeuse_deckel.jpg`
- `CAD_File_Name = AR_343.672.1_gehauese_deckel.prt`
- `Entwicklungsstand_Der_Firmware = 0.4.3-0623`
- `Freigabe_Durch_Auftraggeber = wahr`

Für die Bearbeitung bestimmter Teilprobleme benötigt der Produktentwickler nur einen Teil dieser Parameter. Die einzelnen Domänen der Mechatronik beschreiben das Produkt auf unterschiedliche Art und Weise. Es gibt Parameter, die nur in einer Domäne definiert und auch nur hier benötigt werden. Andere Parameter spielen domänenübergreifend eine wichtige Rolle.

Die Parameter, die in einem Anwendungsgebiet benötigt werden, lassen sich in Anlehnung an die erweiterte Featuredefinition der FEMEX [VW97] zu einer Struktur zusammenfassen. Ein so entstehendes Feature kann auch als Partialmodell und als Optimierungsobjekt verstanden werden, dessen Ausprägung direkt durch die unabhängigen Parameter oder indirekt durch die abhängigen Parameter beeinflusst wird. Die unabhängigen Parameter werden als *Designparameter* bezeichnet. Produktparameter, die für die Bewertung der Ausprägung eines Partialmodells genutzt werden, werden *Bewertungskriterien* genannt.

Welche Parameter unabhängig sind und welche von anderen beeinflusst werden, ist zum einen von der Problemstellung und zum anderen vom Entwicklungszeitpunkt abhängig. Ein Parameter kann also in einem Partialmodell für die Bewertung herangezogen werden, in einem anderen Modell jedoch als Designparameter fungieren. Die Partialmodelle sind dabei hierarchisch strukturiert und überlappen einander. Da die Partialmodelle in unterschiedlichen Anwendungskontexten entstehen und verschiedene Sichten auf das Produktmodell darstellen, werden sie im Folgenden auch als *Sichtweisen* bezeichnet.

Im Zwischenbericht [VK10] wurden die Elemente des Lösungsraumes in Designparameter, Merkmale und Eigenschaften unterschieden. Die Designparameter wurden in Form einer Matrix strukturiert, in der eine Zuordnung der Parameter in der horizontalen Achse zu den Attributen (im Zwischenbericht noch als Sichtweisen bezeichnet) und auf der vertikalen Achse den verschiedenen Eigenschaften und Merkmalen erfolgte. Ein Merkmal ist nach Weber [Web05] etwas, das die Struktur, die Gestalt und die Beschaffenheit eines Produktes charakterisiert und das vom Produktentwickler beeinflusst werden kann. Eigenschaften hingegen definieren ein bestimmtes Verhalten des Produktes und können vom Produktentwickler nicht direkt beeinflusst werden. In der AKT wird das Produktverhalten – wie oben darstellt wurde – durch seine Attribute beschrieben, deren Ausprägung der Produktentwickler mit Hilfe der Produktparameter beeinflusst. Aus diesem Grund wurde die im Zwischenbericht gewählte Darstellung zugunsten der oben beschriebenen aufgegeben.

4.2 Produktentwicklung und Komplexität

Die Aufgabe der Produktentwicklung ist die Erstellung eines Produktmodells, das, wie oben dargestellt, das Produkt in seinen wesentlichen Eigenschaften charakterisiert. Die Anforderungen, die an Produkte gestellt werden, sind in den letzten Jahren kontinuierlich gestiegen. Aus systemischer Sicht ist der Markt komplexer geworden. Komplexität ist eine Systemeigenschaft. Die Elemente komplexer Systeme wechselwirken in vielfältiger Weise miteinander. Dies führt zu einer Vielzahl möglicher Systemzustände. Die Anzahl der Zustände, die ein System einnehmen kann, wird als *Varietät* bezeichnet und stellt ein Maß für die Komplexität dar [Mal98, S. 6]. Das Gesamtverhalten komplexer Systeme

kann nicht eindeutig beschrieben werden, da sie emergente Eigenschaften ausbilden, die sich nicht aus den Eigenschaften ihrer Elemente ableiten lassen [McK04]. Komplexe Systeme verwehren sich einer Vereinfachung und bleiben vielschichtig [Cil98].

Es besteht ein Zusammenhang zwischen der Handlungsvarietät eines Systems und seiner Fähigkeit, ein anderes System zu steuern. Ashby [Ash56] hat hierzu ein Gesetz formuliert: Je größer die Varietät eines Systems ist, desto mehr kann es die Varietät seiner Umwelt durch Steuerung vermindern („Ashbysches Gesetz“). Wendet man dieses Gesetz auf die Systeme „Markt“, „Produkt“, „Produktmodell“, „Produktentwicklung“ und „Unternehmen“ an, so ergibt sich die in Bild 2 dargestellte Wirkungskette.

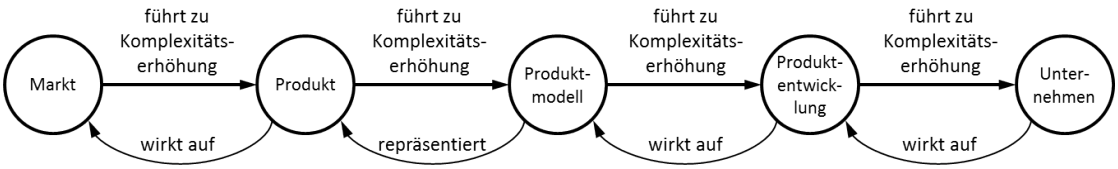

Bild 2: Komplexitätserhöhung als Wirkungskette

Wird das System „Markt“ komplexer, müssen auch die Systeme, die den Markt beeinflussen wollen – die Produkte – komplexer werden. Komplexere Produkte erfordern wiederum komplexere Produktmodelle, da diese ja die Produkte repräsentieren. Komplexere Produktemodelle führen zu einer Komplexitätserhöhung der Produktentwicklung, denn diese müssen die Produktmodelle beherrschen. Letzten Endes müssen jedoch die Unternehmen selbst komplexer werden, um angemessen auf die Anforderungen des Marktes reagieren zu können.

Soll die Produktentwicklung komplexer und ihre Handlungsvarietät vergrößert werden, müssen Strategien, Methoden und Werkzeuge gefunden werden, die der wachsenden Komplexität gerecht werden[1]. Bislang gilt es z. B. als wünschenswert, Strukturen und Prozesse der Produktentwicklung bis ins Detail zu definieren, ihre Zustände zu überwachen und bei einer Abweichung von Ist und Soll steuernd einzugreifen. Dies resultiert aus dem Denken, dass Dinge nur dann optimal gestaltet und gesteuert werden können, wenn sie zuvor *vollständig* verstanden worden sind. Für einfache Systeme mit einer überschaubaren Anzahl von Zuständen bzw. Aktionen gilt dies zweifellos. Komplexe Systeme kennzeichnet jedoch eine sehr hohe Varietät, die eine Gestaltung und Steuerung „von oben herab“ unmöglich macht.

Anregungen zum Umgang mit Komplexität liefert die Natur. Die in Kapitel 3 vorgestellten Tabellen 1 und 2 zeigen, dass es trotz der Analogie zwischen Produktentwicklung und Natur deutliche Unterschiede gibt. Dabei ist festzustellen, dass die Eigenschaften und das Verhalten der Systeme „Produktentwicklung“ und „Natur“ nicht nur deshalb unterschiedlich sind, weil sie aus verschiedenen Elementen bestehen. Die Unterschiede sind vor allem in der Verschiedenartigkeit ihrer inneren Struktur und der Organisation

[1] Zwar handelt der Produktentwickler auch opportun (vgl. [Cle06]) und erhöht dadurch die Handlungsvarietät und damit auch die Komplexität der Produktentwicklung, die meisten Vorgehensmodelle der Produktentwicklung bilden dies jedoch nicht ab.

ihrer Elemente begründet. So werden z. B. in der Natur komplexe Systeme nicht zentral gestaltet und gesteuert. Stattdessen erfolgt die Gestaltung und Steuerung aus ihnen selbst heraus. Selbstorganisation und Selbstregulierung sind universelle Architektur- und Funktionsgesetzmäßigkeiten in der Natur [Mal98, S. 8]. Diese Grundprinzipien können auch als Leitmotiv für die Produktentwicklung übernommen werden. Ziel muss es daher sein, die Strukturen und Prozesse der Produktentwicklung in die Lage zu versetzen, sich selbst zu verändern und zu optimieren. Dazu müssen die einzelnen Elemente des Systems mit erweiterten Eigenschaften und Fähigkeiten ausgestattet werden. Sie müssen eine größere Autonomie und die Möglichkeit erhalten, Veränderungen in ihrem Umfeld zu erkennen und darauf reagieren können.

Ansätze dafür gibt es bereits: Mit der Umstellung auf Projektorganisation haben sich die Strukturen in den Unternehmen verändert. Ein Teil der Steuerungs- und Kontrollaufgaben wurde in kleinere organisatorische Einheiten (Projektteams) verlagert. Deren Aufgabe ist es, ein Subsystem des zu steuernden Gesamtsystems so zu beeinflussen, dass es dem Ziel des übergeordneten Steuerungssystem (Unternehmen) entspricht. Die Projektteams erhalten hierzu neben der Zielvorgabe auch ein Kosten- und Zeitbudget, in dessen Rahmen sie frei agieren können. In der Regel definieren die Projektteams Teilziele und Aufgaben, die notwendig sind, um die Teilziele zu erreichen, und halten diese in einem Projektplan fest. Der Projektplan ist seinerseits ein Instrument für das Projektteam, um das Subsystem kontrollieren und steuern zu können. Er trägt zur systematischen Zielerreichung bei.

Die Auslagerung von Steuerungsaufgaben in Teilsysteme, wie es bei der Bildung von Projektteams erfolgt, genügt allein jedoch nicht, um die Komplexität des Gesamtsystems zu erhöhen. Es muss gleichzeitig sichergestellt werden, dass die Teilsysteme ihre Aufgaben im Sinn des Gesamtsystems erfüllen. In Unternehmen wird dies heute durch eine übergeordnete Instanz überwacht und gesteuert. In einem komplexen System gibt es jedoch keine solche zentrale Kontrollinstanz. Komplexe Systeme organisieren sich selbst. Das übergeordnete Verhalten des Gesamtsystems wird durch die Struktur seiner Systemelemente bestimmt. Einen stabilen Zustand erreicht das System nur dann, wenn seine Elemente – die wiederum selbst Systeme darstellen – durch *negative Rückkopplungen* untereinander verbunden sind. Bei dieser Art von Regelkreisen wirkt der Output des Systems direkt oder indirekt über andere Systeme seinem Input entgegen [Fle70, S. 42f]. Ein Mehr an Output wirkt sich dämpfend auf das System und die Produktion des Outputs aus, während ein Weniger an Output das System anregt, mehr Output zu produzieren. Dies führt zu einem Einschwingen auf einen Zielwert. Ohne die negativen Rückkopplungen stellt sich dieses Gleichgewicht nicht ein und das System zerfällt.

Wie oben dargestellt, ist die Produktentwicklung bislang als zentral gesteuerter Prozess konzipiert. In dieser Arbeit wird ein anderer Ansatz vorgeschlagen: Die Produktentwicklung ist ein komplexer Prozess. Zu seiner Gestaltung sind daher Mittel notwendig, die dieser Komplexität gerecht werden. Die Aufgabe der Produktentwicklungsmethodik ist daher nicht nur, dem Produktentwickler konkrete Handlungsanweisungen zum Entwickeln und Konstruieren technischer System im Sinne eines planbaren Vorgehens bereitzustellen. Zukünftig muss sie verstärkt die Aktivitäten, die der Produktentwickler im Rahmen seiner Handlungsspielräume tätigt, so kanalisieren, dass diese effizient zur

Zielerreichung beitragen. Hierzu müssen die oben beschriebenen negativen Rückkopplungen in den Prozess implementiert werden. Ein Produktentwickler, der eine Aufgabe zur eigenverantwortlichen Bearbeitung übertragen bekommt, muss in der Lage sein, kontinuierlich die Folgen des eigenen Tuns zu reflektieren und ggf. seine Aktivitäten anpassen können.

Ähnliche Ansätze werden auch von anderen Autoren diskutiert. So ist z. B. die von Ottosson [Ott04] vorgestellte *Dynamic Product Development (DPD)* eine auf den Anwender orientiert Philosophie, die Handlungsanweisungen in Form von Faustregeln formuliert. Sie schreibt dem Produktentwickler nicht vor, wann etwas zu tun ist und in welcher Reihenfolge, sondern vertraut darauf, dass dieser in einem selbstorganisierten Prozess die notwendigen Aktivitäten kontinuierlich und opportun den aktuellen Gegebenheiten anpasst [Hol07, S. 31]. Mit dem Produktmodell und dem Vorgehensmodell der AKT, die in den nachfolgenden Abschnitten 4.3 und 4.4 vorgestellt werden, will diese Arbeit einen Beitrag zur Gestaltung einer *selbstorganisierenden Produktentwicklung* leisten.

Auch wenn es, wie dargestellt, für die Produktentwicklung bereits Ansätze zum Umgang mit Komplexität gibt, stellt sich die Frage, wie die Aufbau- und Ablauforganisation in einem Unternehmen prinzipiell konzeptioniert werden müssen, damit diese zur Selbstorganisation fähig sind und angemessen auf komplexe Situationen reagieren können. Die Beantwortung dieser Frage stand nicht im Fokus dieser Arbeit. Eine Bearbeitung sollte jedoch im Rahmen künftiger Forschungsaktivitäten erfolgen.

4.3 Produktmodell

Im Folgeantrag wurden für das Produktmodell die folgenden Forschungsfragen formuliert, wobei anzumerken ist, dass die Fragen im Antrag noch den alten Sichtweisen-Begriff verwendeten. Gemäß den Ausführungen in Abschnitt 4.1 wurden die Fragen daher aktualisiert und der Begriff „Sichtweise“ durch „Attribut“ ersetzt:

- Was ist der Inhalt der Attribute?
- Welche Granularität sollen die Attribute aufweisen?
- Welche Attribute sollen zweckmäßig gewählt werden?
- Sind die Attribute statisch oder dynamisch?

Die Entwicklung eines Produktes ist ein multikriterielles Optimierungsproblem. Der Produktentwickler bildet Teilmodelle, um die Gesamtaufgabe in besser lösbare Teilprobleme zu gliedern. In einem Partialmodell werden die Produktparameter zusammengefasst, die für die Beschreibung und zur Lösung eines bestimmten Teilproblems relevant sind. Da die Partialmodelle, wie oben beschrieben, Sichten auf das Produkt darstellen, richtet sich der Inhalt einer Sichtweise nach den Aufgaben, die mit Hilfe des Partialmodells bearbeitet werden sollen.

In einem mechatronischen Produkt, in dessen Entwicklung mehrere Fachdisziplinen involviert sind, gibt es Partialmodelle, die fachdisziplinspezifisch und solche, die domänenübergreifend sind. Zu den fachdisziplinspezifischen Modellen gehören:

- Berechnungsmodelle für die Finite Element Methode (Maschinenbau)
- Modelle für den logischen Entwurf von Schaltkreisen (Elektrotechnik)
- Modelle zur Beschreibung der Softwarearchitektur (Informationsverarbeitung)

Beispiele für domänenübergreifende Partialmodelle sind:

- Modelle für die Simulation von Systemeigenschaften wie z. B. Eigenfrequenzen
- Modelle für die Gestaltung des Zusammenwirkens von Mensch und Maschine

Die Frage nach der Granularität der Sichtweisen erfolgt mit dem Ziel, eine Aussage darüber zu treffen, ob für das Produktmodell der AKT wenige, relativ grob gestaltete Sichtweisen verwendet werden, oder ob die Definition einer Vielzahl kleiner Sichtweisen die bessere Alternative darstellt. Partialmodelle werden, wie oben dargestellt, vom Produktentwickler gebildet, um die Gesamtaufgabe in besser lösbare Teilprobleme zu gliedern. Welche Sichtweisen für den Produktentwickler zweckmäßig sind und in welcher Granularität er sie wählt, richtet sich nach der Komplexität der zu lösenden Aufgaben und ist abhängig von den Methoden und Werkzeugen, die ihm hierfür zur Lösung zur Verfügung stehen.

Es ist zu erwarten, dass in ähnlichen Produktentwicklungsprojekten ähnliche Partialmodelle genutzt werden. Liegt zwei Produktentwicklungen der gleiche Produktentwicklungsprozess zugrunde, werden zu bestimmten Zeitpunkten des Prozesses die gleichen Teilaufgaben bearbeitet und damit die gleichen Partialmodelle genutzt. Da sich jedoch die Randbedingungen von zwei Entwicklungsprojekten nie völlig gleichen, können Partialmodelle, die in einem Projekt relevant sind, für ein anderes Entwicklungsprojekt ohne Bedeutung sein. Welche Partialmodelle wann und in welchem Umfang für die Bearbeitung des Projektes benötigt werden, lässt sich daher nur in Grenzen vorherbestimmen. Zum Teil ergibt sich die Notwendigkeit für ein bestimmtes Partialmodell erst während der Bearbeitung der Aufgabe.

Analog verhält es sich mit den Produktattributen. Sie leiten sich aus den Anforderungen ab, die an das Produkt gestellt werden, und beschreiben das Verhalten des Produktes. Da die Anforderungen erst während des Produktentwicklungsprozesses erarbeitet werden und sie sich zudem im Verlauf des Prozesses ändern können, sind auch die Attribute als veränderlich zu betrachten. Wie die Partialmodelle können sie auch in bestimmten Grenzen vorherbestimmt werden.

Wie oben dargestellt, können in einem Produktentwicklungsprojekt Partialmodelle und Attribute vorherbestimmt werden, wenn Erfahrungswerte für ähnliche Projekte vorliegen. In diesem Fall können sie als eine Art Leitbild verstanden werden. Sie werden vom Produktentwickler antizipiert, ohne dass die zugehörigen Teilprobleme bzw. Anforderungen formuliert wurden. Auf diese Weise geben sie der Produktentwicklung eine

Richtung vor und stellen sicher, dass die Entwicklung des Produktes in ganzheitlicher Art und Weise erfolgt.

Im Zwischenbericht des DFG-Vorhabens VA 134/10-1 [VK10] wurde ein Produktbeschreibungsmodell vorgestellt, bei dem die Designparameter in Form einer Matrix Merkmale und Eigenschaften (nach neuer Darstellung entspricht dies den Bewertungskriterien) mit Sichtweisen (entspricht jetzt den Attributen) verknüpften. Ziel der Darstellung ist es, die Elemente des Lösungsraumes zu strukturieren, um damit Übersichtlichkeit und Handhabbarkeit zu verbessern. Im Zuge der weiteren Bearbeitung des Forschungsthemas wurde die im Zwischenbericht gewählte Darstellung (vgl.[VK10, S. 20]) verändert und ergänzt. Bild 3 zeigt das Ergebnis.

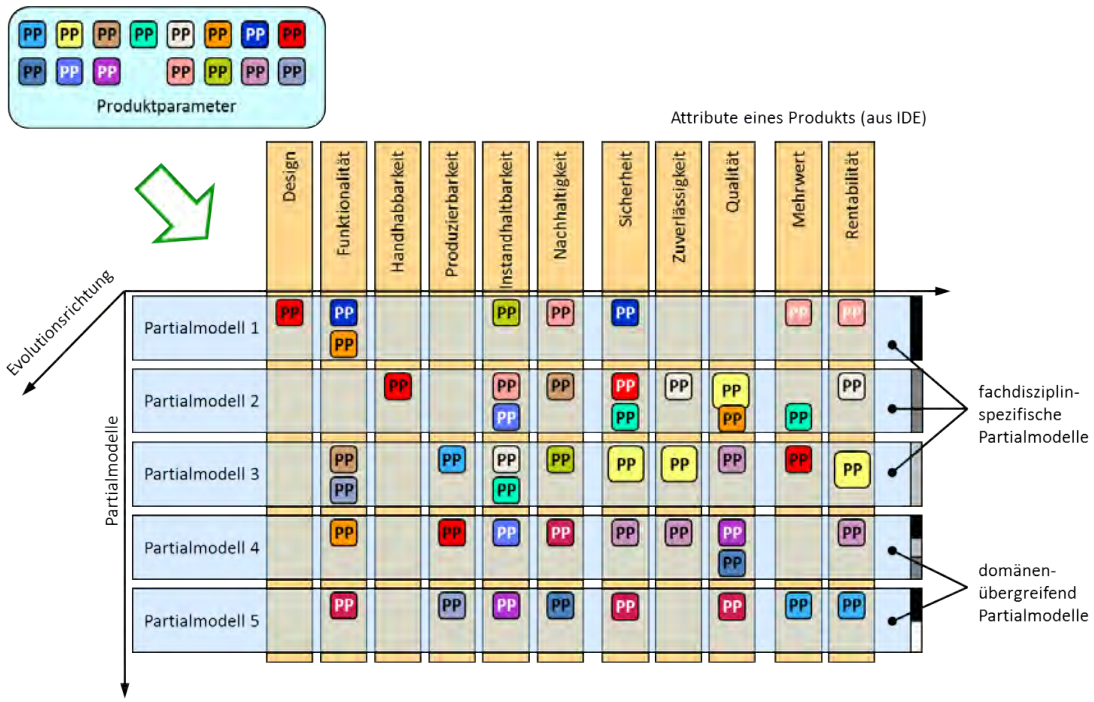

Bild 3: Produktbeschreibungsmodell

Zunächst wurde in der Matrix der Begriff „Designparameter“ durch den allgemeineren Begriff „Produktparameter“ ersetzt. Designparameter sind, wie oben dargestellt, Produktparameter, die direkt vom Produktentwickler verändert werden können. Produktparameter, die zur Bewertung eines Sachverhaltes genutzt werden, werden als Bewertungskriterien bezeichnet. Die Entscheidung, ob ein Produktparameter ein Designparameter oder ein Bewertungskriterium ist, hängt, wie beschrieben, von der Aufgabenstellung und damit vom Partialmodell ab. Da mit Hilfe der Matrix *alle* existierenden Parameter eines Produktes strukturiert werden, wird hier der Begriff „Produktparameter“ verwendet.

Die Eigenschaften und Merkmale des Produktes, die in der Darstellung des Zwischenberichtes in vertikaler Richtung aufgelistet waren, wurden im Bild 3 durch die Partial-

modelle ersetzt. Diese können, wie oben dargestellt, entweder einer bestimmten Fachdisziplin zugeordnet werden, oder sie sind domänenübergreifender Natur. Die einzelnen Fachdisziplinen wurden im Bild 3 am rechten Rand der Partialmodell-Blöcke mit unterschiedlichen Grautönen gekennzeichnet.

Im Zwischenbericht wurde vorgeschlagen, den Designparametern zusätzliche Informationen in Form von Etiketten (Tags) mitzugeben und auf diese Art und Weise innerhalb der Matrix zu ordnen. Tags verknüpfen Produktparameter mit den Partialmodellen und den Attributen des Produktes. Sie haben eine Schlagwortfunktion und können genutzt werden, um die Parameter zu gruppieren und zu filtern. Dieser Vorschlag aus dem Zwischenbericht wird prinzipiell beibehalten, nun jedoch auf die Produktparameter angewendet. So kann z. B. der gelb markierte und größenmäßig hervorgehobene Parameter im Bild 3 mit folgenden Etiketten versehen werden:

- „Partialmodell 2 – Qualität“
- „Partialmodell 3 – Sicherheit“
- „Partialmodell 3 – Zuverlässigkeit“
- „Partialmodell 3 – Rentabilität“

Die Etiketten zeigen, dass der genannte Produktparameter im Partialmodell 2 Auswirkungen auf die Qualität des Produktes hat, im Partialmodell 3 jedoch Sicherheit, Zuverlässigkeit und Rentabilität des Produktes beeinflusst.

Die Frage, wie die Tags gefunden und den Parameter zugeordnet werden, wurde im Zwischenbericht [VK10] nicht beantwortet. Dieser als *tagging* (Verschlagwortung) bezeichnete Prozess sollte im Hinblick auf die im Abschnitt 4.2 diskutierten Strategien zum Umgang mit der wachsenden Komplexität des Produktmodells

- automatisiert ablaufen,
- allgemeingültig sein und
- kein Produktwissen voraussetzen.

Da Produktparameter, Partialmodelle und Attribute eine Form von Produktwissen darstellen, dürfen sie für diesen Prozess nicht explizit vorgegeben werden. Vielmehr sollten sie als veränderliche Größen betrachtet werden und sich aus dem *tagging*-Prozess ergeben.

Softwaretechnisch könnte das *tagging* der Produktparameter mit Hilfe einer Clusteranalyse gelöst werden. Dabei werden die Eigenschaften von Daten einer großen Menge (im Fall der Produktentwicklung sind dies die Produktparameter des Produktmodells) statistisch auf Gemeinsamkeiten untersucht. Ziel der Analyse ist es, innerhalb der Datenmenge Gruppen zu identifizieren [BPW10]. Die Bezeichnungen dieser Gruppen können dann für die Verschlagwortung genutzt werden.

Um eine Clusteranalyse auf den Lösungsraum anzuwenden, muss der Cluster-Algorithmus Zugriff auf die Produktparameter haben und ihre Eigenschaften auswerten können.

Zu den Eigenschaften gehören dabei nicht nur Name und Wert des Parameters. Soll eine Clusteranalyse Zusammenhänge zwischen den Daten aufzeigen, müssen eine Vielzahl weiterer Eigenschaften erfasst werden. Ausgewertet werden könnte z. B. Folgendes:

- Wer hat den Produktparameter erzeugt?
- Wer nutzt den Produktparameter lesend?
- Wer ändert den Produktparameter?

Neben diesen Informationen sind auch die Zusammenhänge zwischen den einzelnen Produktparametern wichtig. Die Beziehungen zwischen ihnen lassen sich z. B. durch die Auswertung verschiedener Entwicklungsstände des Produktmodells sichtbar machen. Folgende Fragen sind dabei von Interesse:

- Wann wurde der Wert eines Produktparameters durch den Produktentwickler geändert?
- Wann wurde der Wert eines Produktparameters aufgrund der Wertänderung eines anderen Parameters verändert?
- Wie hat sich der Wert eines Produktparameters in der Vergangenheit verändert (Wertehistorie)?

Die Informationen, die durch die Beantwortung der Fragen gewonnen werden, geben einen Hinweis darauf, zu welchen Zeiten im Produktentwicklungsprozess welche Partialmodelle existieren und aus welchen Produktparametern sie sich zusammensetzen. Darüber hinaus zeigen sie, wann ein Produktparameter zu einem Designparameter wird und wann er als Bewertungskriterium fungiert. Mit Hilfe der Information kann zudem ein Partialmodell hinsichtlich seines fachspezifischen oder domänenübergreifenden Charakters gruppiert werden.

Im Bild 3 sind in horizontaler Richtung die Attribute des Produktes aufgeführt. Wie in Abschnitt 4.1 dargestellt, dienen sie dazu, das Produktverhalten zu beschreiben und zu bewerten. Die Produktparameter beeinflussen in Abhängigkeit vom Partialmodell das Verhalten des Produktes unterschiedlich. Soll mit Hilfe eines Cluster-Algorithmus der Zusammenhang zwischen den Produktparametern und dem Produktverhalten (und somit der Attribute des Produktes) ermittelt werden, müssen zum einen die Anforderungen an das Produkt in der Analyse berücksichtigt und zum anderen müssen die Gründe, die zur Veränderung von Designparameter führen, erfasst werden. Hierzu können folgende Fragen formuliert werden:

- Welche Anforderungen sind zum gegenwärtigen Zeitpunkt bekannt?
- Welche Anforderungen sind antizipiert?
- Wie haben sich die Anforderungen in der Vergangenheit verändert (Historie)?

- Wie gut werden die Anforderungen gegenwärtig erfüllt?
- Ist die Änderung eines Designparameters zufälliger Natur oder erfolgt sie begründet?

Die Antworten auf diese Fragen können als Eigenschaften der Produktparameter verstanden werden. Für die gefundenen Eigenschaften müssen numerische Distanz- und Ähnlichkeitsmaße definiert werden. Ein Cluster-Algorithmus analysiert diese Maße und bildet Gruppen, die in sich möglichst homogen, untereinander jedoch möglichst unterschiedlich sind.

Für Parameter, deren Werte bereits numerischer Natur sind (z. B. geometrische Informationen, Datumsangaben oder Versionsnummern), ist die Berechnung von Distanzmaßen problemlos möglich. Produktparameter können jedoch, wie oben dargestellt, eine Vielzahl weiterer Informationen aufnehmen. Für die Berechnung der Ähnlichkeit zwischen zwei Bildern, Texten oder anderer Informationen stehen spezielle Methoden und Werkzeuge des *data mining* zur Verfügung. Hierzu zählen z. B.:

Levenshtein-Distanz beschreibt die minimale Anzahl an Einfüge-, Lösch- und Änderungs-Operationen zwischen zwei Zeichenketten und wird vor allem zur Erkennung von Tippfehlern verwendet [Dam64].

Kohonennetze (Selbstorganisierende Karten) sind in der Lage, multidimensionale Datensätze in planare Strukturen zu überführen [Koh95].

Locality-sensitive hashing ist eine Methode zur Bildung von Hash-Werten, die zum Ziel hat, für ähnliche Datensätze ähnliche Hash-Werte zu generieren [GIM99].

Ein Verfahren der Clusterbildung stellt das *Single Linkage* dar. Bei diesem Verfahren werden die beiden Elemente, die das kleinste Distanzmaß aufweisen, zusammengefasst. Aus den Eigenschaftswerten der Elemente wird ein Eigenschaftswert der Gruppe gebildet, mit dem sich wiederum Distanzmaße zu anderen Gruppen berechnen lassen. Schrittweise werden nun alle Elemente zu Gruppen zusammengefasst. Das Ergebnis des Single-Linkage-Verfahrens ist eine hierarchische Clusterstruktur. Da sich die zu clusternden Daten während der Produktentwicklung dynamisch ändern, wird ein Cluster-Algorithmus nach jeder Datenänderung auch veränderte Strukturen finden [Bac+11].

Die gefundenen Cluster lassen sich oft nicht verbal beschreiben. Für das Produktmodell bedeutet dies, dass die Produktparameter zwar geclustert werden, die Bezeichnung der Cluster (die Schlagworte) jedoch nicht bekannt sind. In der Regel kann erst eine nachgeschaltete manuelle Analyse die Gemeinsamkeiten der Elemente eines Clusters aufzeigen. Basierend auf den identifizierten Gemeinsamkeiten kann dann eine Bezeichnung für den Tag gefunden werden. Eine Automatisierung dieses Prozesses ist wünschenswert. Allerdings ist gegenwärtig nicht bekannt, wie ein automatisches Finden und Zuweisen von Cluster-Bezeichnungen technisch realisiert werden kann. Die Lösung dieser Aufgabe sollte im Fokus einer künftigen Forschungsarbeit der Informatik stehen.

Die Beschreibung der Eigenschaften und ihrer Werte müssen, wie oben dargestellt wurde, für den Algorithmus zugänglich sein. Am einfachsten erscheint dies mit einem

Produktmodell möglich, bei dem die Daten zentral verwaltet werden. Tatsächlich wird diese Strategie von Systemanbietern, wie Siemens PLM Software, Dassault Systèmes oder Parametric Technology Corporation, derzeit verfolgt. Langfristig ist der Erfolg eines solchen Ansatzes jedoch unwahrscheinlich, weil ein derartiges monolithisches System zur Verwaltung der Produktdaten wenig flexibel und aufwändig zu warten ist. Erweiterungen oder individuelle Anpassungen können nur durch die Systemanbieter selbst oder durch Unternehmen erfolgen, die vom Systemanbieter autorisiert wurden. Eine kurzfristige Problemlösung ist nicht dabei zu erwarten.

Vor diesem Hintergrund und im Hinblick auf den Umgang mit Komplexität (siehe Abschnitt 4.2) wird vorgeschlagen, das Produktmodell nicht wie bisher in einem klassischen PDM-System zu verwalten, sondern diese Aufgabe in eine dezentrale Struktur zu verlagern. Eine vertiefende Diskussion dieses Ansatzes erfolgt in Abschnitt 4.5. An dieser Stelle wird er daher nur kurz skizziert, um darzustellen, wie ein Cluster-Algorithmus in einem solchen Szenario agieren könnte.

Die Struktur zur Verwaltung der Produktdaten muss flexibel, skalierbar und selbstorganisiert sein. Idealerweise wird sie durch die Anwendungsprogramme selbst gebildet, denn hier werden die Produktparameter erzeugt, genutzt und verändert. Die Anwendungsprogramme müssen daher in die Lage versetzt werden, „ihre“ Partialmodelle selbst zu verwalten und Informationen darüber untereinander auszutauschen. Hierzu muss eine Kommunikationsschnittstelle geschaffen werden, über die auch ein Cluster-Algorithmus die Anwendungsprogramme ansprechen kann, um die Produktparameter und ihre Eigenschaften abzufragen. Da die Informationen zum Produktmodell an vielen Stellen gespeichert sind, müssen die Anwendungsprogramme dem Cluster-Algorithmus auch mitteilen, wo weitere Fragmente des Produktmodells zu finden sind. Die Daten, die der Cluster-Algorithmus von den Anwendungsprogrammen erhält, können widersprüchlich sein. Kleine Inkonsistenzen im Datenbestand stellen bei der Auswertung in der Regel kein Problem dar. Dennoch ist festzustellen, dass Widersprüche in den Daten eine scharfe Abgrenzung der Cluster und damit eine Interpretation der Ergebnisse erschweren.

4.4 Vorgehensmodell

Für das Vorgehensmodell der AKT wurden im Antrag die nachfolgenden Forschungsfragen formuliert. Wie bei den Forschungsfragen zum Produktmodell im Abschnitt 4.3 wurde auch bei diesen Fragen der alte Sichtweisen-Begriff verwendet, der gemäß den Ausführungen in Abschnitt 4.1 an dieser Stelle durch den Begriff „Attribut“ ersetzt wurde:

- Kann es „Vorschläge“ zum zeitlichen Ablauf der Aktivitäten geben?
- Woher weiß der Produktentwickler, in welcher Reihenfolge er die Attribute bearbeiten soll?
- Wie kann die Konsistenz des Produktmodells über den gesamten Entwicklungsprozess hinweg sichergestellt werden?

In Abschnitt 4.1 wurde dargestellt, dass die Partialmodelle als Optimierungsobjekte verstanden werden können, für die der Produktentwickler eine Lösung finden muss. Das Vorgehensmodell der AKT enthält eine Beschreibung dieses Prozesses. Im Zwischenbericht [VK10] wurde hierfür – in Anlehnung an die biologische Evolution – ein Prozess aus Selektion, Duplikation, Rekombination sowie Mutation vorgeschlagen. Wegner [Weg99, S. 83] stellt heraus, dass eine Optimierung in der Produktentwicklung auf mehreren Ebenen erfolgt und stellt die Selbstähnlichkeit des Prozesses heraus. In jeder Entwicklungsphase und auf allen Detaillierungsstufen lassen sich die genannten evolutionären Operatoren identifizieren.

Die biologische Evolution verläuft ziellos. Sie besitzt weder Anfang noch Ende. Für den Gesamtablauf im Unternehmensbereich „Produktentwicklung“ gilt dies in gleichem Maße. Fortwährend werden hier neue Lösungen in Form von Produktmodellen generiert. Dabei fließen Produktmodelldaten vorangegangener Lösungen teilweise in die Entwicklung aktueller Lösungen ein. Verschiedene Entwicklungsstränge laufen parallel. Zusätzlich beeinflussen Impulse von außen (z. B. geänderte Randbedingungen, Anregungen und Ideen) den Entwicklungsprozess. Die Entwicklungsstränge können dadurch enden oder neu initiiert werden. Gleichzeitig sind die entwickelten Lösungen selbst Impulsgeber für Prozesse außerhalb des Unternehmens. Bild 4 stellt diesen Prozess schematisch dar.

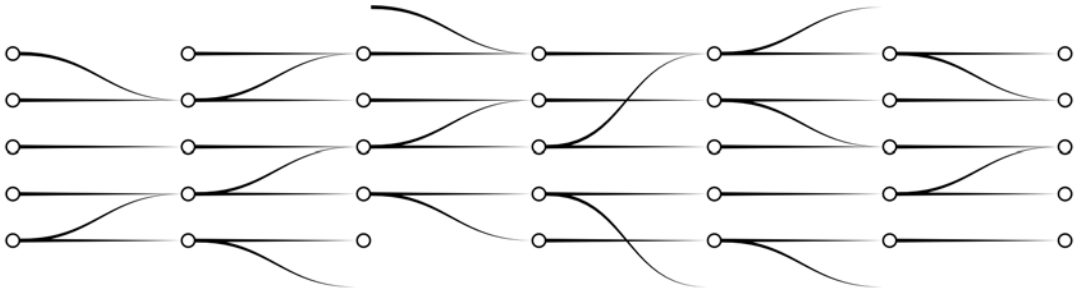

Bild 4: Entwicklungsstränge in der Produktentwicklung

Innerhalb des Gesamtablaufes im Unternehmensbereich „Produktentwicklung“ gibt es verschiedene Produktentwicklungsprojekte, deren Grundlage der Produktentwicklungsprozess ist. Der Produktentwicklungsprozess und die zu ihm gehörenden Projekte haben – im Gegensatz zum Gesamtablauf der Produktentwicklung und der biologischen Evolution – einen Anfangs- und einen Endzeitpunkt. Sie verlaufen zielorientiert (vgl. Tabelle 2). Am Ende eines Projektes existiert in der Regel *ein* Produktmodell, das im Rahmen der Fertigung in ein reales Produkt überführt wird. Da im Verlauf eines Produktentwicklungsprojektes eine Vielzahl von Lösungsideen in ein finale Produktbeschreibung münden, wird der Entwicklungsprozess von manchen Autoren (z. B. [WC92]) trichterförmig dargestellt. Bild 5 verdeutlicht, wie sich dieser „Entwicklungstrichter“ in den Gesamtablauf der Produktentwicklung eingliedert. Für die biologische Evolution ist eine trichterförmige Entwicklung hingegen untypisch.

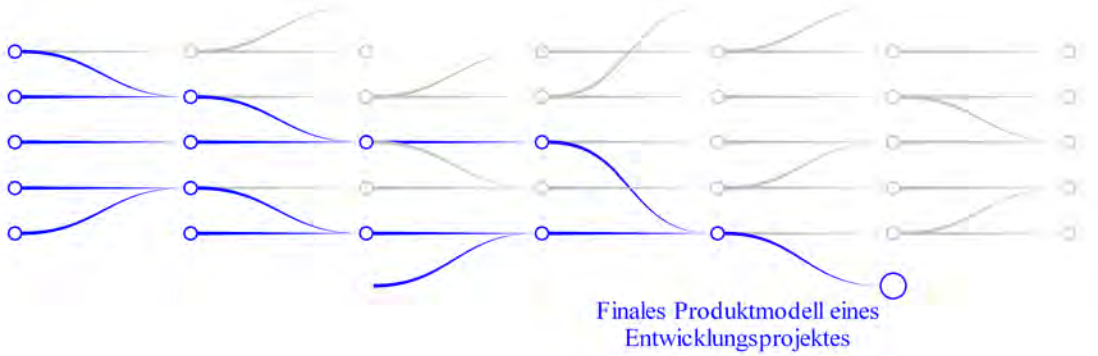

Bild 5: Schematische Darstellung eines Produktentwicklungsprojektes mit Entwicklungstrichter

Ein weiterer Unterschied zwischen der biologischen Evolution und der Produktentwicklung besteht in der Anzahl der generierten Lösungsalternativen und der Ausprägung der evolutionären Operatoren. Im Vergleich zur Natur, wo Lösungsalternativen in nahezu unbgrenzter Anzahl erzeugt werden, stehen dem Produktentwickler in einem Entwicklungsprojekt beschränkte Ressourcen zur Verfügung. Er kann daher nur einen sehr kleinen Teil der möglichen Alternativen detaillieren und bewerten. Die Bewertung der Lösungen erfolgt in der Produktentwicklung – im Gegensatz zur Natur, wo die Lösungsalternativen sich im „realen Umfeld“ bewähren müssen – bevor das Produkt auf dem Markt gebracht wird. Für die Bewertung werden Simulationen und Schätzungsverfahren genutzt. Die Entscheidung, welche Lösungen im Entwicklungsprozess weiterentwickelt und welche verworfen werden, trifft in der Regel der Produktentwickler. Die Selektion kann jedoch auch z. B. durch einen Evolutionären Algorithmus [Poh10] unterstützt werden. Hierzu ist es notwendig, den entsprechenden Teilprozess zu automatisieren. Diverse Arbeiten am LMI zeigen, wie dies innerhalb eines Entwicklungsprojektes umgesetzt werden kann (z. B. [Mac+99]; [Cle+04]; [CJC06a]; [CJC06b]; [KEV06]; [Mar+07]]).

Trotz der oben genannten Unterschiede lässt sich die Evolutions-Analogie auch auf Produktentwicklungsprojekte anwenden. Strategien, die in der biologischen Evolution erfolgreich sind, lassen sich auf den Produktentwicklungsprozess übertragen. Wie oben dargestellt, entscheidet am Ende eines Produktentwicklungsprojektes der Produktentwicklung darüber, welche Instanz des Produktmodells in ein reales Produkt überführt wird. Für eine bestmögliche Entscheidung sollte die Anzahl der Alternativen so groß wie möglich sein. In einem Produktentwicklungsprojekt begrenzen die zur Verfügung stehenden Ressourcen die Anzahl der Alternativen, die hier entstehen. Der Produktentwickler wählt daher nicht nur am Ende eines Projektes aus den verfügbaren Alternativen die Lösung aus, die den Anforderungen am Besten gerecht wird, sondern trifft bereits im Projektverlauf Entscheidungen, die die Anzahl der Alternativen soweit einschränken, dass sie mit den zur Verfügung stehenden Ressourcen weiter bearbeitet werden können. Die verbleidenden Lösungen sind (zum Zeitpunkt der Entscheidung) gleichwertig, aber nicht

gleichartig. Jede dieser Lösung entwickelt der Produktentwickler weiter und generiert Varianten, die dann wiederum bewertet werden können. Durch die Variantenbildung in jedem Entwicklungsschritt wird der Lösungsraum aufgeweitet, die Entscheidungen des Produktentwicklers hingegen verkleinern ihn. Man kann daher von einem „atmenden“ Lösungsraum sprechen.

Ohne die Einschränkung nach jedem Entwicklungsschritt würde sich die Anzahl der Alternativen, die der Produktentwickler bearbeiten muss, von Entwicklungsschritt zur Entwicklungsschritt vervielfachen. Dadurch, dass der Produktentwickler aus den generierten Alternativen eine bestimmte Anzahl gleichwertiger Lösungen auswählt, entsteht im Lösungsraum ein Korridor mit Lösungselementen, aus denen die Lösungsalternativen erzeugt werden. Die Größe des Korridorquerschnitts korreliert mit den zur Verfügung stehenden Ressourcen. Die Form des Korridors und seine Lage im Lösungsraum hängt hingegen wesentlich von den Entscheidungen des Produktentwicklers ab. Da sich sowohl die Ressourcen als auch die Randbedingungen im Verlauf eines Entwicklungsprojektes verändern, ist auch der Korridor veränderlich. Der Lösungsraum wird zudem, wie bereits im Zwischenbericht [VK10] dargestellt, durch die Naturgesetze und die Tabuzonen, die sich ebenfalls dynamisch ändern können, begrenzt. Eine Momentaufnahme des Lösungsraumes ist in Bild 6 schematisch dargestellt.

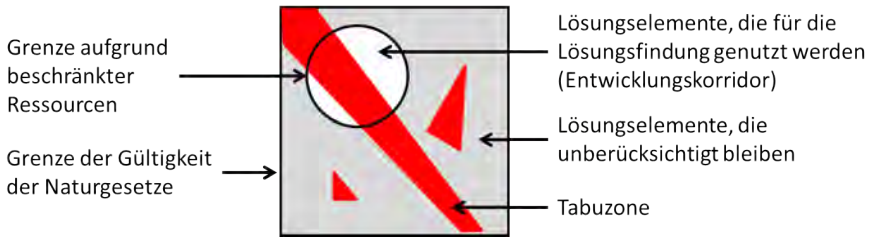

Bild 6: Schematische Darstellung des Lösungraumes der AKT

Im Idealfall befinden sich innerhalb des Entwicklungskorridors die Lösungen, die die bestmögliche Antwort auf die Entwicklungsaufgabe darstellen. Wenn sich jedoch die Anforderungen im Verlauf des Entwicklungsprojektes ändern, verändert sich auch die Bewertung der Lösungselemente. Es kann somit vorkommen, dass die bestmögliche Lösung nicht generiert werden kann, weil dessen Lösungselemente sich nicht im Entwicklungskorridor und somit außerhalb der Blickwinkels des Produktentwicklers befinden. Die AKT löst dieses Problem analog zur biologischen Evolution: Ein geringer Teil der Lösungsalternativen wird durch Mutation verändert und eröffnet der Population so neue Suchrichtungen. Die „mutierten“ Lösungen „splitten“ den Korridor und wirken auf diese Weise einem Determinismus entgegen (vgl. Bild 7). Die Mutationen verhindern, dass der Entwicklungsprozess als Ganzes auf eine Lösung hin konvergiert. Stattdessen wird kontinuierlich ein bestimmte Anzahl gleichwertiger Lösungen vorgehalten.

Zusammenfassend lässt sich feststellen, dass die AKT den Lösungsraum in der Form eines Verbotsraumes, der durch Tabuzonen definiert ist, beschreibt. Die Tabuzonen entstehen durch Invertierung der Anforderungen (externe Randbedingungen). Sind keine

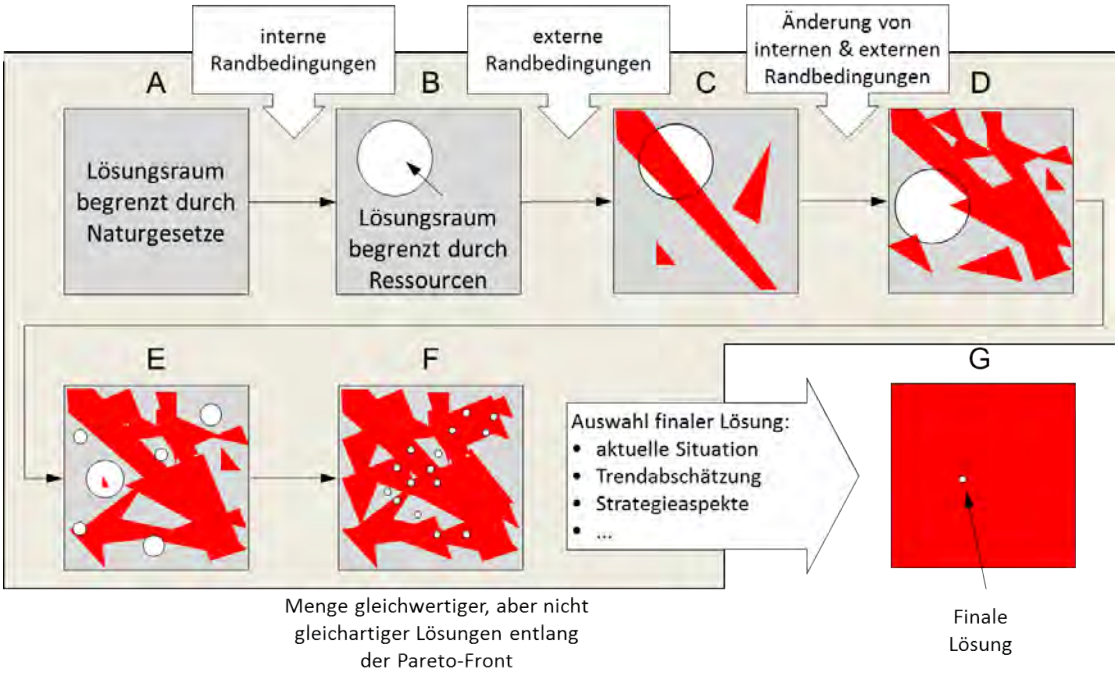

Bild 7: Entwicklung des Lösungsraumes

Anforderungen definiert, wird der Lösungsraum allein durch die Naturgesetze begrenzt (siehe Bild 7 Teil A). In der Regel sind zu Beginn einer Produktentwicklung nur wenige Anforderungen bekannt oder definiert. Der Lösungsraum ist folglich kaum durch Tabuzonen eingeschränkt, was bedeutet, dass eine Vielzahl von Lösungen denkbar sind. Aufgrund begrenzter Ressourcen (interne Randbedingungen) kann der Produktentwickler jedoch nur ein Teil dieser Lösungen berücksichtigen (Teil B im Bild 7). Die Aufgabe des Produktentwicklers besteht darin, unter Berücksichtigung sich ändernder Tabuzonen den Entwicklungskorridor so zu gestalten, dass am Ende der Produktentwicklung eine Vielzahl gleichwertiger, aber nicht gleichartiger Lösungen existiert (Bild 7 Teil F). Aus diesen Lösungen wählt der Produktentwickler die finale Lösung aus, die dann in ein reales Produkt überführt wird (Bild 7 Teil G). Die Grundlagen seiner Entscheidung bilden Strategieaspekte und die Abschätzung von Trends. In dieser letzten Phase eines Produktentwicklungsprojektes wird der Lösungsraum soweit konkretisiert und eingeschränkt, dass dieser (bis auf die finale Lösung) eine einzige Tabuzone bildet.

Wie oben dargestellt wurde, entstehen die Tabuzonen durch die Invertierung der Anforderungen, die sich wiederum aus den Bedingungen, die am Marktes vorgefunden werden oder die ein Kunde vorgibt, definieren. Sie werden als *externe Randbedingungen* bezeichnet. Auf der anderen Seite gibt es Randbedingungen, die das Produktmodell beeinflussen. Sie ergeben sich aus den zur Verfügung stehenden Ressourcen. Hierzu gehören auch innerbetriebliche Strukturen und Abläufe, verwendete Methoden und Werkzeuge. Diese Randbedingungen bilden die *inneren Randbedingungen*.

Im Verlauf des Entwicklungsprozesses trifft der Produktentwickler eine Vielzahl von Entscheidungen, die sowohl die Tabuzonen des Lösungsraumes als auch die Form und Lage des Entwicklungskorridors beeinflussen. Mit der Festlegung des Korridors beschreibt der Produktentwickler indirekt auch das Produktmodell, denn dieses setzt sich aus den Elementen des Lösungsraumes zusammen, die sich innerhalb des Entwicklungskorridors befinden.

Durch die Entscheidungen des Produktentwicklers bilden sich sowohl zwischen den Tabuzonen und dem Produktmodell als auch innerhalb der Tabuzonen und innerhalb des Produktmodells *Inkonsistenzen und Zwangsbedingungen*. Ziel seiner Bemühungen ist es, eine Überschneidung von Produktmodell und Tabuzonen zu vermeiden. Da jedoch sowohl das Produktmodell als auch die Tabuzonen veränderlich sind, kann die Lösungsfindung als Prozess verstanden werden, bei dem das Produktmodell und die Tabuzonen in einer konvergenten Entwicklung einander angenähert werden. Das Vorgehensmodell im Bild 8 spiegelt diesen Prozess wider.

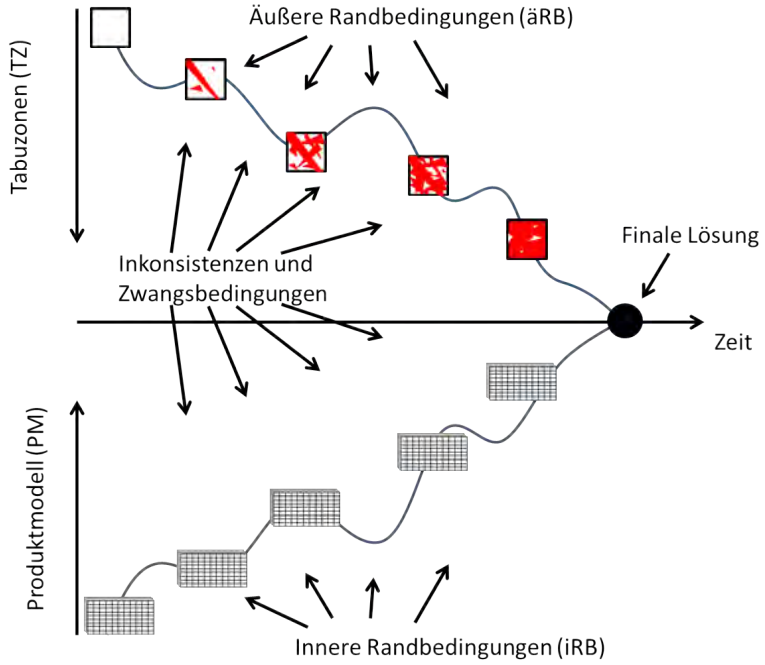

Bild 8: Konvergente Entwicklung von Tabuzonen und Produktmodell

Wie oben dargestellt, entstehen durch Mutation und Rekombination existierender Ansätze neue Lösungen. Aus diesem Lösungspool wird eine Anzahl gleichwertiger, aber nicht gleichartiger Lösungen selektiert, die vom Produktentwickler weiterverfolgt werden. Dieser Prozess geschieht im Kontext der Sichtweisen, d. h. für jedes Partialmodell werden auf diese Art und Weise Alternativen entwickelt. Im Bild 9 ist dieser Schritt als *Produktmodell weiterentwickeln (PMw)* dargestellt.

Die Partialmodelle eines Produktmodells sind heute noch weitgehend isoliert und werden unabhängig voneinander bearbeitet. Dabei entstehen zwischen ihnen, wie beschrieben, Inkonsistenzen und Zwangsbedingungen. In einem integrierten Produktmodell teilen sich die Partialmodelle eine gemeinsame Informationsbasis. Zwischen ihnen entsteht ein Netzwerk von Beziehungen. Änderungen an einem Modell wirken sich unmittelbar auf andere Partialmodelle aus. Die Widersprüche, die dabei zwischen den Partialmodellen entstehen, muss der Produktentwickler in einem nachfolgenden Schritt lösen. Das Bild 9 führt daher den Schritt *Produktmodell konsolidieren (PMk)* auf.

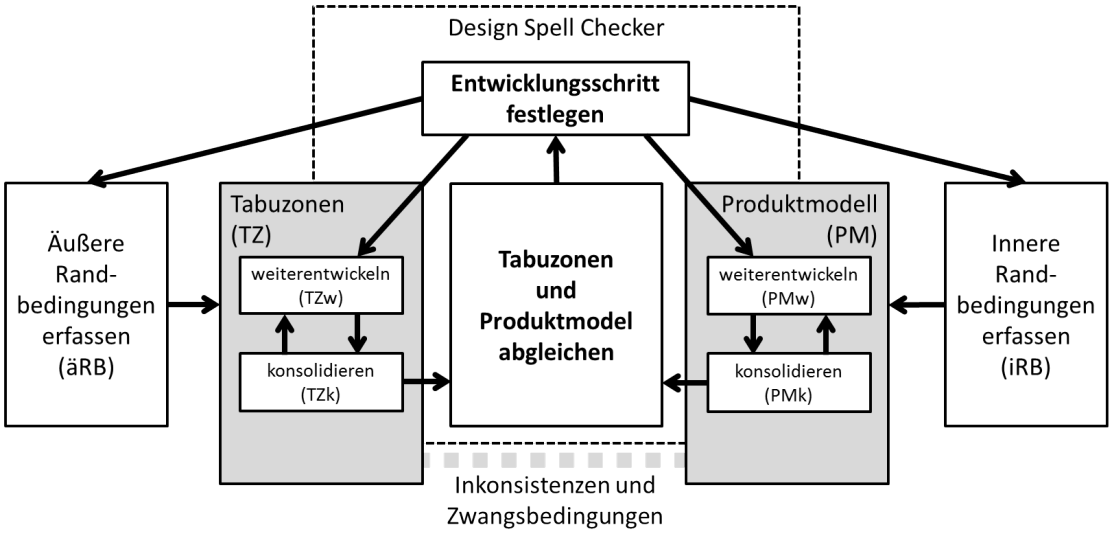

Bild 9: Vorgehensmodell der AKT

Analog zum Produktmodell wird auch die Entwicklung des Lösungsraumes mit seinen Tabuzonen vorangetrieben. Zunächst ist zu prüfen, ob sich die äußeren Randbedingungen verändert haben und zu geänderten Tabuzonen führen. Im Bild 9 entspricht dies dem Schritt *äußere Randbedingungen erfassen (äRB)*. Wie oben dargestellt wurde, werden die Tabuzonen jedoch nicht nur durch geänderte äußere Randbedingungen beeinflusst, sondern auch durch die Entscheidungen, die der Produktentwickler im Verlauf des Produktentwicklungsprozesses trifft. Mit jeder Partialmodell-Alternative ist zu prüfen, ob bislang unbekannte Tabuzonen verletzt worden sind. Im Bild 9 ist dies als *Tabuzonen weiterentwickeln (TZw)* bezeichnet. Zusätzlich ist zu prüfen, ob die bestehenden Tabuzonen-Definitionen in sich schlüssig sind. Inkonsistenzen entstehen hier durch Anforderungen, die sich widersprechen. Solche Widersprüche müssen in einem Konsolidierungsprozess aufgelöst werden. Im Vorgehensmodell geschieht dies im Schritt *Tabuzonen konsolidieren (TZk)*.

Der konsolidierte Lösungsraum und das konsolidierte Produktmodell werden einander vergleichend gegenübergestellt. In diesem Schritt wird geprüft, ob Produktmodell und Tabuzonen überschneidungsfrei und hinreichend genau beschrieben sind. Ist dies nicht der Fall, muss der nächste Entwicklungsschritt festgelegt werden. Diesen Prozess soll

ein sogenannter *Design Spell Checker* unterstützen, auf den im nachfolgenden Abschnitt eingegangen werden soll.

4.5 Design Spell Checker

Im Forschungsantrag wurde der *Design Spell Checker* als ein idealerweise automatisch arbeitendes rechnerunterstütztes Werkzeug dargestellt, mit dem der Produktentwickler den Produktreifegrad überprüfen und bewerten kann. Es dient der Qualitätssicherung des Produktmodells. Wie der Name bereits andeutet, entstand das Konzept des Design Spell Checker in Analogie zur automatisierten Rechtschreibprüfung, wie sie in Textverarbeitungsprogrammen (z. B. Microsoft Word) zu finden ist. Die Rechtschreibprüfung vergleicht hier im einfachsten Fall den geschriebenen Text mit eine Wortliste und markiert die gefundenen Fehler. Es findet ein Ist-Soll-Vergleich statt. In analoger Weise könnte ein Design Spell Checker den Ist-Zustand des Produktmodells mit einem Soll-Zustand vergleichen. Im Forschungsantrag wurde daher das Ziel formuliert, zu überprüfen, ob die Definition eines „Soll-Chromosoms“ möglich und sinnvoll ist.

Ein „Chromosom“ beschreibt, wie im Zwischenbericht dargestellt, eine Menge von Designparametern, die eine bestimmte Eigenschaft festlegen [VK10, S. 21]. Im Soll-Chromosom sind die Werte für die Designparameter zwar unbekannt, jedoch sind diese so gewählt, dass die resultierenden Bewertungskriterien die gewünschten Produkteigenschaften hervorbringen. Das Soll-Chromosom repräsentiert folglich ein ideales Produktmodell und dient dem Produktentwickler als Orientierungshilfe.

Wie im Abschnitt 4.4 dargestellt wurde, verläuft die biologische Evolution ziellos. Im Gegensatz dazu müssen Produktentwicklungsprojekte zu einem definierten Zeitpunkt abgeschlossen sein und ein definiertes Ziel erreicht haben. In der Regel ist dieses Ziel am Beginn einer Produktentwicklung noch nicht vollständig beschrieben, sondern wird erst im Verlauf des Prozesses mithilfe von Anforderungen konkretisiert. Das Ziel – und somit der Soll-Zustand, mit dem das Produktmodell verglichen werden kann – ändert sich dynamisch. Das Produktmodell besteht aus einer Vielzahl von Partialmodellen, die einander überlappen. Zudem entstehen die Partialmodelle zum Teil erst im Verlauf eines Produktentwicklungsprojektes und mit ihnen wird festgelegt, ob ein Produktparameter als Designparameter oder als Bewertungskriterium fungiert (vgl. Abschnitt 4.3). Ein Soll-Chromosom muss diese Randbedingungen berücksichtigen und der Komplexität des Produktentwicklungsprozesses (vgl. Abschnitt 4.2) Rechnung tragen.

Vor diesem Hintergrund wird vorgeschlagen, die Aufgabe einer begleitenden permanenten Qualitätssicherung, die bei jedem Arbeitsschritt des Produktentwicklers einen Abgleich mit den Anforderungen durchführt und bei Bedarf eine Korrektur der Vorgehensweise initiiert, nicht an zentraler Stelle (z. B. durch eine Softwareanwendung) zu implementieren, sondern die Funktionen an vielen Stellen bereitzustellen. Das Produktmodell soll hierzu dezentral verwaltet und die hierfür notwendigen Funktionen auf die Seite der Anwendungsprogramme verlagert werden (vgl. Abschnitt 4.4).

Aus dem eben Gesagten wird deutlich, dass für jedes Partialmodell ein Soll-Chromosom zu definieren ist. Das Soll-Chromosom antizipiert die Bewertungskriterien eines Partialmodells und beschreibt das Ziel der Entwicklung. Die Zielbeschreibung ist dyna-

misch und enthält oft Unschärfen, so dass sie sich in der Regel nicht als mathematische Zielfunktion formulieren lässt. Eine eindeutige Abbildung der Bewertungskriterien auf die Produkteigenschaften ist daher nicht möglich. Die Entscheidung, ob die gewählten Designparameter eines Partialmodells zu den gewünschten Produkteigenschaften führen, basiert daher nicht zuletzt auf dem Erfahrungswissen und der Intuition des Produktentwicklers.

Der Design Spell Checker soll den Produktentwickler bei seiner Entscheidungsfindung unterstützten. Darüber hinaus leistet er Hilfestellung bei folgenden Aufgaben (vgl. Bild 9):

- Tabuzonen weiterentwickeln
- Tabuzonen konsolidieren
- Produktmodell weiterentwickeln
- Produktmodell konsolidieren
- Tabuzone und Produktmodell vergleichen
- nächsten Entwicklungsschritt festlegen

Die genannten Aufgaben gehen über einen Ist-Soll-Vergleich im Sinne eines Word Spell Checkers (siehe oben) hinaus. Sie sind typisch für ein Assistenz- oder Navigationssystem. Das Konzept für ein solches *Assistenzsystem für die Produktentwicklung* wird nachfolgend skizziert. Da die konkrete Ausgestaltung der genannten Aufgaben vom Partialmodell und den Möglichkeiten, die das Anwendungsprogramm bietet, abhängig ist, wird an dieser Stelle nur beschrieben, wie die Interaktion zwischen den Anwendungsprogrammen gestaltet werden kann.

Wie oben dargestellt, sollen die Anwendungsprogramme ihre Produktmodelldaten selbst verwalten. Die Produktparameter, die innerhalb einer Anwendung genutzt werden, bilden ein Partialmodell. Überschneidungen zwischen den Partialmodellen entstehen dadurch, dass Anwendungsprogramme die gleichen Produktparameter verwenden. Inkonsistenzen zwischen den Partialmodellen sollen die Programme selbst lösen können. Sie bilden hierzu ein Netzwerk interagierender Knoten und müssen daher die Möglichkeit haben, miteinander zu kommunizieren.

Heutige Anwendungsprogramme sind in der Regel „von Hause aus“ nicht in der Lage, mit anderen Programmen zu interagieren, weil weder eine gemeinsame Sprache noch die benötigten Schnittstellen und Funktionen verfügbar sind. Diese müssen durch das Assistenzsystem bereitgestellt werden. Für die Realisierung eines solchen Systems sind verschiedene Lösungsansätze denkbar. Bild 10 zeigt drei mögliche Fälle.

Im Fall A greift das Assistenzsystem direkt auf die Modelldaten zu, die z. B. in Form einer Datei oder einer Datenbank vorliegen. Das Anwendungsprogramm selbst ist in den Vorgang nicht involviert. Es gibt keine Integration von Anwendungsprogramm und Assistenzsystem. Im Fall B erfolgt der Zugriff auf die Modelldaten über die Programmierschnittstelle (API) des Anwendungsprogramms. Ein Teil der Funktionalität wird von

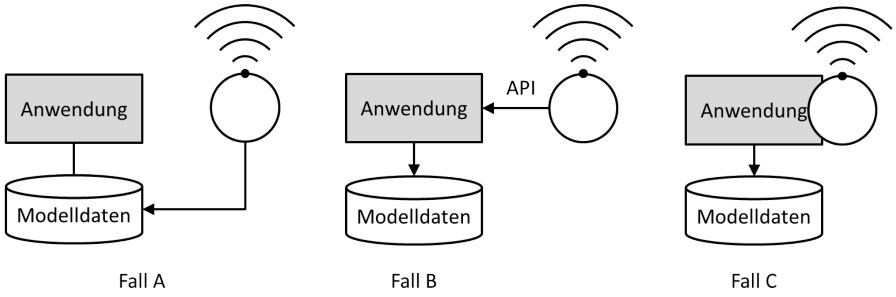

Bild 10: Möglichkeiten der Anwendungsintegration

der Anwendung übernommen, so dass hier von einer partiellen Integration gesprochen werden kann. Im Fall C wird das Anwendungsprogramm um die Funktionalität des Assistenzsystems erweitert. Das Assistenzsystem ist vollständig in das Anwendungsprogramm integriert. Fall C beschreibt somit einen Idealzustand: Jedes Anwendungsprogramm ist in der Lage, mit anderen Programmen zu kommunizieren und Informationen über Produktmodelldaten auszutauschen. Es besitzt die hierfür benötigten Funktionalitäten und greift direkt auf „seine“ Produktparameter zu.

Auch wenn eine Integration der Funktionalität des Assistenzsystems in die Anwendungsprogramme wünschenswert erscheint, so sind die Chancen, dass sich dieser Ansatz in der Praxis realisieren lässt, sehr gering. Bei den Anwendungsprogrammen, die in der Produktentwicklung eingesetzt werden, handelt es sich in der Regel um proprietäre (d. h. urheberrechtliche geschützte) Software. Änderungen am Programm kann nur der Hersteller vornehmen. Da die Entwicklungsstrategien der Hersteller von einer zentralisierten Produktdatenverwaltung ausgehen, ist von dieser Seite keine Unterstützung zu erwarten.

Ein Szenario ohne Integration der Anwendungsprogramme beschreibt der Fall A. Alle benötigten Funktionalitäten werden hier vom Assistenzsystem bereitgestellt. Wie bei der vollständigen Integration (Fall C) greift das System direkt auf die Modelldaten zu. Einschränkend ist dabei festzustellen, dass ein direkter Zugriff nur für solche Modelldaten möglich ist, deren Format bekannt ist. Für viele Dateiformate sind die Spezifikationen – d. h. die Art der Codierung und die Anordnung der Daten innerhalb der Datei – beschrieben. Bei Dateien, bei denen dies nicht der Fall ist und die als ASCII-Text gespeichert wurden, kann der Aufbau der Datei nachvollzogen werden. Viele Anbieter von Anwendungsprogrammen, die im Bereich der Produktentwicklung genutzt werden, verwenden jedoch native, binäre Formate, die nur von dem jeweiligen Anwendungsprogramm gelesen werden können. Um solche Dateiformate außerhalb ihrer Anwendung nutzen zu können, müssen in der Regel Lizenzgebühren entrichtet werden.

Ein weiterer Nachteil von Fall A ist, dass in dem Assistenzsystem viele Funktionen für den Datenzugriff implementiert werden müssten, die bereits vom Anwendungsprogramm bereitgestellt, jedoch vom Assistenzsystem nicht genutzt werden. Zudem sind Konflikte zu erwarten, wenn das Assistenzsystem und das Anwendungsprogramm gemeinsam auf die Modelldaten zugreifen. Einen Kompromiss stellt der Fall B dar. Der Zugriff auf die Modelldaten erfolgt hier über die API der Anwendung. Der Ansatz geht allerdings davon

aus, dass

- die Anwendung eine Programmierschnittstelle besitzt,
- die Modelldaten mithilfe der Programmierschnittstelle gelesen und verändert werden können,
- die Programmierschnittstelle das Erstellen von Komponenten der grafischen Benutzungsoberfläche (GUI) ermöglicht.

Die meisten in der Produktentwicklung genutzten Anwendungsprogramme erfüllen diese Voraussetzungen. Sie ermöglichen es, die Anwendung um die gewünschten Assistenzfunktionen zu erweitern. Als Nachteil ist jedoch zu nennen, dass für jede Anwendung, die in das Netzwerk eingebunden werden soll, Aufwand für die Implementierung entsteht. Aus entwicklungstechnischer Sicht ist es daher anzustreben, möglichst viele Funktionen zu generalisieren und anwendungsunabhängig zu implementieren. Die Funktion, die API eines Anwendungsprogramms anzusprechen, könnte in ein separates Modul ausgelagert werden. Für die Einbindung weiterer Anwendungen müsste in diesem Fall nur dieses eine Modul neu entwickelt.

Ungeachtet dessen sollten die Assistenzfunktionen nicht als eigenständiges Programm implementiert werden, sondern innerhalb der jeweiligen Anwendung zur Verfügung stehen. Dies ist wichtig, denn anderenfalls würde der Arbeitsfluss des Produktentwicklers unterbrochen werden, weil er seine gewohnte Arbeitsumgebung verlassen muss.

Die Kommunikation der Anwendungen im Netzwerk sollte in Form eines Hintergrundprozesses realisiert werden. Die Anwendungen müssen Informationen darüber austauschen:

- in welchen anderen Partialmodellen „ihre“ Produktparameter vorkommen,
- wann eine Änderung eines Produktparameters erfolgt,
- welche Tabuzonen ihnen bekannt sind,
- wann Tabuzonen geändert wurden,
- welche Regeln bei der Lösung von Inkonsistenzen gelten.

Für die Verwaltung dieser Informationen kann eine *verteilte Hashtabelle (DHT)* verwendet werden. Dabei handelt es sich um ein Protokollsystem, das dazu genutzt werden kann, den Speicherort von Informationen im Netzwerk zu speichern. Dezentralisierung und Effizienz stehen bei der Datenspeicherung im Vordergrund. Es existieren eine Reihe von Implementierungen (z. B. Chord[2], Gnutella[3], Kademlia[4]), auf die ein Assistenzsystem zurückgreifen könnte. Die Implementierungen werden oft von Programmen verwendet, die einen Dateiaustausch in einem Peer-to-Peer-Netzwerk ermöglichen. Die Nutzung

[2] `http://pdos.csail.mit.edu/chord`
[3] `http://rfc-gnutella.sf.net`
[4] `http://kademlia.scs.cs.nyu.edu`

von verteilten Hashtabellen ist jedoch nicht auf solche Anwendungen beschränkt. Sie werden z. B. auch in folgenden Softwareprodukten genutzt:

- YaCy – eine dezentrale Suchemaschine[5]
- Bitcoin – eine virtuelle Währung, die dezentral durch ein Computernetz geschöpft und verwaltet wird[6]
- Bitmessage – ein Protokoll für den vertraulichen Austausch von E-Mail-ähnlichen Botschaften in einem Peer-to-Peer-Netzwerk[7]

Die Anwendungsbeispiele zeigen, dass verteilte Hashtabellen geeignet sind, vielfältige Informationen dezentral zu verwalten. Die Protokolle und die genannten Anwendungen stehen unter einer quelloffenen Lizenz, d. h. ihre Quelltexte können auch von anderen Projekten genutzt und weiterentwickelt werden. Es ist zu überlegen, auch das Assistenzsystem als quelloffene Anwendung zu lizenzieren und als Open-Source-Programm zu veröffentlichen. Durch diesen Schritt könnte zum einen der Aufwand für die Entwicklung geteilt werden und zum anderen wäre es möglich, frühzeitig Feedback von potentiellen Anwendern zu erhalten.

5 Zusammenfassung und Ausblick

Im Rahmen des DFG-Vorhabens VA 134/10-2 wurde die Autogenetische Kontruktionstheorie (AKT) in den Bereichen Produktmodell, Verfahrensmodell und Design Spell Checker weiterentwickelt. Im Fokus stand dabei die Entwicklung mechatronischer Produkte. Diese sind – wie andere multidisziplinär entwickelte Produkte – durch eine deutlich größere Menge produktbeschreibender Informationen charakterisiert. Als Partner konnte das Institut für Rechnergestützte Methoden im Maschinenbau an der Johannes-Kepler-Universität Linz (Österreich) gewonnen werden, wo man sich schwerpunktmäßig mit der Modellierung mechatronischer Produkte beschäftigen. Die Ergebnisse des Projektes sind das Resultat eines intensiven Austausches mit den Linzer Kollegen.

Nach einer kurzen Einleitung erfolgte im Kapitel 2 die Darstellung der Problemstellung und der Motivation für das Projekt. Im Anschluss daran wurden im Kapitel 3 die Forschungsziele benannt und die Forschungsmethodik erläutert. Dabei wurde herausgestellt, dass die entdeckte Analogie zwischen Produktwicklung und biologischer Evolution den Ausgangspunkt für die AKT bildet. Für ihre Weiterentwicklung wurde daher die Analogiebildung systematisch betrieben. Die Schritte zum Aufbau eines ganzheitlichen Analogiebildes, das auch die Grenzen der Analogie aufzeigt, wurden erläutert. Gemeinsamkeiten und Unterschiede zwischen Produktentwicklung und Natur wurden für Produkte und Abläufe zusammengetragen und zeigten Ansatzpunkte für die Weiterentwicklung der AKT.

[5] `http://yacy.net`
[6] `https://bitcoin.org`
[7] `https://bitmessage.org`

Ein wesentlicher Unterschied zwischen Produktentwicklung und Natur besteht im Umgang mit Komplexität. Zusätzlich zum geplanten Forschungsprogramm wurde daher im Abschnitt 4.2 die Bedeutung von Komplexität für die Produktentwicklung beleuchtet und mögliche Strategien aufgezeigt. Darüber hinaus wurde im Zuge der Forschungsaktivitäten deutlich, dass die Begriffe „Sichtweise“, „Aspekt“ und „Attribut“ bislang nicht einheitlich verwendet wurden. Im Abschnitt 4.1 wurden die Begriffe daher voneinander abgrenzt.

Im Abschnitt 4.3 wurden die Ergebnisse zum Produktmodell dargestellt. Es wurde vorgeschlagen, das Produktmodell dezentral in Form vom Partialmodellen zu verwalten. Die Partialmodelle entstehen aus dem Produktentwicklungsprozess heraus und können mithilfe einer Cluster-Analyse sichtbar gemacht werden.

Die Forschungsergebnisse zum Thema „Vorgehensmodell“ wurden im Abschnitt 4.4 beschrieben. Es wurde herausgestellt, dass das Vorgehensmodell das opportune Handeln des Produktentwicklers unterstützen muss. Aus diesem Grunde wurde das Vorgehensmodell als ein Rahmenwerk beschrieben, das aus 8 Schritten besteht, zwischen denen der Produktentwickler in Abhängigkeit von der aktuellen Situation und dem Stand der Produktentwicklung wechselt.

Der Design Spell Checker war der dritte Themenbereich, der im Rahmen DFG-Vorhabens bearbeitet wurde. Die Ergebnisse sind in Abschnitt 4.5 dargestellt. Das Konzept sieht vor, die Anwendungen, die im Rahmen der Produktentwicklung genutzt werden, über ihre API zu erweitern, so dass diese untereinander interagieren und Informationen zu Produktparametern austauschen können.

Die im Antrag formulierten Forschungsziele wurden im Wesentlichen erreicht. Einschränkend ist festzustellen, dass der Design Spell Checker nicht wie geplant prototypisch umgesetzt werden konnte. Die Gründe hierfür sind zum einen, dass der tatsächliche Bearbeitungsaufwand höher war als ursprünglich geschätzt, zum anderen ergaben sich durch den Wechsel des Bearbeiters (Konstantin Kittel übergab Bearbeitung des Forschungsvorhabens an Dr.-Ing. André Jordan) Verzögerungen infolge der Einarbeitung.

Die im DFG-Vorhaben bearbeiteten Themen werden auch weiterhin Gegenstand der Forschung am Lehrstuhl für Maschinenbauinformatik sein. So existiert bereits ein von der Austrian Center of Competence in Mechatronics GmbH (ACCM) gefördertes Vorhaben, das zum Ziel hat, das Konzept des Design Spell Checkers weiter auszuarbeiten und prototypisch umzusetzen. In diesem Vorhaben wird die Zusammenarbeit dem Lehrstuhl für Maschinenbauinformatik der Otto-von-Guericke-Universität Magdeburg und dem Institut für Rechnergestützte Methoden im Maschinenbau an der Johannes-Kepler-Universität Linz fortgesetzt.

Literatur

[Ash56] W. R. Ashby. *An introduction to Cybernetics*. New York: Wiley, 1956.

[Bac+11] K. Backhaus u. a., Hrsg. *Multivariate Analysemethoden : eine anwendungsorientierte Einführung*. Springer-Lehrbuch. Berlin [u.a.]: Springer, 2011.

[BPW10] J. Bacher, A. Pöge und K. Wenzig. *Clusteranalyse. anwendungsorientierte Einführung in Klassifikationsverfahren*. 3., erg., vollst. überarb. und neu gestaltete Aufl. München: Oldenbourg, 2010. 538 S. ISBN: 9783486584578.

[BV94] T. Bercsey und S. Vajna. „Ein autogenetischer Ansatz für die Konstruktionstheorie“. In: *CAD-CAM-Report* 13 (1994), S. 98–105.

[Cil98] P. Cilliers. *Complexity and postmodernism. understanding complex systems*. London [u.a.]: Routledge, 1998. X, 156. ISBN: 0415152879.

[CJC06a] S. Clement, A. Jordan und F. Clement. „Evolutionsbasierende Produktentwicklung im Walzwerksbau“. In: *CAD-CAM-Report : engineering* 25.10 (2006), S. 64–67.

[CJC06b] S. Clement, A. Jordan und F. Clement. „Evolutionsbasierende Produktentwicklung im Walzwerksbaum, Teil 2“. In: *CAD-CAM-Report : engineering* 25.12 (2006), S. 32–37.

[Cle+04] S. Clement u. a. „Prototypeinsatz evolutionärer Algorithmen in der Motorenentwicklung bei Volkswagen“. In: *MTZ — Motortechnische Zeitschrift* 2004-03 (März 2004), S. 220–226.

[Cle06] S. Clement. „Erweiterung und Verifikation der Autogenetischen Konstruktionstheorie mit Hilfe einer evolutionsbasierten und systematisch-opportunistischen Vorgehensweise“. Diss. Otto-von-Guericke-Universität Magdeburg, 2006.

[Dam64] F. J. Damerau. „A technique for computer detection and correction of spelling errors“. In: *Communications of the ACM* 3 (März 1964), S. 171–176. ISSN: 0001-0782.

[Fle70] H.-J. Flechtner. *Grundbegriffe der Kybernetik. Eine Einführung*. 1. Aufl. Stuttgart: Wissenschaftliche Verlagsgesellschaft mbH, 1970.

[GIM99] A. Gionis, P. Indyk und R. Motwani. „Similarity Search in High Dimensions via Hashing“. In: *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*. Hrsg. von M. P. Atkinson u. a. Edinburgh, Scotland, UK: Morgan Kaufmann, 1999, S. 518–529. ISBN: 1-55860-615-7. URL: `http://www.vldb.org/conf/1999/P49.pdf`.

[Hol07] L. Holmdahl. „Complexity aspects of product development“. Diss. Magdeburg: Otto-von-Guericke-Universität Magdeburg, 2007. 4, 214. ISBN: 9783980768894.

[Hol73] H. Holz. „Analogie“. In: *Handbuch philosophischer Grundbegriffe*. München: Kösel, 1973, S. 51–64.

[Jor08] A. Jordan. „Methoden und Werkzeuge zur Unterstützung des Wissenstransfers in der Bionik“. Diss. Otto-von-Guericke-Universität Magdeburg, 2008.

[KBW73] H. Krings, H. M. Baumgartner und C. Wild, Hrsg. *Handbuch philosophischer Grundbegriffe. Studienausgabe; Band 1-6.* München: Kösel, 1973. 1874 S. ISBN: 3466400554.

[KEV06] K. Kittel, J. Edelmann-Nusser und S. Vajna. „Optimierung von Bogenmittelteilen aus verschiedenen Metalllegierungen mit Evolutionären Algorithmen“. In: *Sport und Informatik* IX (2006), S. 263–268.

[Koh95] T. Kohonen. *Self-organizing maps.* Springer series in information sciences 30. Berlin [u.a.]: Springer, 1995. XV, 362. ISBN: 3540586008.

[KT02] U. Küppers und H. Tributsch. *Verpacktes Leben – Verpackte Technik. Bionik der Verpackung.* Weinheim: Wiley-VCH, 2002. ISBN: 3527304436.

[Kun98] P. Kunzmann. *Dimensionen von Analogie. Wittgensteins Neuentdeckung eines klassischen Prinzips.* 1. Aufl. Düsseldorf [u.a.]: Parerga, 1998. 245 S. ISBN: 393045033X.

[Mac+99] P. Mack u. a. „Computer Aided Optimisation of Cylinder Head Inlet Ports Using Genetic Algorithms“. In: *Proceedings of International Conference on Engineering Design ICED.* Hrsg. von U. Lindemann u. a. München, 1999, 1203—1206.

[Mal98] F. Malik. *Komplexität - was ist das?* 1998. URL: `http://www.kybernetik.ch/dwn/Komplexitaet.pdf`.

[Mar+07] L. Marks u. a. „Optimierung und Berechnung von Losflanschverbindungen aus GFK“. In: *Chemie Ingenieur Technik* 79.9 (2007), S. 1424–1424. ISSN: 1522-2640. DOI: `10.1002/cite.200750034`. URL: `http://dx.doi.org/10.1002/cite.200750034`.

[McK04] B. McKelvey. „Toward a complexity science of entrepreneurship“. In: *Journal of Business Venturing* 19.3 (Mai 2004), S. 313–341. DOI: `10.1016/S0883-9026(03)00034-X`. URL: `http://dx.doi.org/10.1016/S0883-9026(03)00034-X`.

[MK05] H. Meerkamm und M. Koch. „Design for X“. In: J. Clarkson und C. Eckert. *Design process improvement. A review of current practice.* Cambridge: Springer, 2005, S. 306–323. ISBN: 978-1-85233-701-8.

[MR98] V. Mosbrugger und A. Roth. „Prinzipien der Evolution natürlicher Konstruktionen. Pflanzliche Wasserstoffsysteme als Modelle für anthropogene Transportsysteme“. In: *Bionik. Ökologische Technik nach dem Vorbild der Natur?* Hrsg. von A. v. Gleich und R. Bannasch. Stuttgart: Teubner, 1998, S. 91–108. ISBN: 3519061953.

[Nac98] W. Nachtigall. „Zehn Grundprinzipien natürlicher Konstruktionen. Zehn Gebote bionischen Designs“. In: *BIONA-report*. Hrsg. von A. Wisser und W. Nachtigall. BIONA-Report 12. Stuttgart: G. Fischer, 1998, S. 295–306. ISBN: 1560813105.

[Ott04] S. Ottosson. „Dynamic product development – DPD“. In: *Technovation* 24 (3 2004), S. 207–217.

[Poh10] H. Pohlheim. *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis*. Springer, 2010. ISBN: 3540664130.

[RM81] G. A. Rodan und T. J. Martin. „Role of osteoblasts in hormonal control of bone resorption–a hypothesis.“ eng. In: *Calcif Tissue Int* 33.4 (1981), S. 349–351.

[Sta73] H. Stachowiak. *Allgemeine Modelltheorie*. Wien [u.a.]: Springer, 1973. XV, 494. ISBN: 0387811060.

[Tie93] A. Tiemann. *Analogie. Analyse einer grundlegenden Denkweise in der Physik*. Reihe Physik 11. Frankfurt am Main: Harry Deutsch, 1993. 161 S. ISBN: 3817112947.

[VK10] S. Vajna und K. Kittel. *Zwischenbericht zum Forschungsvorhaben „Weiterentwicklung der Autogenetischen Konstruktionstheorie“ (VA 134/10-1)*. Zwischenbericht. 2010.

[VW97] S. Vajna und B. Wegner. „Features – Information Carriers for the Product Creation Process“. In: *Proceedings of the ASME Design Engineering Technical Conferences*. Vortrag DETC97/DAC-3737. Sacramento: Fraunhofer-IRB-Verlag, 1997.

[WC92] S. C. Wheelwright und K. B. Clark. *Revolutionizing product development. quantum leaps in speed, efficiency, and quality*. 7. pr. New York [u.a.]: Free Press, 1992. XIV, 364. ISBN: 0029055156.

[Web05] C. Weber. „CPM/PDD – An Extended Theoretical Approach to Modelling Products and Product Development Processes“. In: *Proceedings of the 2nd German-Israeli Symposium*. Hrsg. von H. Bley u. a. Stuttgart: Fraunhofer-IRB-Verlag, 2005, S. 159–179.

[Weg99] B. Wegner. „Autogenetische Konstruktionstheorie. Ein Beitrag für eine erweiterte Konstruktionstheorie auf der Basis Evolutionärer Algorithmen“. Diss. Magdeburg: Otto-von-Guericke-Universität Magdeburg, 1999. X, 149.

[You93] W. J. Youmans, Hrsg. *The Popular science monthly*. Bd. XLII. New York: D. Appleton und Company, 1893. URL: `openlibrary.org/books/OL13495494M`.

Publikationen

Im Rahmen des Forschungsvorhabens entstanden folgende Publikationen:

[Heh+11] P. Hehenberger u. a. „Autogenetic Design and Optimization using reduced System Models“. In: *56th International Scientific Colloquium Ilmenau University of Technology*. Sep. 2011.

[Kit+11] K. Kittel u. a. „Product Model of the Autogenetic Design Theory“. In: *Proceedings of the 18th International Conference on Engineering Design (ICED11)*. Hrsg. von S. J. Culley u. a. Bd. 4. Kopenhagen, 2011, S. 309–318.

[Kit+12] K. Kittel u. a. „Modeling and optimization of mechatronic systems using the autogenetic design theory“. In: *Eurocast 2011*. Lecture Notes in Computer Science 6928 (2012). Hrsg. von R. Moreno-Díaz, F. Pichler und A. Quesada-Arencibia, S. 97–104. ISSN: 978-3-642-27578-4.