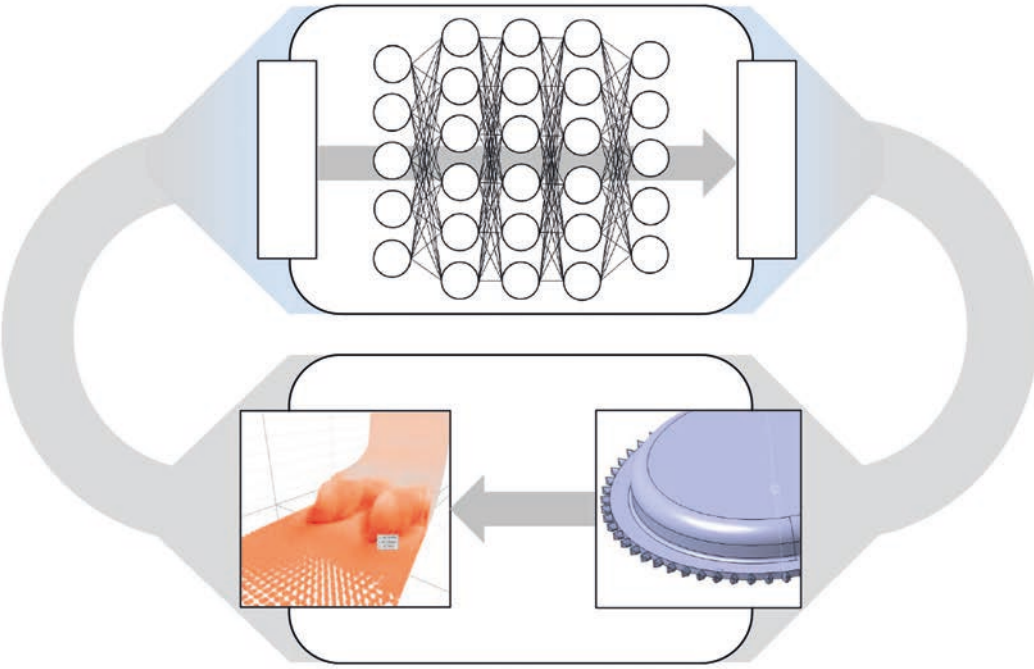


FAU Studien aus dem Maschinenbau 441

Fabian Dworschak

Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion

FAU University Press

Fabian Dworschak

Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion

FAU Studien aus dem Maschinenbau

Band 441

Fabian Dworschak

Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion

Dissertation aus dem Lehrstuhl für Konstruktionstechnik (KTmfk)
Prof. Dr.-Ing. Sandro Wartzack

Erlangen
FAU University Press
2024

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Kontakt: Fabian Dworschak, Friedrich-Alexander-Universität Erlangen-Nürnberg, (ROR https://ror.org/00f7hpc57), https://orcid.org/0000-0001-6987-4927

Bitte zitieren als
Dworschak, Fabian. 2024. *Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion.* FAU Studien aus dem Maschinenbau Band 441. Erlangen: FAU University Press. DOI: 10.25593/978-3-96147-740-1

Der vollständige Inhalt des Buchs ist als PDF über OPEN FAU
der Friedrich-Alexander-Universität Erlangen-Nürnberg abrufbar:
https://open.fau.de/home

Verlag und Auslieferung:
FAU University Press, Universitätsstraße 4, 91054 Erlangen

Druck: docupoint GmbH

ISBN: 978-3-96147-739-5 (Druckausgabe)
eISBN: 978-3-96147-740-1 (Online-Ausgabe)
ISSN: 2625-9974
DOI: 10.25593/978-3-96147-740-1

Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur
Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Fabian Dworschak, M.Sc.

aus Nürnberg

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen
Prüfung: 24.11.2023

Gutachter: Prof. Dr.-Ing. Sandro Wartzack
Prof. Dr.-Ing. Detlef Gerhard,
Ruhr Universität Bochung

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Konstruktionstechnik der Friedrich-Alexander-Universität Erlangen-Nürnberg. Ich möchte mich an dieser Stelle bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr.-Ing. Sandro Wartzack für die Betreuung dieser Arbeit. Sein in mich gesetztes Vertrauen ermöglichte es mir, mein Forschungsthema in umfassender Eigenverantwortung voranzutreiben und persönlich daran zu wachsen.

Bei Prof. Dr.-Ing. Detlef Gerhard möchte ich mich für den konstruktiven Austausch und sein Interesse an meiner Arbeit sowie die Übernahme des Zweitgutachtens bedanken. Ebenso gilt mein Dank Prof. Dr. Michael Kohlhase für die Übernahme des fachfremden Gutachtens und Prof. Dr.-Ing. Tino Hausotte für die Übernahme des Prüfungsvorsitzes.

Der Europäischen Union möchte ich für die Förderung meiner Arbeit meinen Dank aussprechen. Erkenntnisse aus dem Technologie Transferprojekt „Optimierungsbasierte Entwurfsmethodik in der frühen Phase der mechatronischen Produktentwicklung“, welches im Rahmen des Europäischen Fonds für regionale Entwicklung finanziert wurde, dienten als Grundlage für die vorliegende Arbeit.

Allen Kolleginnen und Kollegen des Lehrstuhls für Konstruktionstechnik möchte ich für die wertvolle und schöne Zeit danken. Ein besonderer Dank gilt meinem fachlichen Vorgänger Dr.-Ing. Christof Küstner. Durch seine Betreuung meiner studentischen Arbeiten hat er mir den Weg zum Lehrstuhl und damit in die Wissenschaft ermöglicht. Er stand und steht mir stet mit Rat und Tat zur Seite. Den weiteren Mitgliedern des A-Teams, Christopher Sauer und Thilo Breitsprecher, danke ich für die stets humorvolle und wissenschaftliche Zusammenarbeit. Der AEG-Crew danke ich für die unvergessliche Zeit am herausragendsten Standort des Lehrstuhls. Besonders die vielen, kreativen Impression, die es mir ermöglicht haben, stets mit neuem und klarem Fokus auf die Forschung zu arbeiten und mich jeden Tag meinen wissenschaftlichen Aufgaben zuzuwenden. Vielen Dank für all die wöchentlichen Routinen, die einen strukturierten Alltag ermöglicht haben.

Auch den technischen und administrativen Kolleginnen und Kollegen des Lehrstuhls für Konstruktionstechnik sei gedankt. Ihre Unterstützung in IT und Verwaltung ermöglichte den Fokus auf die wissenschaftlichen Themen.

An dieser Stelle möchte im mich bei meinen Eltern bedanken, die mir stets den Rücken frei gehalten haben und mir diesen Weg ermöglicht haben. Ohne eure Basis und euren Rückhalt wäre ich diesen Weg nicht gegangen. Dafür möchte ich euch von ganzem Herzen Danke sagen. Meinem älteren Sohn Carl danke ich für die Lektionen, die er mich bereits gelehrt hat und für alle, die er mir noch beibringen wird. Meinem jüngeren Sohn Toni danke ich für sein Lächeln, das einen aus jeder noch so schlechten Stimmung reist.

Liebe Anja, vielen Dank, dass du an meiner Seite diesen Weg mitgegangen bist. Es war ein lehrreicher und schöner. Vielen Dank, dass du mich immer eingefangen hast und mich wieder in die Spur gesetzt hast. An allen Herausforderungen sind wir gemeinsam gewachsen und ich freue mich auf unsere Zukunft.

Rückersdorf, 23.12.2023 Fabian Dworschak

Inhaltsverzeichnis

Formelzeichen- und Abkürzungsverzeichnis . . . . . **ix**

1 Einleitung **1**

1.1 Problemstellung 2

1.2 Aufgabenstellung und Aufbau der Arbeit 4

2 Wissenschaftliche Grundlagen und Stand der Forschung . . **7**

2.1 Definitionen 7

2.2 Die Anpassungskonstruktion 9

2.2.1 Der Konstruktionsprozess 9

2.2.2 Die Einordnung nach Konstruktionsarten 12

2.2.3 Die Abgrenzung der Anpassungskonstruktion . . . 18

2.2.4 Zusammenfassung der Anpassungskonstruktion . . 18

2.3 Die Automatisierung des Konstruktionsprozesses 19

2.3.1 Die Charakterisierungsmöglichkeiten der Automatisierung 20

2.3.2 Inhärente Limitationen der Automatisierung 23

2.3.3 Die Forschungslandschaft der Automatisierten Anpassungskonstruktion 27

2.3.4 Die Rechnerunterstützen Systeme 29

2.3.5 Die Strukturoptimierung 34

2.3.6 Die Wissensbasierte Produktentwicklung 39

2.3.7 Das Maschinelles Lernern 44

2.3.8 Zusammenfassung Automatisierung 49

2.4 Das Selbstverstärkende Maschinelle Lernen 49

2.4.1 Die Entscheidungsprobleme nach Markov 50

2.4.2 Das Q-Learning 54

2.4.3 Das Künstliche Neuronale Netz 55

2.4.4 Das Deep Q-Learning 57

2.4.5 Das Selbstverstärkende Lernen in der Produktentwicklung 58

2.4.6 Zusammenfassung Selbstverstärkendes Lernen . . . 59

3 Ableitung des Handlungsbedarfs und der Forschungsfragen **61**

3.1 Fazit aus den wissenschaftlichen Grundlagen 61

3.2 Ableitung des Forschungsbedarfs 63

3.3 Ableitung der Forschungsfragen 64

4 Aufbau der Arbeit 65

5 Das Konzept des Selbstverstärkenden Lernens zur Automatisierung der Anpassungskonstruktion 69

5.1 Das Computer-verarbeitbare Modell der Anpassungskonstruktion 69
5.1.1 Das Modell der Anpassungskonstruktion 70
5.1.2 Die Wissensrepräsentation der Anpassungskonstruktion 74
5.1.3 Zusammenfassung des computer-verarbeitbaren Modells der Anpassungskonstruktion 76

5.2 Der Aufbau des Selbstverstärkenden Lernprozesses für die Anpassungskonstruktion 77
5.2.1 Die Komponenten und ihr Zusammenspiel 77
5.2.2 Die Relevanz der Nutzerinteraktion 83
5.2.3 Zusammenfassung des Zusammenspiels der Komponenten 87

5.3 Die Überführung von Konstruktionsaufgaben in Lernumgebungen 87
5.3.1 Die Darstellung der Merkmale als Parameter Grid . 88
5.3.2 Die Darstellung der Anforderungen als Belohnungsfunktionen 88
5.3.3 Zusammenfassung der Überführung der Konstruktionsaufgaben 89

5.4 Die Stellschrauben für Produktentwickelnde 89
5.4.1 Die Stellschrauben der zeitlichen Abfolge 91
5.4.2 Die Stellschrauben der Explorations-Strategie . . . 92
5.4.3 Die Stellschrauben der Netzarchitektur 93
5.4.4 Die Stellschrauben der Netzanpassung 93
5.4.5 Zusammenfassung der Stellschrauben 94

5.5 Indikatoren für das Lernverhalten und den Trainingsaufwand 96
5.5.1 Der Loss 97
5.5.2 Der Reward 98
5.5.3 Der Regret 99
5.5.4 Der Trainingsaufwand 100
5.5.5 Zusammenfassung der Indikatoren 100

5.6 Das Konzept für Übertragung auf nachfolgende Produktgenerationen 101

5.7 Zusammenfassung der vorgestellten Konzepte 103

6 Die Umsetzung des Modells und des Automatisierungswerkzeugs . . . 105
6.1 Die Modellerstellung . . . 106
6.1.1 Die Applikation für die Modellerstellung . . . 107
6.1.2 Die Anwenderinteraktion bei der Modellerstellung 110
6.1.3 Zusammenfassung zur Modellerstellung . . . 113
6.2 Die Steuerung des Selbstverstärkenden Lernprozesses . . . 114
6.2.1 Die Architektur des Automatisierungswerkzeugs . . 114
6.2.2 Die Anwenderschnittstelle zur Übergabe von Konstruktionsaufgaben . . . 116
6.2.3 Die Anwenderschnittstelle zur Einstellung der Hyperparameter . . . 118
6.2.4 Die Anwenderschnittstelle zur Evaluation der Indikatoren . . . 120
6.2.5 Zusammenfassung zur Steuerung des Selbstverstärkenden Lernprozesses . . . 120

7 Anwendungsbeispiele für automatisierte Anpassungskonstruktionen . . . 123
7.1 Das Modell des Anpassungsprozesses für individualisierte Produkte . . . 124
7.2 Machbarkeitsstudie . . . 127
7.2.1 Die Ergebnisse des Kurbelarms . . . 128
7.2.2 Die Ergebnisse der Umlenkwippe . . . 131
7.2.3 Die Ergebnisse des Bremshebels . . . 134
7.2.4 Zwischenfazit zur Machbarkeitsstudie . . . 137
7.3 Hyperparameterstudie . . . 137
7.3.1 Effekte der Hyperparameter der Explorations-Strategie 138
7.3.2 Effekte der Hyperparameter der Netzanpassung . . 139
7.3.3 Effekte der Hyperparameter der Netzarchitektur . . 141
7.3.4 Zwischenfazit zur Hyperparameterstudie . . . 141
7.4 Transferstudie . . . 142

8 Diskussion der Erkenntnisse hinsichtlich der Forschungsfragen . . . 145

9 Zusammenfassung und Ausblick . . . 153

10 Summary and Outlook . . . 157

Anhang . . . 161
Appendix: Exemplarische Umsetzung in Python . . . 161

Appendix: Zusätzliche Auswertungen der Anwendungsbeispiele 181
Literaturverzeichnis **182**

Formelzeichen- und Abkürzungsverzeichnis

Symbol	*Beschreibung*
CDS	Common Design Structure
CDSD	Common Design Structure Discoverya
CFD	Computational Fluid Dynamics
CRISP-DM	Cross-Industry Standard Process for Data Mining
DA	Design Automation
DQN	Double Q-learning Network
FEM	Finite Element Method
IGES	Initial Graphics Exchange Specification
JT	Jupiter Tesselation
KBE	Knowledge-Based Engineering
KBS	Knowledge-based Systems
KNN	Künstliches Neuronales Netz
LoA	Level of Automization
MDP	Markov Decision Process
ML	Machine Learning
MMO	Multi-Objective Optimization
OOP	Object-Oriented Programming
OWL	Web Ontology Language
RDF	Resource Description Framework
RL	Reinforcement Learning
ReLU	Rectified Linear Unit
STEP	STandard for the Exchange of Product model data)
UML	Unified Modeling Language
URI	Uniform Resource Identifier
XML	Extensible Markup Language

Symbol	***Beschreibung***
$f(x)$	Zielfunktion
$g_j(x)$	Ungleichheitsrestriktion
$h_l(x)$	Gleichheitsrestriktion
$P(x)$	Strafterm
p	Skalierungsfaktor d. Strafterms
s_t	Zustand zum Zeitpunkt t
a_t	Aktion zum Zeitpunkt t
r_t	Belohnung zum Zeitpunkt t
$P_{ss'}^{a}$	Übergangswahrscheinlichkeit von s in den Zustand s'
$R_{ss'}^{a}$	Erwartungswert für die Belohnung r_{t+1} beim Übergang von s nach s'
Π	Policy
$Q^{\pi}(s,a)$	Q-Value für den Zustand s, der Aktion a unter der Policy π
α	Lernrate
γ	Discount Faktor
ϵ	Schwellwert der Epsilon-Greedy Strategie
w	Kantengewicht
N_L	Anzahl der Schichten
N_K	Anzahl der Knoten
M_i	Merkmal
A_i	Anforderung

1 Einleitung

Neben politischen und makroökonomischen Randbedingungen ist der technologische Wandel ein entscheidender Taktgeber im Kampf um Wettbewerbsvorteile. Ausgehend von der Innovation bis hin zu deren wirtschaftlicher Umsetzung konkurrieren Unternehmen um Erkenntnisse, Erfahrungen und Einsatzmöglichkeiten im Umgang mit neuen Technologien. In der Produktentwicklung werden seit der Jahrtausendwende diese Trends durch die Informations- und Kommunikationstechnologien dominiert, weil sie einen rasanten Informationsaustausch innerhalb und über die Grenzen des Unternehmens hinaus ermöglichen und damit neue Geschäftsfelder oder neue Geschäftsmodelle wie individualisierte oder *As-a-Service* Produkte eröffnen [1, 2]. In Kombination mit hohen Personalkosten zwingt die damit einhergehende Informationsflut die Unternehmen zur Automatisierung ihrer Prozesse. Aus der Übergabe von Tätigkeiten vom Menschen an die Maschine entstehen enorme Veränderungen. Auf Seite der Arbeitnehmenden führt die Automatisierung zu einer Entlastung von repetitiven Routinetätigkeiten – Tätigkeiten, die definierten Regeln folgen und sich daher in Algorithmen übersetzen lassen [3]. Folglich rücken kreative Aufgaben, welche außerhalb klar definierbarer Regeln ablaufen, in den Vordergrund. Diese fordern und erhöhen die Motivation der Arbeitnehmenden. Umfragen aus den Jahren 2012 – 2015 zeigen, dass mehr als die Hälfte aller deutschen Arbeitnehmenden eine Steigerung ihrer Produktivität auf Grund von Informations- und Kommunikationstechnologien wahrnehmen [4]. Je höher das Bildungsniveau ist, desto signifikanter ist dieser Effekt. Auf Seite der Arbeitgebenden eröffnen sich unter anderem neue Geschäftsfelder, schnellere Reaktionszeiten und effizientere Nutzung der Personalkosten [5, 6]. Besonders riskant sind diese Veränderungen, weil sie einerseits tief in die Unternehmensprozesse eingreifen und anderseits einen dynamischer Prozess unter demografischen Einflussfaktoren beschreiben [7]. Aktuelle Beispiele für abrupte und unvorhersehbare Katalysatoren für den Einsatz von Informations- und Kommunikationstechnologien sind die derzeitige Rohstoffknappheit und die Maßnahmen gegen die globale COVID-19 Pandemie [8].

Mit dieser ökonomischen Motivation im Hintergrund ist es eine Aufgabe der Ingenieurswissenschaften, Automatisierungspotentiale zu schaffen, um schnell und flexibel auf Veränderungen eingehen zu können. Das heißt, technische Möglichkeiten zu erforschen, die es erlauben bislang durch den Menschen ausgeführte Tätigkeiten an die Maschine zu übergeben. Erforschen bedeutet hier den Brückenschlag zwischen Technologie und deren produktiven Einsatz zu beschreiben. Dazu zählen das Verstehen und Interpretieren

von Technologien aus anderen Wissenschaftsdomänen bis hin zur Identifikation von möglichen Anwendungsszenarien. In diesem breiten Feld beschreibt die Anpassungskonstruktion das Anpassen von Merkmalen zum Erfüllen neuer Anforderungen (s. Abschnitt 2.2) und ist auf Grund der eingeschränkten konstruktiven Freiheitsgrade eng mit dem Begriff der Routineaufgaben verknüpft [9, 10]. Sie ist in Bild 1 schematisch dargestellt. In Kombination mit dem hohen Anteil der Anpassungskonstruktionen in Unternehmen und der direkten Kopplung an individuelle Kundenwünsche ist sie ein äußerst relevantes Anwendungsgebiet für die Automatisierung.

1.1 Problemstellung

Seit der Einführung der Rechnerunterstützung in die Produktentwicklung wurde der Anteil der automatisierten Tätigkeiten innerhalb der Anpassungskonstruktion durch Informations- und Kommunikationstechnologien immer weiter vergrößert. Im letzten Jahrzehnt haben Sensorisierung, Vernetzung sowie Web- und Cloud-Technologien zu einer hohen Verfügbarkeit von Daten aus allen Phasen des Produktlebenszyklus geführt. Im Sinne des *Predictive Engineering* nach WARTZACK ermöglicht die datengetriebene Produktentwicklung eine Rückführungen von Daten aus späten Phasen vergangener Produkte zur frühzeitigen, virtuellen Unterstützung der Entwicklung zukünftiger Produkte [11]. Datengetriebene Ansätze eignen sich besonders als Analysewerkzeug d. h. zur Abschätzung potentieller Produkteigenschaften auf Basis definierter Merkmale. Ihr Mehrwert ergibt sich aus der frühzeitigen, virtuellen Abschätzung der Produkteigenschaften ohne den Einsatz anderer kostenintensiverer Absicherungswerkzeuge wie beispielsweise numerische Simulationen. Außerdem können maschinelle Lernprozesse Zusammenhänge approximieren, die mit anderen Analysewerkzeugen nicht erfasst werden

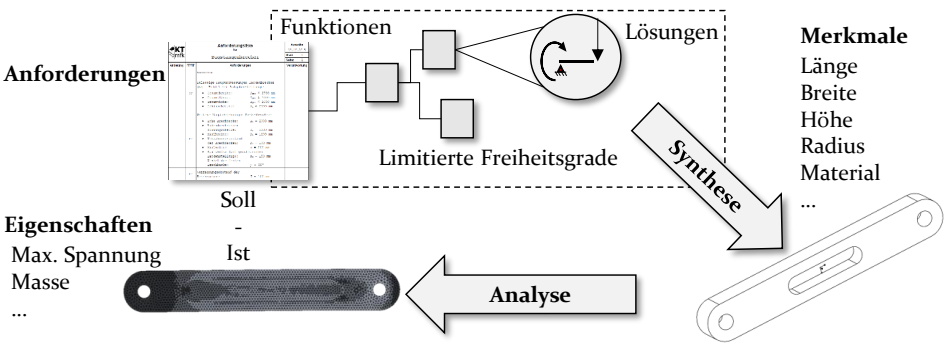


Bild 1: Darstellung der Anpassungskonstruktion als iterativer Prozess aus Analyse und Synthese unter limitierten konstruktiven Freiheitsgraden

Trotz der vielseitigen Methoden und Werkzeuge kommen datengetriebene Ansätze nur eingeschränkt als Synthesewerkzeug zum Einsatz, wodurch die Automatisierungspotentiale der Anpassungskonstruktion stark limitiert werden. Hierfür werden in dieser Arbeit hierfür zwei Gründe betrachtet – Übertragbarkeit und Skalierbarkeit. Zum einen werden Automatisierungswerkzeuge größtenteils als Insellösungen für einen speziellen Anwendungsfall entwickelt [3, 15]. Die so erlangten Erkenntnisse sind ohne menschliche Interaktion oder erneute Berechnung nicht auf weitere Anwendungsfälle übertragbar. Ein Beispiel hierfür sind Optimierungsalgorithmen für die Synthese von Merkmalsausprägungen [12, 14]. Diese müssen bei einer Änderung der Randbedingung neu berechnet werden. Das Interpolieren zwischen bereits gesammelten Erkenntnissen ist ohne weitere Modelle nicht möglich. Der Berechnungsaufwand für jede neue Erkenntnis bleibt folglich konstant, obwohl davon auszugehen ist, dass für ähnliche Produkte ähnliche Zusammenhänge zwischen Merkmalen und Eigenschaften gelten müssen. Die Wiederverwendung der aufwändig gesammelten Erkenntnisse ist daher aufwändig und bleibt häufig aus.

Eine weitere Einschränkung ist die vergleichsweise schlechte Skalierbarkeit von datengetrieben Ansätzen bei der Anwendung auf komplexere Produkte, da hier die Anzahl der Eigenschaften und typischerweise auch die der Merkmale steigt. Diese müssen unter Umständen gleichzeitig abgeschätzt werden, beispielsweise, wenn mehrere Merkmale zusammen eine Eigenschaft beeinflussen oder wenn ein Merkmal Einfluss auf verschiedene Zielgrößen hat. Im Falle der Analyse spannen die Merkmale den Raum der möglichen Lösungen auf. Die Eigenschaften bilden die Zielgrößen. Im Falle der Synthese ist dies umgekehrt. Ein Produkt ist eine Kombination aus Merkmalen sowie deren resultierender Eigenschaften und stellt eine Instanz des Lösungsraums dar. Alle Instanzen vergleichbarer Produkte bilden gemeinsam die Produkthistorie. Bei einer großen Anzahl von Merkmalen und/oder Eigenschaften müssen folgende Überlegungen in Betracht gezogen werden. Einerseits können Meta-Modelle wie Kriging (Gaußprozesse), Radial Basis Function, Support Vector Regression and Moving Least Square nur bis zu einer definierten Anzahl von Zielgrößen eingesetzt werden. Die exakte Anzahl der Zielgrößen ist zwar spezifisch für den Anwendungsfall, jedoch wird in der Literatur acht als obere Grenze definiert [16–21]. Im Falle der Synthese bedeutet das, dass maximal acht Merkmale gleichzeitig abgeschätzt werden können. Andererseits können flexiblere Modelle wie beispielsweise Künstliche Neuronale Netze (KNN) nur eingesetzt werden, wenn ein entsprechend großer, aufbereiteter Datensatz vorhanden ist. Aus mathematischer Sicht liegen die Gründe hierfür im

sogenannten *Curse of Dimensonality* nach BELLMAN [22][1]. Dieser beschreibt die Grundlage verschiedener Phänomene, die auftreten, wenn der Lösungsraum höherdimensional ist und wird in 2.3.3 auf Seite 47 detailliert erörtert. Im Falle von datengetrieben Ansätzen bedeutet dies, dass die Anzahl der gleichzeitig zu bestimmenden Merkmale oder Eigenschaften von der Anzahl der vorhandenen vergleichbaren Produkte abhängig ist [25–27]. Sind nicht ausreichend Produkte vorhanden, ist die Wahrscheinlichkeit ein aussagekräftiges Modell zu erlangen gering. Folglich wird der verhältnismäßig große Teil von Produkten mit einer hohen Anzahl von Merkmalen bei niedriger Anzahl von vergleichbaren Produkten noch unzureichend abgedeckt. Beispielsweise hat die in Bild 1 dargestellte Fahrradkurbel bereits acht Merkmale.

Beide Gründe limitieren die Automatisierung der Anpassungskonstruktion. Diese sind in der Praxis jedoch sehr weit verbreitet und äußerst relevant für die Wettbewerbsfähigkeit vieler Unternehmen. Daher müssen für diese Lücken weitere Ansätze gefunden werden. Die folgende Arbeit soll einen Beitrag zur Steigerung des Automatisierungspotentials der Anpassungskonstruktion durch die Erforschung grundlegender Erkenntnisse des Selbstverstärkenden Lernens in der Produktentwicklung leisten. Dabei soll Selbstverstärkendes Lernen für Anwendungsszenarien ohne ausreichende Datengrundlage oder mit hoher, geforderter Übertragbarkeit eingesetzt werden, um die bestehenden Automatisierungswerkzeuge zu ergänzen.

1.2 Aufgabenstellung und Aufbau der Arbeit

Aufgrund der eingeschränkten Übertragbarkeit und Skalierbarkeit sind datengetriebene Lösungen bei der Synthese der Produktmerkmale immer noch limitiert. Dies reduziert die Automatisierungsmöglichkeiten der Anpassungskonstruktion und mindert das Rationalisierungspotenzial. Daher sollen beide Punkte durch ein neues Automatisierungswerkzeug verbessert werden. Im Kern dieser Arbeit steht die Erforschung des Selbstverstärkenden Lernens als alternatives Automatisierungswerkzeug. Selbstverstärkendes Lernen beruht auf Erfahrungen mit einer Umgebung und nicht auf bestehenden Daten, wodurch die Abhängigkeit von der Produkthistorie entfällt und die Skalierbarkeit nur in Relation zum Berechnungsaufwand zu betrachten ist. Die zu erlernende Umgebung ist ein Modell der Anpassungskonstruktion bestehend aus Synthese der Merkmale, Analyse der Eigenschaften und Vergleich mit den Anforderungen, wie sie in Bild 1 dargestellt ist. Des weiteren beschreibt die Anpassungskonstruktion den Prozess der Anpassung eines vorgegangen

[1] Im Bereich maschnielles Lernen auch unter den Begriffen *Peaking Phenomenon* [23] oder *Hughes Phenomenon* [24] beschrieben

Produkts auf neue Anforderungen. Daher soll das Automatisierungswerkzeug so gestaltet werden, dass gefundenen Erkenntnisse im Umgang mit dem vorangegangen Produkt für das nachfolgende Produkt zur Verfügung stehen, um die Übertragbarkeit zu verbessern.

Dazu wird wie in Bild 2 dargestellt vorgegangen. Die wissenschaftlichen Grundlagen umfassen drei Teilaspekte. Zunächst sollen die wissenschaftlichen Grundlagen der Anpassungskonstruktion erarbeitet und zusammengefasst werden. Die Anpassungskonstruktion reduziert den Produktentwicklungsprozess auf das Gestalten und Ausarbeiten der Merkmale. Folglich soll der zweite Teil der Grundlagen die Automatisierungswerkzeuge, welche für das Gestalten eingesetzt werden können, beschreiben. Dies soll einerseits die Limitationen der Automatisierungswerkzeuge aufzeigen. Anderseits sollen dabei die Werkzeuge betrachtet werden, welche Grundbausteine für das Selbstverstärkende Lernen liefen. Der letzte Abschnitt soll sich auf das Selbstverstärkende Lernen selbst konzentrieren. Dabei soll explizit auf die Stellschrauben der Produktentwickelnden eingegangen werden, da sie bekannt sein müssen, um das Lernverhalten zu kontrollieren und zu steuern. Abschließend wird aus der Forschungslandschaft der Handlungsbedarf und die wissenschaftliche Fragestellung herausgearbeitet.

Im Hauptteil dieser Arbeit soll zunächst das in Abschnitt 2.2 zusammengefasste Verständnis der Anpassungskonstruktion in ein computerverarbeitbares Modell übersetzt werden. Dies soll dem Selbstverstärkenden Lernen als Lernumgebung dienen. Die Vorstellung des Selbstverstärkenden Lernens teilt sich in Konzept und dessen Umsetzung. Das Konzept soll vor allem herausstellen, welche Berührungspunkte sich mit den Produktentwickelnden bilden. Deren Aufgabe ist das Einschätzen, Kontrollieren und Kalibrieren des Selbstverstärkenden Lerners durch sogenannte Hyperparameter. Auf diese Punkte geht die Umsetzung durch die Vorstellung von Kontrollwerkzeugen ebenfalls explizit ein. Die Effekte der Hyperparameter sollen anschließen an Demonstratoren gezeigt, um die Rolle der Produktentwickelnden zu betonen. Die Ergebnisse und Erkenntnisse sind kritisch zu diskutieren. Abschließend wird die Arbeit zusammengefasst und ein Ausblick gegeben werden.

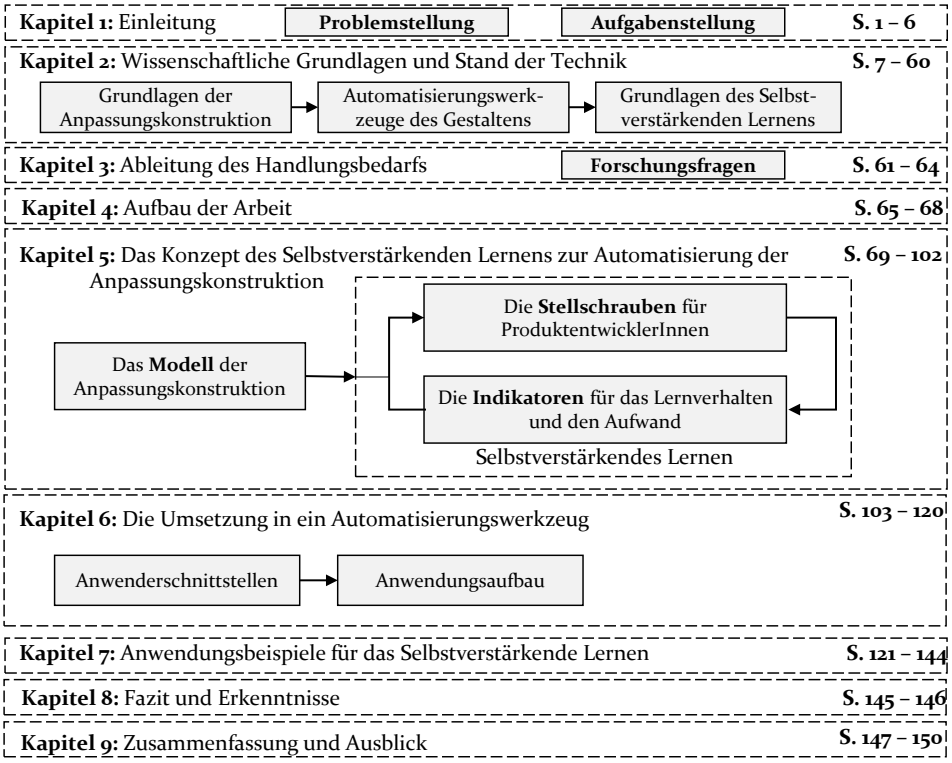


Bild 2: Aufbau dieser Arbeit

2 Wissenschaftliche Grundlagen und Stand der Forschung

Innerhalb dieses Kapitels werden die benötigten wissenschaftlichen Grundlagen und deren Stand der Technik vorgestellt. Sie unterteilen sich in drei Abschnitte. Zuerst werden die wissenschaftlichen Grundlagen der Anpassungskonstruktion analysiert und zusammengefasst. Die Anpassungskonstruktion wird als Konstruktionsart verstanden. Sie kann beispielsweise von der Neukonstruktion durch eine starke Einschränkung der konstruktiven Freiheitsgrade unterschieden werden. Um sie zu automatisieren muss ein tiefes Verständnis der Grenzen und Möglichkeiten der Anpassungskonstruktion vorliegen. Folglich beantwortet Abschnitt 2.2 die Frage, was automatisiert werden soll. Im anschließenden Abschnitt wird die Automatisierung im Kontext der Anpassungskonstruktion untersucht. Dabei konzentriert sich diese Arbeit auf die virtuelle Produktentwicklung und lässt die physikalische Interaktion außen vor. Aus dieser Einschränkung ergibt sich ein starker Fokus auf Informations- und Kommunikationstechnologien, wie beispielsweise Optimierung, Maschinelles Lernen und Wissensbasiertes Konstruieren. Es werden deren technologische Grundlagen, welche in dieser Arbeit genutzt werden, und deren aktuelle Forschung vorgestellt. Abschnitt 2.3 beantwortet die Frage, was im Rahmen der Anpassungskonstruktion und nach derzeitigem Stand bereits automatisierte werden kann. Im dritten Abschnitt werden die Grundlagen des Selbstverstärkenden Lernens vorgestellt, da dieses als zusätzliches Werkzeug für die Automatisierung der Anpassungskonstruktion in dieser Arbeit erschlossen werden soll. Da nach derzeitigem Stand noch kaum Selbstverstärkendes, Maschinelles Lernen in der Produktentwicklung eingesetzt wird, stellt Abschnitt 2.4 dessen Grundlagen und deren Stand der Forschung vor. Vorab werden Begriffe vorgestellt und definiert, welche von Beginn an wichtig für das Verständnis der Arbeit sind.

2.1 Definitionen

Diese Arbeit beinhaltet Begriffe der Produktentwicklung, der Automatisierung und des Selbstverstärkenden Lernens. Es werden somit verschiedene Fachdisziplinen adressiert, in welchen für jeden Begriff mehrere Definitionen gefunden werden können. In diesem Abschnitt sind deshalb ausgewählte Interpretationen vorweggenommen. Ziel dieses Abschnitts ist nicht die fachliche Definition der Begriffe, sondern die Einordnung und die Klarstellung des Verwendungskontextes der Begriffe in dieser Arbeit.

Eine **Anpassungskonstruktion** ist eine Konstruktionsart. Die Spezifizierung wird nach Innovationsgrad vorgenommen, da dieser ein bestimmender Kosten- und damit Risikofaktor in der Produktentwicklung ist [9]. Der Produktentwicklungsprozess wird limitiert, um innovative oder kreative Aktivitäten zur Lösungsfindung zu beschränken oder auszuschließen. Somit schließt die Anpassungskonstruktion die Neukonstruktion des gesamten Produktes aus [10]. Sie erfordert jedoch, dass der gesamte Produktentwicklungsprozess bereits vor der Anpassungskonstruktion mindestens einmal vollständig durchlaufen wurde und setzt folglich eine initiale Neukonstruktion voraus [28].

Die **Prinzipielle Lösung** ist ein Meilenstein in der Produktentwicklung. Sie grenzt die Lösungssuche von deren Gestaltung ab und enthält mindestens die Definition der verwendeten Lösungsprinzipien sowie deren Struktur. Als Bindeglied zwischen den anforderungserfüllenden Funktionen und den definierbaren Merkmalen der physikalischen Produktbeschreibung bildet sie die quantifizierbaren Zusammenhänge der verwendeten physikalischen Effekte ab. [29]

Eine **Konstruktionsaufgabe** ist eine Tätigkeit des Produktentwicklungsprozesses, welcher notwendig ist, um von einem unerwünschten Ausgangszustand zu einem erwünschten Sollzustand zu gelangen ohne, dass dabei die Überwindung einer Barriere notwendig ist. Hieraus ergibt sich die Voraussetzung, dass sowohl Ziel als auch der Weg dorthin bekannt sein muss. [29]

Die **Automatisierung** ist ein evolutionärer Prozess, welcher die Übergabe von Aktivitäten vom Menschen an die Maschine beschreibt [30]. Im Rahmen der Produktentwicklung muss differenziert werden, ob den Aktivitäten Aufgaben oder Probleme zugrunde liegen. Probleme beinhalten eine Barriere und setzen kreatives Denken voraus. Sie bedingen nach dem momentanem Stand der Technik den menschlichen Produktentwickelnden. Erst wenn die benötigten Zusammenhänge nach einer festgelegten Methodik erfasst und bearbeitet werden können, können diese an die Maschine übergeben werden. Im Hinblick auf die Produktentwicklung wird im Englischen von *Design Automation* gesprochen.

Das **Wissensbasierte Konstruieren** (engl. *Knowledge-Based Engineering*, KBE) ist ein Forschungsbereich, welcher die Methoden und Technologien des Erfassens und Wiederverwendens von Produkt- oder Prozesswissen untersucht [3]. Im Sinne des *Predictive Engineerings* beschreibt KBE die Rückführung von Wissen aus vorangegangenen Produktentwicklungen in zukünftige [11].

Das **Maschinelle Lernen** (engl. *Maschine Learning*, ML) ist eine Teildisziplin des Forschungsbereichs der Künstlichen Intelligenz, welche die Methoden und Technologien des Anpassens an neue Umstände sowie des Erkennens und Extrapolierens von Mustern beinhaltet [31].

Das **Selbstverstärkende Lernen** (engl. *Reinforcement Learning*, RL) ist ein Paradigma des Maschinellen Lernens, welches das zielorientierte Lernen eines Agenten auf Basis von Interaktionen mit einer Umgebung beschreibt [32]. Im einfachsten Fall besteht eine Interaktion aus der Aktion des Agenten, welche die Umgebung in einen neuen Zustand bringt und als Antwort auf die Aktion eine Belohnung erhält. Um die Belohnungen zu maximieren, lernt der Agent eine Strategie, welche als Verhalten des Agenten für einen definierten Zustand verstanden werden kann.

2.2 Die Anpassungskonstruktion

Der Produktentwicklungsprozess ist ein zielgerichteter Problemlösungsprozess, der sich in Aktivitäten unterteilen lässt [29]. Aktivitäten können in verschieden Abstraktionsgraden beschrieben werden. Beispielsweise ist die Synthese von Lösungen eine generische Beschreibung einer Aktivität und die Übertragung spezifizierter Daten in eine Tabelle eine sehr detaillierte Beschreibung. Ebenso bilden Aktivitäten die Grundlage der Automatisierung. Dies wird in Unterabschnitt 2.3.1 genauer ausgeführt. Für diese Arbeit definiert die Anpassungskonstruktion die Aktivitäten, welche automatisiert werden sollen und somit den Kontext für diese Arbeit. Die Automatisierung erfordert eine maschinenlesbare und -verarbeitbare Darstellung der Aktivitäten. Diese abstrahierte, theoretische Darstellung wird in Kapitel Abschnitt 5.1 als Modell der Anpassungskonstruktion erarbeitet. Hierfür ist eine Charakterisierung der Anpassungskonstruktion notwendig. Daher erfolgt in diesem Kapitel zunächst eine knappe Vorstellung der Konstruktionsarten. Im Anschluss werden die für die Einteilung benötigten Kriterien zusammengetragen. Dies ist notwendig, um im Zwischenfazit festzuhalten, welche Möglichkeiten das Model der Anpassungskonstruktion abdecken sollte und durch welche Grenzen es limitiert wird.

2.2.1 Der Konstruktionsprozess

Die Tätigkeit des Konstruierens umfasst alle Aktivitäten der Synthese und Analyse, die notwendig sind, um auf der Basis einer definierten Aufgabenstellung die zu einem bestimmten Zeitpunkt bestmögliche Lösung zu beschreiben [33]. Dieser Prozess ist in seiner abstraktesten Form in Bild 3 skizziert. Dabei beschreiben Aktivitäten zielgerichtetes Denken und Handeln [29].

Die Aktivitäten der Synthese beinhalten zum einen die qualitative Definition relevanter Merkmale (Entwicklung und Auswahl von Prinziplösungen, qualitative Gestaltung) und zum anderen die quantitative Festlegung der Merkmalausprägungen (Dimensionierung, quantitative Gestaltung) [34]. Die Aktivitäten der Analyse beschreiben die Informationsgewinnung durch Zerlegen und Aufgliedern sowie durch Untersuchen der Eigenschaften einzelner Elemente und der Zusammenhänge zwischen ihnen [10]. Es wird in Situations-/Problemanalyse und der Analyse von Lösungen unterschieden [10, 29]. Eigenschaften ergeben sich aus den Merkmalen und deren Ausprägungen. Sie werden anhand des virtuellen Produktes durch Simulationen und Berechnungen oder anhand physikalischer Prototypen durch Versuche untersucht [9, 35].

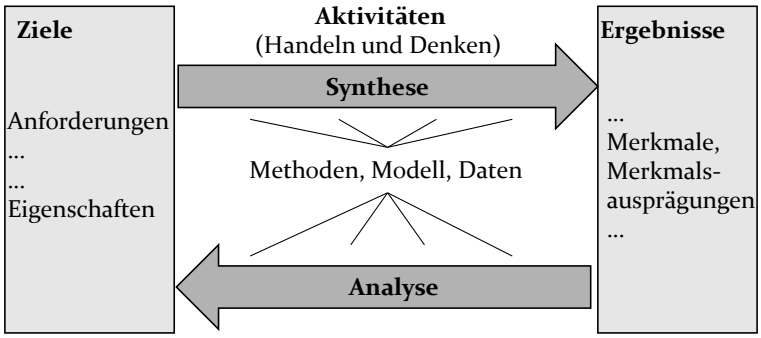


Bild 3: Aktivitäten des Konstruktionsprozesses in Anlehnung an [10, 29]

Da komplizierte Produkte nur eine Vielzahl in sich verschachtelter und iterativer Aktivitäten realisieren lassen, wurden zahlreiche Methoden und Modelle entwickelt, um den Konstruktionsprozess unter Berücksichtigung seiner besonderen Merkmale zu verstehen, zu verbessern und zu unterstützen. Die Komplexität ist jedoch so groß, dass kein einziges Modell alle Probleme lösen kann [36]. Darüber hinaus unterscheiden sich die vielen entwickelten Modelle in ihrer Ausrichtung und Formulierung [37]. Obwohl neuere Theorien und Modelle immer differenziertere Aspekte des Konstruktionsprozesses beleuchten, lässt sich ein großer Anteil auf die frühen Prozessmodelle nach HANSEN, KESSELRING, HUBKA, KOLLER, PAHL & BEITZ, ROTH und EHRLENSPIEL & MEERKAMM zurück führen [9, 10, 38–41]. Diese stammen fast alle aus dem Maschinenbau und wurden in Europa, vor allem, aber nicht ausschließlich im deutschsprachigen Raum, entwickelt [42]. WALLACE & BLESSING zeigen den Einfluss auf die internationalen Weiterentwicklungen um die Jahrtausendwende [43]. Eine detaillierte Analyse der Modelle des Produktentwicklungsprozesses findet sich in [36]. Obwohl sich die frühen Prozessmodelle ebenfalls in Bezug auf Hintergrund und Schwerpunktsetzung

unterscheiden, wurde eine gemeinsame Schnittmenge in den VDI-Richtlinien 2221 und 2222 formuliert [29, 44]. Kern all dieser Ansätze ist ein generisches Phasenmodell des Konstruktionsprozesses, das mit der Aufgabenklärung beginnt, über Funktions- und Logiküberlegungen geht und mit der Strukturierung in realisierbare Module sowie deren detaillierter Ausarbeitung endet und in Bild 4 dargestellt ist. Ziel der Ansätze ist die Unterstützung der Produktentwickelnden, insbesondere bei der systematischen Entwicklung von Lösungsalternativen, deren Bewertung (und anschließender Entscheidungsfindung) und der systematischen Detaillierung der gewählten Lösungen. In den meisten Fällen konzentrieren sich die Theorien und Modelle auf die Entwicklung eines neuen Produktes, obwohl diese Art der Konstruktions

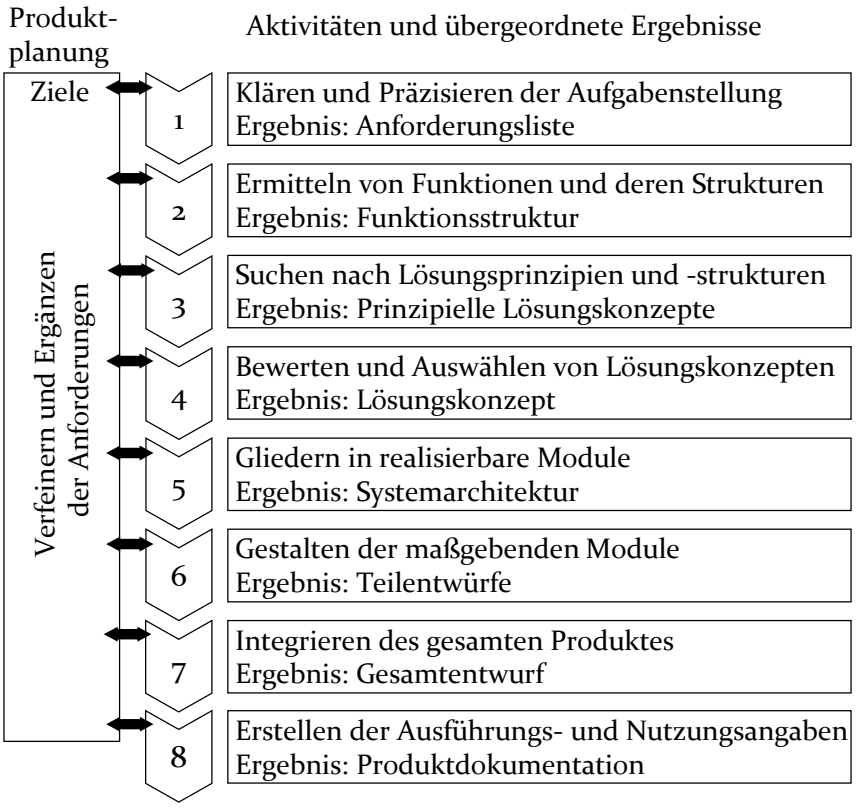


Bild 4: Allgemeines Prozessmodell des Konstruktionsprozesses nach der VDI 2221-1 [29]

2.2.2 Die Einordnung nach Konstruktionsarten

Der Begriff Konstruktionsart ist entweder ein vorgegebener Einflussfaktor auf den Konstruktionsprozess oder als dessen bestimmbares Merkmal zu verstehen [9, 45]. Folglich wird unterschieden, ob ein Konstruktionsprozess eine Konstruktionsart bedingt oder, ob die diese vor oder nach dem Durchlaufen des Konstruktionsprozess festgelegt wird. Erste Sichtweise ist demnach für die Analyse eines existierendes Prozesses geeignet und die Zweite für dessen Planung. In beiden Fällen ermöglicht die Spezifizierung eine zielgerichtetere, methodische Unterstützung des Konstruktionsprozesses. Diese ist vor allem in Anbetracht der herrschenden Prozess- und Produktdiversität erforderlich, um geeignete Methoden für den Konstruktionsprozess auszuwählen [10]. In der deutschsprachigen Literatur findet sich vorrangig die Unterteilung in die Konstruktionsarten Neu-/ Anpassungs-/ Varianten-/ Wiederholkonstruktion [10, 29]. Synonym dazu ist die Unterteilung in Neu-/ Anpass-/ Varianten-/ Wiederholentwicklung, die auf der Vorherigen aufbaut [46]. Es wird in folgenden Konstruktionsarten unterschieden. Eine Übersicht der groben Einteilung in Konstruktionsarten ist in Bild 5 dargestellt.

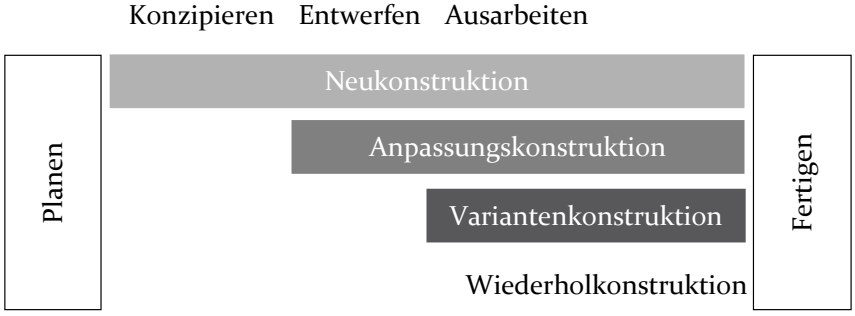


Bild 5: Unterscheidung in Konstruktionsarten nach EHRLENSPIEL & MEERKAMM [9, 47]

Eine Neukonstruktion liegt vor, wenn die drei Phasen Konzipieren, Entwerfen und Ausarbeiten vergleichbar neu bearbeitet werden müssen [9]. Es werden dabei komplett neue Aufgaben und Probleme gelöst oder eine Konstruktion mit neuen Lösungsprinzipien oder einer Neukombination bekannter Lösungsprinzipien erstellt [10]. Eine Neukonstruktion muss durchgeführt werden, wenn eine völlig neue Problemstellung gegeben ist [28], Sie beinhaltet keine Einschränkung der Freiheitsgrade [46] und es gibt kein Vorgängerprodukt [28]. Eine Anpassungskonstruktion liegt vor, wenn das Lösungsprinzip erhalten bleibt und die Gestalt an neue Randbedingung angepasst wird [10]. Demnach existiert ein Vorgänger und die konstruktiven Freiheitsgrade sind teilweise blockiert [28, 46]. Im Gegensatz zur Neukonstruktion wird die Phase Entwerfen nicht neu durchlaufen [9]. Die Variantenkonstruktion liegt

vor, wenn zusätzlich zur Prinzipiellen Lösung die Gestalt und der Werkstoff vordefiniert sind [9]. Außerdem existiert ein verwandter Vorgänger [28]. Die konstruktiven Freiheitsgrade sind noch stärker eingeschränkt und es wird lediglich die Phase der Ausarbeitung durchlaufen [9, 46]. Die Wiederholkonstruktion liegt vor, wenn ein neuer Fertigungsanlauf für ein bereits früher konstruiertes und gefertigtes Produkt durchgeführt wird [10].

Der Vollständigkeit halber sei erwähnt, dass neben diesen vier Konstruktionsarten STEINHILPER & RIEG weiter in die Weiterentwicklung und die Innovation unterscheiden. Wobei bei einer Weiterentwicklung eine bekannte Lösung um eine neue Funktion ergänzt wird und für eine Innovation eine neue Lösung für eine neue Funktion gefunden werden muss [48]. Dies sei der Vollständigkeit halber erwähnt.Der Begriff Innovation wird im Folgenden erneut aufgegriffen, da die hier genannte Auffassung noch durch wirtschaftlichen Aspekte (wie beispielsweise der Markterfolg) ergänzt werden muss. Außerdem wird im Folgenden die Wiederholkonstruktion außen vorgelassen, da sich alle weiteren Überlegungen dieser Arbeit auf den Produktentwicklungsprozess konzentrieren.

Bei der Betrachtung von Industriestudien [49–51] wird klar, dass in der industriellen Praxis eine exakte Einordnung in die Konstruktionsarten schwer möglich ist, weil sie zusätzlich von einer Vielzahl von unternehmensspezifischen Aspekten abhängig ist. Weitere Hürden, welche die Einordnung erschweren sind, dass der Konstruktionsprozess nicht vollständig vorhergesehen werden kann, da vorab Entscheidungen häufig auf der Grundlage unzureichender oder vorläufiger Informationen getroffen werden müssen [52]. Außerdem ist jeder Konstruktionsprozess einzigartig und mit einem gewissen Maß an Unsicherheit verbunden [53]. Der Konstruktionsprozess muss dynamisch betrachtet werden, da neue Aktivitäten in der Regel erst während des Projekts vollständig erfasst werden können [54]. Folglich ist der Prozessablauf nicht exakt vorhersehbar, da die Aufgaben im Laufe der Arbeit nach und nach konkretisiert und angepasst werden [55]. Sogar bei der theoretischen Einordnung gibt es Grauzonen und die Grenzen zwischen den einzelnen Konstruktionsarten können nicht scharf gezogen werden. Beispielsweise darf nach [9] eine Anpassungskonstruktion eines komplizierten Produktes die Neukonstruktion einer untergeordneten Komponente umfassen. Des weiteren führen Konzepte wie Simultaneous und Concurrent Engineering zu parallelen Aktivitäten aus verschiedenen Phasen des Produktentwicklungsprozesses [29]. Eine klare Trennung zwischen den Phasen des Produktentwicklungsprozesses ist nur noch bedingt möglich. Dies bestärkt die VDI 2221-2, welche anhand von Stereotypen die Einordnung der Phasen mit den Aktivitäten des Konstruktionsprozesses auf einer zeitlichen Schiene gegenüberstellt [56]. Im Rückblick auf die Aufgabenstellung

dieser Arbeit reicht die grobe Aufteilung, wie sie in Bild 3 auf 10 dargestellt ist, nicht aus. Es muss also eine Aktivität und die mit ihr verbunden Tätigkeiten der Produktentwickelnden analysiert werden, um sie der Anpassungskonstruktion zuordnen zu können. Um genauer differenzieren zu können, wird die Herkunft dieser Definitionen näher betrachtet. Aus ihnen lassen sich vier Indikatoren ableiten: die Anzahl der Limitationen hinsichtlich konstruktiver Freiheitsgrade, die Bearbeitungstiefe, der Einfluss des Vorgängerproduktes und der Neuheits- bzw. Innovationsgrad.

Die Limitation der konstruktiven Freiheitsgrade

Die Limitation der konstruktiven Freiheitsgrade wird vor allem im Kontext des Münchner Konkretisierungsmodells bzw. dem zugrundeliegenden Lösungsraum nach RUDE verwendet [57]. Innerhalb dieses dreidimensionalen Beschreibungsmodells wird der Entwicklungsprozess durch den Konkretisierungsgrad anhand der genutzten Produktmodelle beschrieben [46]. Produktentwickelnde bewegen sich auf den Dimensionen (1: abstrahieren – konkretisieren, 2: variieren – einschränken, 3: zerlegen – zusammenführen) durch den Lösungsraum nach RUDE, welches in Bild 6 dargestellt ist [57]. Limitationen der Freiheitsgarde sind demnach Einschränkungen der Bewegungsmöglichkeiten. Wird die Prinzipielle Lösung fixiert, so ist eine Abstraktion über die Prinzipebene hinaus nicht möglich. Die verbleibenden zwei Achsen für variieren – einschränken und zerlegen – zusammenführen sind insofern eingeschränkt, dass Merkmalsausprägungen in der Gestaltebene nicht mehr erreichbar sind, weil sie eine Änderung der Prinzipiellen Lösung erfordern würden.

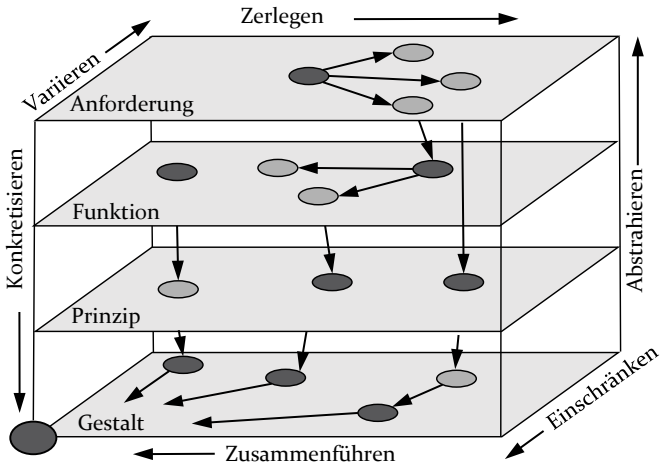


Bild 6: Modellraum des Konstruierens nach RUDE [57]

Die Bearbeitungstiefe

Die Bearbeitungstiefe ergibt sich aus der Variabilität der produktbestimmenden Merkmale [9]. EHRLENSPIEL & MEERKAMM nennen acht Kategorien für produktbestimmende Merkmale und drei Ausprägungen für die Variabilität (siehe Bild 7). Da sich die Merkmale aus den jeweiligen Meilensteinen des Produktentwicklungsprozesses ergeben, legt die Bearbeitungstiefe auch fest, welche Phasen neu durchlaufen werden müssen. Synonym zu Variabilität wird auch hier der Begriff Neuheit der Merkmale verwendet. Die Ausprägungen der Merkmale „neu“ , „ahnlich wie“ und „gleich wie“ spiegeln den Bezug zu vorangegangen Konstruktionen wieder. Dabei muss berücksichtigt werden, dass die Einteilung in Phasen von Produkt, Branche und Organisation auf unterschiedlich Art und Weise erfolgen kann [29, 56]. Die Bearbeitungstiefe sollte demnach auf Basis der Aktivitäten individuell erfasst werden.

	Merkmal	Neuheitsgrad		
		Neu	Ähnlich wie	Gleich wie
1	Anforderungen, Restriktionen			
2	Prinzipielle Lösung			
3	Qualitative Gestalt			
4	Quantitative Gestalt			
5	Werk-, Betriebsstoff, Oberflächenbehandlung			
6	Teilefertigung			
7	Montage			
8	Entsorgungsverfahren			

Bild 7: Einteilung der Konstruktionarten nach Bearbeitungstiefe nach EHRLENSPIEL & MEERKAMM [9] (Anpassungskonstruktion in schwarz, Variantenkonstruktion in dunkel grau und Neukonstruktion in hellgrau)

Der Einfluss des Vorgängerproduktes

Der Einfluss des Vorgängerproduktes wird vor allem in [28] als zusätzlicher Indikator für die Konstruktionsart hervorgehoben. Er beschreibt die Ergebnisse, welche aus vorangegangen Produktentwicklungsprozessen mit übernommen werden können. Dies bezieht sich auf physische Ergebnisse, die sich

beispielsweise durch die Meilensteine des Produktentwicklungsprozesses

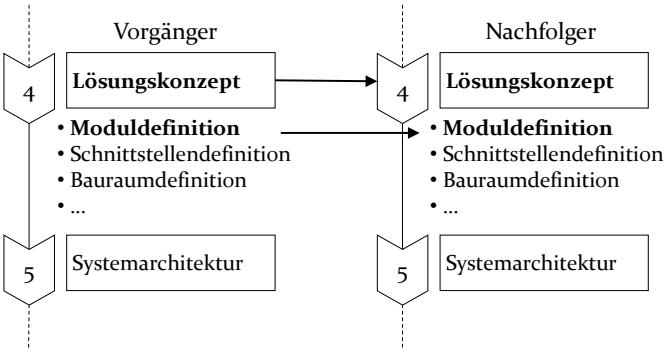


Bild 8: Schematische Darstellung der Einfluss des Vorgängerproduktes auf seinen Nachfolger durch die Übernahme von Arbeitsergebnissen

Der Neuheitsgrad

Der Neuheitsgrad beschreibt den Grad der Änderung, den ein Produkt erfüllen muss, um die Anforderungen eines neuen Produktentwicklungsprozesses zu erfüllen [58]. Der Neuheitsgrad legt die Grundlage für den Innovationsgrad, welcher zusätzlich die wirtschaftliche Verwertbarkeit des Produktes berücksichtigt und die Grundlage der Einteilung in [10] bildet. Nach [59] wird der inhaltliche Neuheitsgrad mit der Intensitäts-, der Akteursdimension, der subjektiven, der prozessualen und der normativen Dimension erweitert, um das Risiko der Konstruktion unternehmerisch absichern zu können. RAJAPATHIRANA & HUI zeigen, wie vielfältig die Einflussgrößen und deren Relationen auf den Innovationsgrad aus ökonomischer Sicht sind [60]. Der Innovationsgrad ist in der industriellen Praxis ein dominanter Indikator, weil er den Wettbewerbsvorteil des Unternehmens direkt beeinflusst [50].

Sowohl aus den beschriebenen Definitionen der Konstruktionsarten als auch für deren Indikatoren lassen sich Relationen ableiten. Diese lassen sich aus den beiden eingangs genannten Perspektiven veranschaulichen. Bei der vorausgehenden strategischen Ausrichtung der Produktentwicklung wird Bild 9 von oben nach unten durchlaufen. Der angestrebte Innovationsgrad definiert zusammen mit den unternehmensspezifischen Kontextfaktoren den Neuheitsgrad für eine zukünftige Produktentwicklung. Die Limitation der konstruktiven Freiheitsgrade ist eine abstrakte Darstellung des Neuheitsgrades, weil sich der Lösungsraum des Produktentwicklungsprozesses nur in sehr eingeschränkten Fällen konkret darstellen lässt. Die Bearbeitungstiefe

und der Einfluss der Vorgänger sind Möglichkeiten, den Neuheitsgrad zu quantifizieren und daher konkret darzustellen. Bei der zweiten Perspektive – der Analyse eines Produktentwicklungsprozesses – wird Bild 9 von unten nach oben durchlaufen. Bearbeitungstiefe und Einfluss des Vorgängerproduktes sind konkrete, messbare Indikatoren, die interpretiert werden können, um Aufschluss auf die Limitationen der konstruktiven Freiheitsgrade zu erhalten und damit den Neuheitsgrad und anschließend den Innovationsgrad zu analysieren.

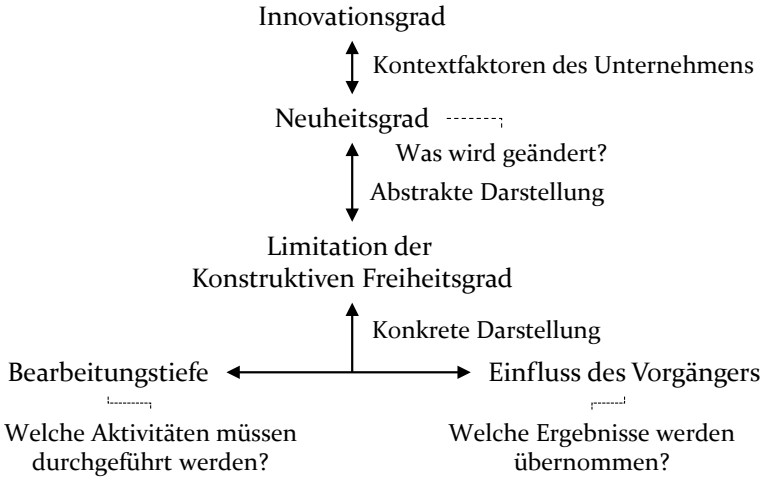


Bild 9: Darstellung der Abhängigkeiten der Einflussfaktoren untereinander

Folgende Gemeinsamkeiten lassen sich festhalten. Konstruktionsarten dienen der Klassifikation von Konstruktionsprozessen. Aus der wirtschaftlichen Perspektive dienen sie vorrangig der Risikoabschätzung und aus der wissenschaftlichen Perspektive erleichtern sie die Methodenauswahl. Die meisten Konstruktionsmethodiken unterteilen in Neu-/ Anpassungs-/ Variantenkonstruktion (/-Entwicklung). Die unterschiedlichen Definitionen beschreiben verschiedene Restriktionen, welche Konstruktionsarten voneinander abgrenzen und sich weitestgehend ineinander übertragen lassen. Für die Einordnung in eine Konstruktionsart ist eine individuelle Betrachtung des Produktentwicklungsprozesses notwendig. Aktivitäten und deren Ergebnisse können zur Einordnung herangezogen werden. Da in dieser Arbeit kein realer Produktentwicklungsprozess dienen kann, wird im Folgenden das generische Vorgehen der nach PAHL & BEITZ zu Abgrenzung der Anpassungskonstruktion verwendet [10]. Zusätzlich findet sich die Modellvorstellung des Produktentwicklungsprozess der VDI 2221-1 in dem Modell der Anpassungskonstruktion wieder [29]

2.2.3 Die Abgrenzung der Anpassungskonstruktion

Das von PAHL & BEITZ in [10] beschriebene allgemeine Modell der Produktentwicklung geht von einem systematischen Ablauf der Produktentwicklung aus, bei dem die Entwicklungstätigkeit in logische Phasen gegliedert wird, die sich durch Ziele, Aktivitäten, Reifegrad und Umfang unterscheiden. Dort wird die Anpassungskonstruktion als Konstruktionsart mit fester Prinzipieller Lösung beschrieben. Synonym dazu werden in anderen Quellen die Begriffe Musterentwurf, Qualitative Eigenschaften in [61] und in [9], Konzept in [9], Wirkkonzept in [46] verwendet. Folgende Arbeitsergebnisse werden von Vorgänger übernommen: Lösungsprinzipien (synonym Wirkprinzipien), Modulare Struktur (Module und deren Schnittstellen), Kinematische Bedingungen (Wirkflächen, -Paarungen, bspw. Getriebeschemata), Geometrische Bedingungen (Anschlussmaße, Bauraum, Hauptabmessungen) und Weitere: (Bauweisen, Fertigungsverfahren) [9, 61]. Die Bearbeitungstiefe geht nicht über das Entwerfen hinaus. Es verbleiben folgende Aktivitäten in der Anpassungskonstruktion: Gestaltungselemente detaillieren (Form, Oberflächen, Lage und Abmessungen festlegen, Halbzeuge, Rohteile, Werkstoffe und Hilfsstoffe festlegen, fertigungs-,montage-, recyclinggerechtes Gestalten), Gestaltungselemente vorausberechnen (Berechnen, Simulieren, Optimieren), Gestaltungselemente variieren und Gestaltungselemente auswählen [9, 61].

In der abstrakteren Darstellung der konstruktiven Freiheitsgrade ist die Achse Abstrahieren – Konkretisieren ab der Prinzipebene eingeschränkt (siehe Bild 10). Dies führt in den weiteren Achsen (Zerlegen – Zusammenführen und Variieren – Einschränken) ebenfalls zu Einschränkungen, da sich in der Gestaltebene nur von einer Prinzipiellen Lösung aus bewegt werden kann. Diese können nicht verallgemeinert werden, sondern ergeben sich aus der individuellen Anpassungskonstruktion. Diese spiegeln den Neuheitsgrad und im Unternehmenskontext den Innovationsgrad wieder.

2.2.4 Zusammenfassung der Anpassungskonstruktion

Die Anpassungskonstruktion unterscheidet sich von der Neukonstruktion durch eine vorgegebene Prinzipielle Lösung. Dabei ist immer von der Bauteilebene auszugehen, weil die Anpassungskonstruktion einer Baugruppe die Neukonstruktion einer Komponente umfassen kann. Eine Neukonstruktion kann in höherem Maß kreative Problemlösungsschritt enthalten, welche nicht Teil dieser Arbeit sind. Die Prinzipielle Lösung besteht aus den Lösungsprinzipien und deren Relationen als modulare Struktur. Von diesem Meilenstein aus werden die Aktivitäten des Gestaltens durchgeführt. Diese Aktivitäten unterteilen sich auf einen hohen Abstraktionsgrad in Synthese

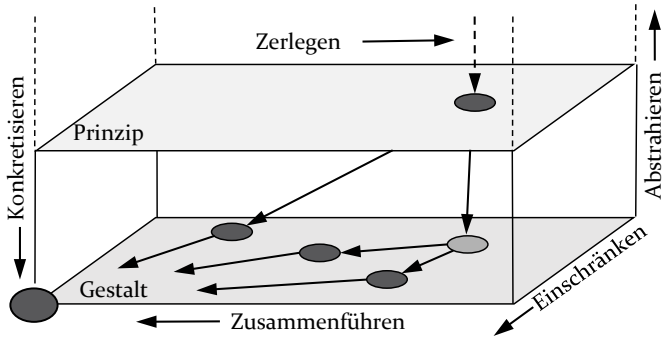


Bild 10: Ausschnitt des Lösungsraums der Anpassungskonstruktion durch Festhalten der Prinzipiellen Lösung nach RUDE [57]

und Analyse. Aus geometrischer Sicht werden bei dieser Synthese Form-, Oberflächen-, Lagetoleranzen und Abmessungen variiert. Es handelt sich folglich um quantitative Anpassungen der Merkmale aufgrund von neuen Anforderungen. Die Analyse schätzt die Eigenschaften, welche sich aus den Merkmalsausprägungen ergeben ab. Sie werden mit den neuen Anforderungen gegenübergestellt. Dabei werden kinematische, geometrische und materialspezifische Restriktionen, welche sich aus den Lösungsprinzipien und deren Struktur ergeben berücksichtigt. Diese Gestaltungsmöglichkeiten werden im Folgenden als Stellgrößen der Anpassungskonstruktion bezeichnet. Sie bilden die Grundlage für das Modell der Anpassungskonstruktion, welches in Unterabschnitt 2.3.1 erarbeitet wird, und für dessen Automatisierung in Abschnitt 5.1. Auf Basis dieser Einschränkungen werden im Folgenden die Automatisierungsmöglichkeiten für eine Anpassung der Merkmalsausprägungen zusammengetragen.

2.3 Die Automatisierung des Konstruktionsprozesses

Die Automatisierung beschreibt ein breites Spektrum von Technologien, die menschliche Eingriffe in Prozesse reduzieren. Menschliche Eingriffe werden reduziert, indem Entscheidungskriterien, Prozessbeziehungen und damit verbundene Aktivitäten im Voraus festgelegt werden und in Maschinen realisiert werden [30]. Im Kontext der Produktentwicklung beschreibt die Automatisierung die Übergabe von Aktivitäten aus dem Konstruktionsprozess vom Menschen an die Maschine [3, 15]. Das Streben der Ingenieurswissenschaften, Produkte so effizient wie möglich zu entwickeln und zu produzieren, hat die Automatisierung zu einem ständigen Begleiter der Produktentwicklung gemacht. Daher ist Automatisierung immer im Kontext der momentanen Möglichkeiten und der zukünftig gewünschten oder geforderten Veränderungen zu betrachten. Derzeit befindet sich die Produktentwicklung auf dem

Weg von der Rechnerunterstützen Konstruktion zur virtuellen Produktentwicklung [62, 63]. Der Grad der Digitalisierung[1] wird dabei kontinuierlich erhöht [13]. Folglich nehmen aus technologischer Sicht die Informationstechnologien, die digitale Vernetzung und das ganzheitliche Systemdenken eine zentrale Rolle ein. Das sich hieraus ergebende Forschungsgebiet wird im Englischen als *Design Automation* [65] und im Hinblick auf das Gestalten von Produktmerkmalen auch *Computational Design Synthesis* (CDS) bezeichnet [66, 67]. Im Fokus stehen Methoden und Technologien des Maschinellen Lernens, der Optimierung (engl. *Optimization*), des Wissensbasierten Konstruierens und der rechnerunterstützten Produktentwicklung (engl. *Computer-Aided Engineering*). Deren Umsetzung als eigenständige Software oder als integriertes Softwaremodul wird als *Design Automation System* bezeichnet [28].

Dieses Kapitel charakterisiert zunächst den Begriff Automatisierung im Unterabschnitt 2.3.1. Anschließend wird in Unterabschnitt 2.3.2 auf die Limitationen der Automatisierung durch digitale Technologien eingegangen. Sie beschreiben den Rahmen der betrachteten Technologien aus denen in Unterabschnitt 2.3.3 die Forschungslandschaft für die Automatisierung der Anpassungskonstruktion herausgearbeitet wird. Unterabschnitt 2.3.3 betrachtet dabei Technologien der rechnerunterstützten Produktentwicklung, der Optimierung, des Wissensbasierten Konstruierens sowie des Maschinellen Lernens. Dabei kann auf Grund der Vielseitigkeit der Technologien nur auf wissenschaftliche Grundlagen, die für die Ausarbeitung dieser Arbeit relevant sind, eingegangen werden. An den entsprechenden Stellen wird daher an weiterführende Literatur verwiesen. Ziel von Unterabschnitt 2.3.3 ist, die Automatisierungsmöglichkeiten der einzelnen Technologien darzustellen.

2.3.1 Die Charakterisierungsmöglichkeiten der Automatisierung

Im Kontext der Produktentwicklung wird die Automatisierung unterschiedlich aufgefasst. Einerseits beinhaltet der Konstruktionsprozess kreative Tätigkeiten, welche nicht an die Maschine übergeben werden können (dies wird in Unterabschnitt 2.3.2 aufgegriffen und diskutiert), und führt folglich häufig zur Aussage, dass dieser nicht automatisiert werden kann. Andererseits streben die Ingenieurswissenschaften seit Beginn nach Effizienz durch den Einsatz von Hilfsmitteln, wie beispielsweise zuverlässige und schnelle Berechnung mathematischer Gleichungen durch den Computer. Folglich sind Teile des Konstruktionsprozesses bereits automatisiert. Mit dieser Gratwanderung

[1] Grad der Digitalisierung bezeichnet den unternehmensspezifische Einsatz von Informations- und Kommunikationstechnologien und deren Vernetzung [64]

hat sich in den vergangen Jahren eine Vielzahl von Forschenden auseinandergesetzt. Deren Überlegungen sollen im Folgenden zusammengefasst werden, um ein besseres Verständnis des Begriffes Automatisierung zu ermöglichen.

Die Automatisierung führt zwangsläufig an die Schnittstelle zwischen Mensch und Maschine und die Frage welcher Anteil vom Menschen und welcher Anteil von der Maschine getragen wird. Diese Mensch-Maschine Interaktion kann durch verschiedene „Automatisierungslevel" (engl. *Level of Automation* (LoA)) ausgedrückt werden. Jedes dieser Level definiert einen unterschiedlichen Grad der Automatisierung einer Aktivität. Diese bedeutet, dass die Automatisierung binär ist, sondern über ein Kontinuum, zwischen vollständig manuell und vollständig automatisiert (autonom) ist [68]. Mehrere verschiedene Möglichkeiten diskrete Level in diesem Kontinuum zu definieren wurden von verschiedenen Forschenden vorgeschlagen. Dies hat zu zahlreichen Taxonomien für die Interaktion zwischen Menschen und Maschinen geführt, welche durch VAGIA et al. [68] untersucht und gegenübergestellt wurden. Das Ergebnis dieser Analyse ist in Tabelle 1 dargestellt. Dort enthält die linke Spalte der Tabelle alle möglichen LoA, die von VAGIA et al. gefunden wurden. Jeder Tabellenzeile wurde ein Codename zugeteilt, welcher den Grad der Automatisierung beschreibt. Jede Spalte entspricht einer Taxonomien für die Interaktion zwischen Menschen und Maschinen.

Aus Tabelle 1 folgern VAGIA et al. [68], dass es keine einheitliche Unterteilung gibt, dass es Level gibt die häufiger gebraucht werden als andere und, dass die Level vom jeweiligen Einsatzgebiet der Taxonomie abhängig sind. Außerdem unterstreichen die Autoren, dass es kein richtig oder falsch bei der Auswahl der Taxonomie gibt. Tabelle 1 wurde in Graustufen hinterlegt, um gemeinsame Stufen leichter zu identifizieren. Dabei fällt auf, dass drei Level auf dem Weg zur vollständigen Automatisierung („Autonom") bei den meisten Taxonomien berücksichtigt werden (**hervorgehoben** in Tabelle 1). Dabei wurden die Level, die sich auf die Steuerung beziehen, zurückgestellt. Diese stammen vorrangig aus der Luft- und Raumfahrt und Robotik. Sie nehmen für die Aktivitäten der Anpassungskonstruktion bei denen Produktentwickelnde meist noch direkt in der Produktentwicklungsumgebung an einem Desktop Computer arbeiten eine untergeordnete Rolle ein. Das erste dieser signifikanten Level ist erreicht, wenn die Maschine Vorschläge generieren kann. Über die nächsten Zwischenlevel werden die Vorschläge immer weiter durch die Maschine eingeschränkt, bis eine Option den Anwendende vorgeschlagen wird und diese nur zustimmen müssen. Die dritte signifikante Stufe ist erreicht, wenn die Tätigkeit autonom abläuft und nur unterbrochen wird, wenn die Anwendende eingreifen. Diese Überlegung legt offen, dass Tätigkeiten im Kontext der Automatisierung aus einer Handlung, beispielsweise

Tabelle 1: Übersicht der Taxonomien der Automatisierung in Anlehnung an [68]

	Sheridan & Verplanck [69, 70]	**Endsley** [71]	Nituen & Park [72, 73]	Riley [74]	Milligram [75]	Endsley & Kiris [76]	Draper [77]	Endsley & Kaber [78]	Lorenz et al. [79]	Clough [80, 81]	Proud et al. [82]	Fereidunian et al. [83, 84]
Vollständig manuell	1	0	1	1	1	1	1	1	1	1	1	0
Datenakquise	1	0	1	2	1	1	1	1	1	1	2	1
Telepräsenz	1	0	1	2	2	1	1	1	1	1	2	1
Intelligenter Assistent	1	0	1	2	2	1	2	1	1	1	2	1
Fernsteuerbar	1	0	1	2	2	1	2	1	1	2	2	1
Vorschläge	2	1	2	3	2	2	2	2	2	2	3	2
Eingeschränkte Vorschläge	3	1	2	3	2	2	2	7	2	2	3	3
Direkte Steuerung	3	1	2	6	3	2	2	3	2	2	3	3
Gemeinsame Steuerung	3	1	2	6	3	2	3	4	2		3	3
Vorgeschlagene Alternativen	4	1	2	6	3	2	3	4	2	2	3	4
Ausführung nach Zustimmung	5	2	3	8	3	3	3	6	2	2	4	5
Entscheidungsunterstützung	5	2	3	8	3	3	3	5	2	3	4	5
Ausführung bis Veto	6	3	4	8	3	4	3	5	3	3	5	6
Ausführung und Benachrichtigung	7	3	4	9	3	4	3	5	3	3	6	7
Partner	7	3	4	10	3	4	4	5	3	3	6	7
Benachrichtigung nur auf Anfrage	8	3	4	10	3	4	4	5	3	3	7	8
Benachrichtigung bei Entscheidung	9	3	4	10	3	4	4	8	3	3	7	9
Überwacht	9	3	4	11	4	4	5	9	3	3	7	9
Autonom	10	4	5	12	5	5	5	10	3	4	8	10

das Erstellen von Vorschlägen, und einer Entscheidung, beispielsweise das Auswählen eines Vorschlages, bestehen.

Für diese Arbeit wird eine Taxonomie in Anlehnung an ENDSLEY [71] verwendet, weil für die Konstruktion untergeordnete Technologien, wie beispielsweise die Steuerung eine untergeordnete Rolle spielen und die Levelbezeichnungen mit den Vorstellungen der Produktentwicklung einhergehen (s. Tabelle 2). Es wurden zwei Anpassungen vorgenommen. Zum einen wurde das Level 0 für den manuellen Fall hinzugezogen. Zum anderen wird nicht das Wort KI verwendet, um die Automatisierungsgrade zu beschreiben, da nicht alle der folgend vorgestellten Automatisierungstechnologien sich der KI zu ordnen lassen. Jedoch beruhen alle Informations- und Kommunikati-

onstechnologien auf Daten. Folglich bedeutet Handeln eine Änderung der zugrundeliegenden Daten. Dem entsprechend ist das zweite Level erreicht, so bald Vorschläge von der Maschine generiert werden können. Entscheiden bedeute diese Änderung zu speichern und/oder folgende Aktionen auszuführen. Die Level 2 – 4 werden durch die Bedingungen dieser Entscheidung unterschieden. Der Unterschied zwischen Handeln und Speichern lässt sich an der Arbeitsweise eines Computers einfach verbildlichen. Während der Ausführung eines Programms, beispielsweise einer Berechnung, sind die Daten im Arbeitsspeicher hinterlegt. Sie sind nicht persistent. Ein Absturz oder Abbruch hätte zur Folge, dass der Fortschritt verloren ist. Erst wenn die Änderungen gespeichert werden, werden die Daten auf der Festplatte des Computers hinterlegt und sind vom Programm unabhängig.

Tabelle 2: Taxonomie der Automatisierung in Anlehnung an ENDSLEY [71]

Level	Name	Beschreibung
Level 0	Manuell	Vollständig manueller Fall
Level 1	Erklärend	Mensch handelt nach Empfehlungen der Maschine
Level 2	Unterstützend	Maschine handelt nach Zustimmung des Anwenders
Level 3	Überwacht	Maschine handelt eigenständig solange kein Eingriff erfolgt
Level 4	Vollständig Automatisiert	Vollständig automatisierter Fall

Zuletzt sei angemerkt, dass eine explizite Taxonomie für die Automatisierung der Konstruktion noch nicht existiert. Dies könnte eine potentielle Forschungslücke darstellen, da sich die Konstruktion vor allem im Bezug auf zeitkritische Aktionen und Steuerung von den bisher betrachteten Domänen (Luft- und Raumfahrt, Robotik und Produktionssteuerung) stark unterscheidet.

2.3.2 Inhärente Limitationen der Automatisierung

In Unterabschnitt 2.3.1 wurde gezeigt, dass sich Automatisierung auf Handeln (eine Änderung der Daten) und Entscheiden (eine Speicherung der Änderung) zurückführen lässt. Zur Anwendung der digitalen Technologien der Automatisierung wird vorausgesetzt, dass die Daten zugänglich und deren Änderungsmöglichkeiten für die Maschine möglich sind [85]. Folglich muss eine Aktivität des Konstruktionsprozesses unter zwei Aspekten betrachtet werden – zum einen das Denken in technischen Systemen und zum anderen die Beschränkung der Aktivitäten auf Aufgaben im Konstruktionsprozess.

Das Denken in Systemen und die Modellierung

Die erste Limitation erfordert, dass für den Einsatz der Methoden und Technologien der Digitalisierung die benötigten Ressourcen und die resultierenden Arbeitsergebnisse in computerverarbeitbaren Repräsentationsformen vorliegen [85]. Daher müssen die Ressourcenauswahl, deren Relationen sowie der Weg zu den resultierenden Arbeitsergebnissen computerverarbeitbar beschrieben werden, da ohne Abstraktion und Zweckmäßigkeit keine Verarbeitung durch die Informations- und Kommunikationstechnologien möglich ist [85, 86]. Folglich beruht die erste Limitation auf der Voraussetzung, die notwendigen Informationen sowie die erforderlichen Ressourcen im Bezug auf einen zweckmäßigen Zusammenhang in Systemen darzustellen [29]. Dabei sind Systeme Mengen von Elementen und zwischen ihnen bestehende Relationen, welche einen definierten Zweck erfüllen und von einer Umgebung durch eine gedachte Systemgrenze abgegrenzt werden [87]. Das System steht mit der Umgebung durch Ein- und Ausgangsgrößen in Beziehung. Die Funktion eines Systems kann durch die Ein- und Ausgangsgrößen beschrieben werden. Die Systemelemente können selbst wiederum Systeme sein, die aus Elementen und Beziehungen bestehen (s. Bild 11). [9]

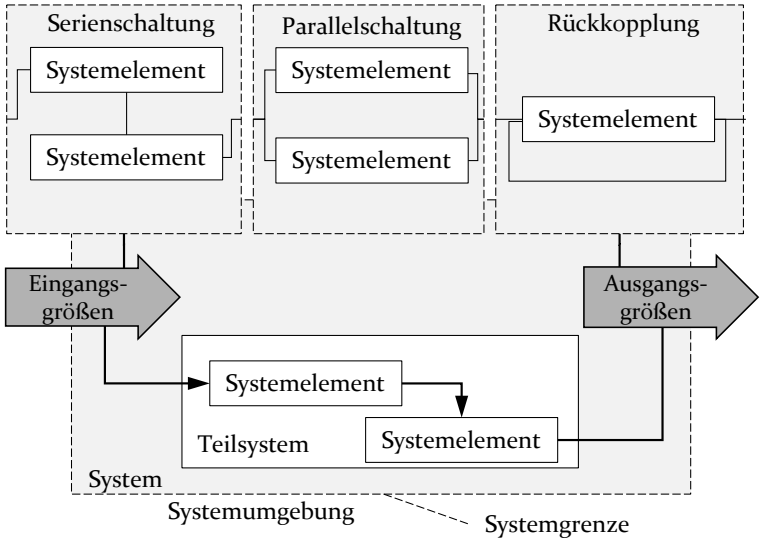


Bild 11: Darstellung der Bestandteile eins Systems und deren Kopplungsmöglichkeiten

Um Systeme für die Maschine verarbeitbar zu machen, können Modelle genutzt werden. Modelle sind mögliche Abbilder von Systemen und beschreiben diese mittels geeigneter Zeichen [88]. Diese Modelle werden Produktmodelle genannt [89]. Die Richtlinie VDI 2249 definiert das Produktmodell als

ein Modell, das alle relevanten Informationen über ein Produkt in hinreichender Vollständigkeit enthält [90]. Nach STACHOWIAK ist ein Modell immer das Abbild eines Originals (Abbildungsmerkmal) für einen bestimmten Zweck (pragmatisches Merkmal), wobei das Original jedoch nur in reduziertem oder abstrahiertem Umfang (Verkürzungsmerkmal) wiedergegeben wird [91]. Der Prozess der Modellierung ist wie der Konstruktionsprozess selbst ein komplexer, iterativer Prozess [92]. Er kann bereits selbst Teil des Problemlösungsprozesses der Konstruktion sein [28]. Kernaufgabe der Modellierung ist die zielgerichtete Abstraktion der Realität um ein Modell zu erhalten. DUFFY & ANDERSON ordnen sie in vier Phasen, welche in Bild 12 dargestellt sind [92]. Dort beruht ein Phänomenmodell auf Beobachtungen und Analysen der Realität. Durch Semantik können diese Modelle zu Informationsmodellen (engl. Information model) weiterentwickelt werden. Durch computerverarbeitbare Darstellungsformen werden sie zu Berechnungsmodellen (engl. Computer Model). Im Rahmen der Automatisierung ist es notwendig, dass die Modelle in der letzten Phase – als Berechnungsmodell vorliegen [86]. Der Weg dort hin ist abhängig von dem „zu Modellierenden" (engl. *Modellee*), dem Ziel, den Randbedingungen, und dem gewählten Modellierungsansatz (s. Bild 12). Der Begriff Modellee wurde gewählt, weil kein anderer passender Begriff gefunden wurde, der die Inhalte von Modellen zusammenfassen kann [86]. Der Modellierungsansatz kann weiter in Methoden, Werkzeuge, Hilfsmittel und Konzepte charakterisiert werden. Beispielsweise kann ein Prozess (Modellee) zur dessen Analyse (Ziel) durch UML (Methode) auf Basis der Prozessdaten (Hilfsmittel) auf dem Flipchart (Werkzeug) unter Einhaltung der Geheimhaltung (Randbedingung) in einem Workshop erarbeitet werden. Dabei ist zu beachten, dass diese Faktoren sich gegenseitig beeinflussen und unter Umständen sogar ausschließen können. Außerdem umfasst die Modellierung typischerweise Aktivitäten, die Erfahrung bei der Auswahl des geeigneten Modellierungsansatzes sowie bei der Interpretation der Ergebnisse und Kreativität bei der Erstellung des Modells beinhalten [28]. Folglich kann die Modellierung nur durch rechnerunterstützende Hilfsmittel begleitet aber derzeit noch nicht vollständig automatisiert werden.

Der Unterscheidung zwischen Aufgaben und Problemen

Die zweite Limitation bezieht sich auf die Abgrenzung zwischen Problemen und Aufgaben innerhalb des Konstruktionsprozesses nach [46]. Ein Problem wird dadurch charakterisiert, dass bei der Transformation von einem unerwünschten Anfangszustand in einen gewünschten Endzustand eine Barriere existiert. Folglich sind die Transformationsregeln und/oder die benötigten Hilfsmittel (teilweise) unbekannt [93]. Die Transformationsregeln beschreiben den Weg vom Anfangs- zum Endzu-

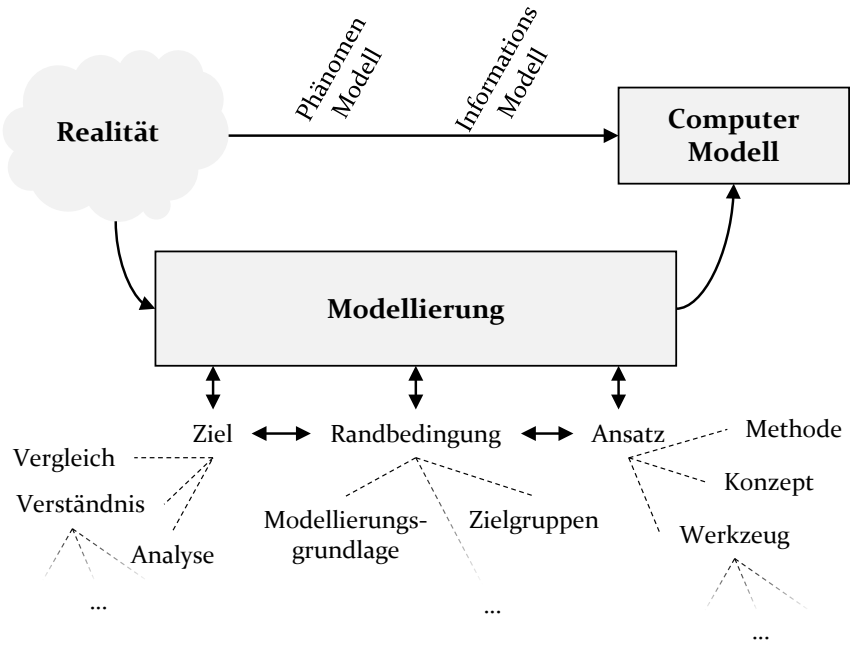


Bild 12: Darstellung des Prozesses der Modellierung nach [92] und der Charakterisierungsmöglichkeiten für Modellierungsprozesse nach [86]

stand anhand von Modellen. Eine automatische Überwindung von gänzlich unbekannten Transformationsregeln ist nicht möglich, da selbst die Techniken der Künstlichen Intelligenz auf der Verfügbarkeit von Wissen in expliziter Form basieren [31]. Mit entsprechendem Hintergrundwissen ist es möglich, die Auswahl der Transformationsregeln flexibel zu gestalten. Dies setzt voraus, dass sich sie die verwendeten Modelle dynamisch anpassen lassen oder ein übergeordnetes Modell entscheidet, welche Transformationsregeln die größten Erfolgswahrscheinlichkeiten hat. Ersteres beschreibt den Einsatz von dynamischer Programmierung [94]. Zweiteres beschreibt ein Modell aus Modellen [12].

Jedoch müssen für beide Möglichkeiten die Modelle der einzelnen Transformationsregeln vorab bekannt sein. Die Hilfsmittel beziehen sich einerseits auf die Operatoren, die beschreiben wie der Zustand verändert wird, und andererseits auf die benötigten Daten, Informationen und/oder Wissen, welche die Zustände, deren Relationen und Operatoren beschreiben. Die benötigten Ressourcen sind Daten, Informationen und/oder Wissen. Für eine Automatisierung müssen die Mittel verarbeitbar für den Computer vorliegen [31, 95, 96]. Dies schließt unbekannte Mittel aus und führt zu dem Schluss, dass nur Aufgaben des Konstruktionsprozesses automatisierbar sind. Nur

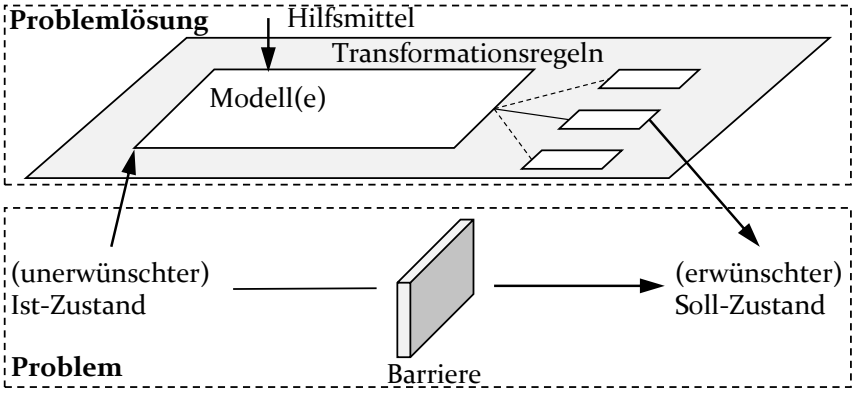


Bild 13: Charakterisierung des Problems durch eine Barriere zwischen Ist- und Sollzustand sowie Darstellung der Problemlösung mit zugehörigen Transformationsregeln und Hilfsmitteln in Anlehnung an die VDI 2221-1 [29]

wenn eine Aktivität als computerverarbeitbar beschrieben werden kann und keine kreativen Prozesse notwendig sind, kann sie automatisiert werden.

Im Kontext der Automatisierung der Anpassungskonstruktion sind die Merkmale und deren Ausprägung das zentrale Element der Modellierung. Diese werden in Produktmodellen dargestellt und über rechnerunterstützte Systeme sowohl für den Mensch als auch die Maschine zugänglich gemacht.

2.3.3 Die Forschungslandschaft der Automatisierten Anpassungskonstruktion

In den folgenden Abschnitten werden die Facetten der Automatisierung im Kontext der Anpassungskonstruktion vorgestellt, um die derzeitigen Möglichkeiten darzustellen und potentielle Forschungslücken zu identifizieren. Da selten ein expliziter Bezug zur Anpassungskonstruktion gegeben ist, werden Methoden und Technologien betrachtet, welche Aktivitäten des Konstruktionsprozesses mit Hilfe von Informations- und Kommunikationstechnologien automatisieren und dabei Merkmale und Merkmalsausprägungen als Eingangsgrößen heranziehen (Analyse) oder als Ergebnis beschreiben (Synthese). Diese werden unter dem Begriff *Design Automation* verortet [65]. Es werden Technologien, welche mindestens dem zweiten Level der Taxonomie von ENDSLEY entsprechen, berücksichtigt (s. Tabelle 2 auf Seite 22) [71]. Dementsprechend werden mindestens Vorschläge für die Gestalt des Produkt durch den Computer generiert. Auf Basis der zentralen Technologien und Methoden der Digitalisierung wurden deren Schnittmengen sowie die Einordnung der Automatisierung der Anpassungskonstruktion

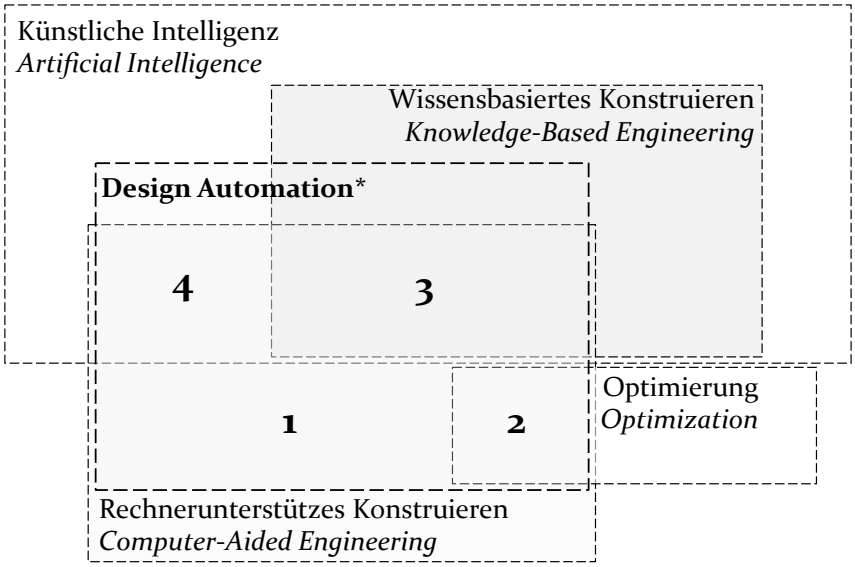


Bild 14: Darstellung der Forschungslandschaft der Automatisierung des Konstruktionsprozesses durch: 1 Rechnerunterstützte Systeme, 2 Optimierungsalgorithmen, 3 Wissensbasierte Systeme und 4 Maschinelles Lernen als Resultat der analysierten Literatur und den Erkenntnissen aus [P1]

erarbeitet und in Bild 14 als Forschungslandschaft dargestellt. Die vier gefundenen Cluster werden anschließend vorgestellt. Inhalt diese Arbeit ist ein rein digitaler Produktentwicklungsprozess, daher werden die physikalische Automatisierung – beispielsweise durch Roboter in der Produktentstehung, oder physische Interaktion innerhalb des Produktentwicklungsprozesses – beispielsweise durch Eyetracking in der Konstruktionsumgebung – nicht berücksichtigt. Hierfür wird auf weiterführende Literatur für Robotik [97] und Interaktion [98] verwiesen. Außerdem muss beachtet werden, dass der Begriff Prozessautomatisierung bereits stark mit der Automatisierung von technischen Produkten und Anlangen verknüpft und damit belegt ist [28]. Mit der Prozessautomatisierung überschneiden sich zwar viele Methoden wie beispielsweise das Denken in ganzheitlichen Systemen, jedoch sind die Technologien von der Hardware getrieben.

Unabhängig von den genanten Methoden und Technologien bezieht sich die Automatisierung des Konstruktionsprozesses in der Literatur fast ausschließlich auf Routinetätigkeiten, da diese einerseits als sehr zeit- und kostenintensiv gelten und anderseits eher selten unerwartete Probleme beinhalten [99–101]. Diese Einschränkung ist bei individueller Betrachtung von Routinetätigkeiten sinnvoll jedoch subjektiv, da sie unter anderem vom Kontext der technologischen Möglichkeiten und Vorwissen abhängig ist [97]. Da sich alle Technologien der Automatisierung des Produktentwicklungspro-

zess auf computerverarbeitbare Algorithmen beziehen, können die zwei in Unterabschnitt 2.3.2 vorgestellten, objektiven Gesichtspunkte, die Aktivitäten erfüllen müssen als Limitationen für die Automatisierung herangezogen werden.

2.3.4 Die Rechnerunterstützen Systeme

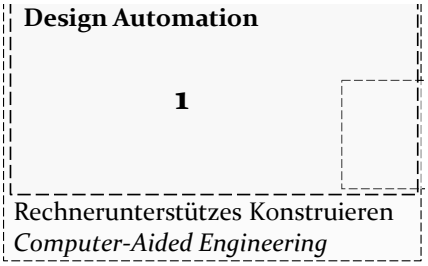


Bild 15: Ausschnitt der Forschungslandschaft für die Rechnerunterstützen Systeme

CAx-Systeme ist die zusammenfassende Bezeichnung aller Systeme der Rechnerunterstützung in produzierenden Unternehmen. Das CA steht für den Englischen Begriff Computer-Aided. Übersetzt mit rechnerunterstützt unterstreicht er die Tatsache, dass der Mensch nach wie vor die Prozesse führt und Rechnersysteme ihm lediglich Hilfestellung für besseres Entscheiden und Problemlösen anbieten. Das x als Platzhalter für eine Vielzahl von Akronymen, die bestimmte Einsatzbereiche näher spezifizieren. Rechnerunterstütze Systeme spiegeln die aktuellen Bedürfnisse und Herausforderungen der Unternehmen wider. Im Kontext zunehmend globalisierter Märkte müssen Unternehmen immer kürzere Innovationszyklen und höheren Variantenreichtum bei zeitgleicher gesteigerter Produktkomplexität realisieren, um ihre Wettbewerbsfähigkeit aufrecht zu erhalten oder zu steigern. Diesen Herausforderungen wird durch immer enger vernetzte Unternehmensbereiche, in denen feingranular zerlegt und damit parallelisierbare Aufgaben in global verteilten Teams bearbeitet werden, begegnet. Auf der Seite der CAx-Systeme bedeuten diese organisatorischen Maßnahmen eine gesteigerte Vernetzung oder sogar Verschmelzung der einzelnen Informations- und Kommunikationstechnologien. Dadurch soll erreicht werden, dass für alle Aktivitäten ein digitales Abbild den aktuellen Zustand dieses Produktes in einem Produktmodell widerspiegelt. [28]

Hinsichtlich der Automatisierung erfüllen CAx-Systeme zwei Aufgaben. Einerseits werden Sie als Werkzeug verstanden, welches das Produktmodell für den Menschen darstellen kann. Andererseits stellen Sie selbst Technologien für die Automatisierung der Konstruktion bereit. Diese zwei Perspektiven werden anschließend beleuchtet.

Rechnerunterstütze Systeme als Werkzeug für die Automatisierung

In der Forschungslandschaft (s. Bild 14) überschneiden sich die Automatisierungsmöglichkeiten der Anpassungskonstruktion fast vollständig mit den rechnerunterstützten Systemen. Aus dieser Beobachtung kann geschlossen werden, dass ein Großteil der Automatisierungstechnologien, die im Folgenden vorgestellt werden, eine CAx-Integration nutzt (2 – 4 in Bild 14) oder selbst eine CAx-Technologie (1 in Bild 14) ist. Für die übrigen Automatisierungstechnologien ist eine CAx-Integration theoretisch möglich, da sie unter der Prämisse ausgewählt wurden, dass sie sich auf die Merkmale und deren Ausprägungen beziehen müssen. Dementsprechend sind die Randbereiche in Bild 14 marginal klein gezeichnet. Wäre dies nicht der Fall, müssten alle folgenden Technologien jeweils mit und ohne CAx-Integration vorgestellt werden, obwohl eine Übertragung möglich ist. Die Beobachtung, dass eine CAx-Integration sinnvoll ist, lässt sich aus zwei Perspektiven begründen, welche in Bild 16 dargestellt sind.

Zum einen werden die existierenden Produktmodelle genutzt. Vor allem im Hinblick auf die Geometrie bieten die etablierten Modelle wiederum zwei Vorteile. Einerseits müssen grundsätzliche geometrische Beziehungen nicht eigens entwickelt und aufwändig modelliert werden, sondern sind bereits vorhanden. Dies erspart einerseits den Modellierungsaufwand und beugt andererseits Modellierungsfehler vor. Im Vergleich zu anderen Modellen sind Standards für geometrische Produktmodelle vergleichsweise weit verbreitet. Die Standardisierung erfolgt über definierte Datenformate und deren Schnittstellen. Ein Datenformat beschreibt eine definierte Syntax und Semantik für das digitale Abspeichern von Informationen [102]. Eine Schnittstelle ist ein Gebilde von Regeln, Vereinbarungen und Bedingungen, die dem Informationsaustausch zwischen zwei oder mehreren miteinander kommunizierenden Systemen oder Systemkomponenten dient [28]. Etablierte Datenformate sind unter anderem STEP (STandard for the Exchange of Product model data), JT (Jupiter Tessellation), XML (Extensible Markup Language)) und IGES (Initial Graphics Exchange Specification). Für die Automatisierung hinsichtlich der Merkmale ist wichtig die Modellierungsart, welche der Kernel des CAx-Systems beschreibt, zu kennen, da sie die Schnittstelle zu den Geometriedaten darstellt. Aus ihr leitet sich ab, welche Daten direkt abgefragt werden können und welche zusätzlich modelliert werden müssen. Relevante 3D-Modellarten sind Kanten-(Drahtmodelle, *Wireframes*), Flachen-(*Surface Models*, *Sheet Bodies*), Volumen-(*Solids*) oder Voxelmodelle [103]. Volumenmodelle werden nach ihrer Erzeugungsvorschrift in generativen (prozeduralen, bspw. CSG-Modelle) oder akkumulativen (deskriptiven, bspw. B-Rep-Modelle) Geometriemodellen unterteilt. Neben den Informationen über die geometrischen Artefakte bieten CAD-Systeme weitere Informationsquellen wie die Model-

lierungschronologie und die Erzeugnisstruktur in Form von Produktgraphen, welcher die Eltern-Kind-Beziehungen von geometrischen Artefakten angibt, an [103]. Ersteres beschreibt die Konstruktionshistorie und zweiteres die Relationen zwischen geometrischen Artefakten. Außerdem wird das geometrische Modell als Informationscontainer genutzt, weil sich hier geometrische Informationen leicht verständlich mit weiterer Semantik anreichern lassen. Einerseits bietet die nachfolgend vorgestellte Feature-Technologie die Möglichkeit an semantische Informationen über die Geometrie hinaus im CAx-System zu modellieren. Andererseits zeigen zahlreiche Veröffentlichungen die Möglichkeit des Annotiertes von semantischen Informationen auf geometrische Artefakte [104–107].

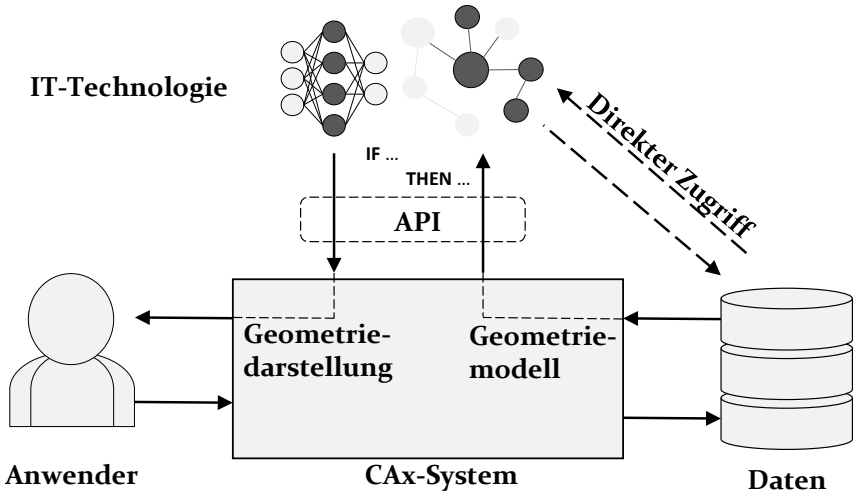


Bild 16: Darstellung der CAx-Integration die es Informations- und Kommunikationstechnologien erlaubt das Geometriemodell und dessen Darstellung nutzen können ohne sie selbst realisieren zu müssen

Der zweite Grund neben den existierenden Modellen ist der Vorteil der grafischen Darstellung von Produktmodellen. CAx-Modelle bieten die Möglichkeit eines Front-Ends der Automatisierungstechnologien für den menschlichen Anwender. Zwar ist es möglich, die Eigenschaften eines Produktes in Kennzahlen darzustellen, jedoch können Experten auf Basis der Darstellungen im CAx-System rein visuell Ergebnisse auf Plausibilität oder Fehler prüfen. Beispielsweise das Auftreten von Spannungsüberhöhungen bei geometrischen Einkerbungen im Kraftfluss oder die Abstände oder sogar Überschneidungen in einer Mehrkörpersimulation. Die grafische Darstellung ist nicht nur bei der Anwendung der Automatisierungstechnologien relevant, sondern auch für deren Entwicklung. Folglich ist das Produktmodell in der

CAx-Umgebung ein wichtiges Werkzeug für die Fehlersuche bei der Entwicklung von Automatisierungstechnologien. Solange die Konstruktion ein Hybrid aus menschlichen und computergesteuerten Aktivitäten ist, sollte das Produktmodell sich an den menschlichen Aktivitäten orientieren, da er die kreativen Aufgaben lösen muss. Darüber hinaus können Automatisierungstechnologien so in die gewohnte Arbeitsumgebung integriert werden, so dass die Einstiegshürde reduziert wird.

Rechnerunterstütze Systeme als Technologie für die Automatisierung

CAx-Systeme sind nicht nur ein wichtiges Werkzeug der Automatisierung, sondern stellen selbst auch Technologien für die Automatisierung zur Verfügung. Diese sind in Schnittmenge 1 in Bild 14 verortet. Hintergrund dieser Technologien ist die Wiederverwendung von möglichst großen Teilen eines bestehenden Produktes, um diesen Teil nicht erneut modellieren zu müssen. In diese Kategorie zählen Makros, Konstruktionsschablonen und -Kataloge. Während Makros und Konstruktionsschablonen nur für spezifische Produkte, -Variationen oder Baukästen verwendet werden können, bieten Konstruktionskataloge wiederverwendbare Lösungen für ein breiteres Spektrum an. Konstruktionskataloge fassen wiederkehrende Lösungen nach einheitlichen Gesichtspunkten zusammen und machen sie nach spezifischen Gesichtspunkten zugreifbar [41]. Sinnvoll ist eine Katalogerstellung, wenn die Inhalte immer wieder nötig sind und wenn es gelingt, komplizierte konstruktive Zusammenhänge in einfache und überschaubare Bruchstücke zu zerlegen [41]. Hinsichtlich der Merkmale und deren Ausprägungen sind Material- und Normteilbibliotheken die bekanntesten Beispiele dieser Kategorie. Das Konzept des methodischen Zerlegens, Erkennens und Wiederverwendens wurde in den vergangenen Jahren konsequent weiter entwickelt [108]. Dabei ist eine zentrale Frage, wie Produkte modelliert werden müssen, um eine Wiederverwendung zu erleichtern. Im Bezug auf geometrische Merkmale wird Forschung in diese Richtung unter *Common Design Structure Discovery* (CDSD) verortet.

CDSD ist die logische Weiterentwicklung der Norm- und Bauteil-Kataloge, weil es den selben Zweck erfüllt. In beiden Fällen wird der Gedanke verfolgt, ein Produkt oder Teile davon nicht neu zu modellieren, sondern ein bereits existierendes Modell zu verwenden und damit Modellierungsaufwand drastisch zu senken [109]. Da moderne Produkte und deren Bauteile immer differenzierteren Anforderungen genügen müssen, ist es oft nicht möglich vollständige Bauteile zu übernehmen, sondern nur Teile. Damit diese Teilmodelle so groß wie möglich sind, beschäftigt sich CDSD zuerst mit der Modellierung zusätzlicher Bauteilinformation, welche anschließend das Auffinden ähnlicher Modelle erleichtern. Zur Repräsentation zusätzlicher

Informationen haben sich Graphen durchgesetzt. Sie lassen sich leicht aus den vorhanden Produktmodellen ableiten, indem geometrische Artefakte als Knoten verwendet werden. Dennoch sind sie sehr flexibel, weil sie die Mög-

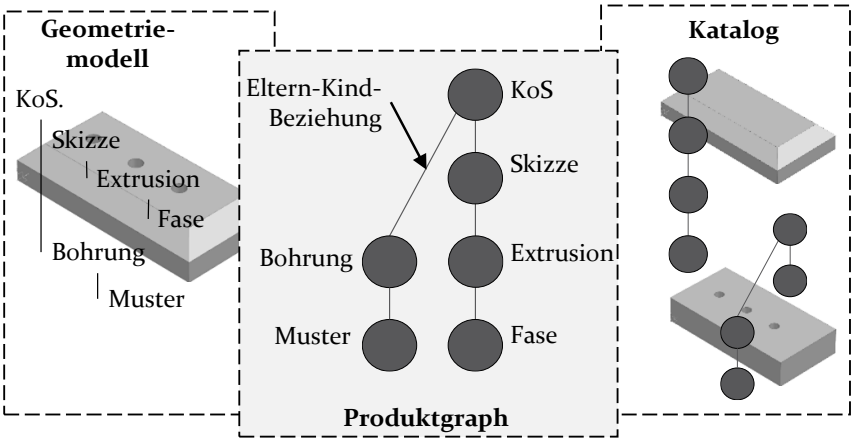


Bild 17: Schematische Darstellung eines graphen-basierten CDSD Prozesses, welcher das Geomtriemodell in einen Graphen überführt, um bereits vorhandene Geometriemodelle aus einem Konstruktionskatalog vorzuschlagen oder selbständig auszuwählen

Hierfür setzt CDSD direkt am Kernel des CAD-Systems an, um den vorhandenen Produktgraphen zu nutzen, zu erweitern oder einen Eigenen zu erstellen. Je nach Ziel werden sogenannte Reeb Graphs, Shock Graphs oder Feature Graphs verwendet [110]. Reeb Graphen sind Skelett-Modelle deren Knoten kollabierte Oberfläche des Modells darstellen [111–113]. Bei Shock Graphen wird der Skelettierungsprozess in der Regel durch Extraktion der Mittelachse realisiert [114, 115]. Diese Graphen sind geeignet für Strukturdarstellung. Jedoch hindert der Zeitaufwand für die Erstellung der Graphen und ihr abstrakter Charakter konkrete Repräsentationen gängiger Bauteile [110]. Konkrete Repräsentationen können mit Hilfe von geometrischen Features im Nachhinein wieder an das Skelett modelliert werden [116, 117]. Feature Graphen modellieren die Postion von geometrischen Features im Raum oder andere Beziehungen zwischen ihnen [118–120]. Beispiele für andere Beziehungen sind Wiederholungen oder Muster von Features und geschlossener Gruppen mit engen Relationen untereinander aber nur wenigen Verknüpfungen zu anderen Features [109]. Das Beispiel in Bild 17 zeigt einen Graphen, der die Eltern-Kind Beziehung aus dem geometrischen Modell – genauer dem Modellbaum – ableitet und damit die einzelnen geometrischen Features in Relation bringt. Diese Graphen werden mit Hilfe mathematischen

oder statistischer Algorithmen zerlegt um zu analysieren, welche Muster möglichst häufig in verschiedenen Bauteilen auftreten. Besonders relevant gelten Muster, die besonders häufig auftreten, oder besonders signifikant sind [109]. Gespeichert in Konstruktionskatalogen, können diese Muster für folgende Konstruktionen wiederverwendet werden. Außerdem kann die graphen-basierte Darstellung des Produktmodells und das dort verankerte Erkennen repetitiver Zusammenhänge eine Grundlage für Produktontologien sein [109].

2.3.5 Die Strukturoptimierung

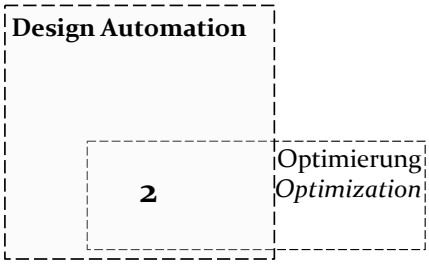


Bild 18: Ausschnitt der Forschungslandschaft für die Strukturoptimierung

Seit 1960 werden die verschiedenen numerischen Simulationen der Absicherung von virtuellen Produkten mit deterministischen und stochastischen Algorithmen gekoppelt, um geometrische Merkmale zu optimieren [121, 122]. Produktentwickelnde werden bei der iterativen Ausarbeitung und Absicherung entlastet. Die Optimierungsprobleme werden in Parameter- (engl. *Sizing*), Struktur- (engl. *Shape*) und Topologieoptimierung (engl. *Topology Optimization*) unterschieden [123]. Das Ziel der Parameteroptimierung ist die Bestimmung der optimalen Dicke oder Querschnittsfläche eines Bauteils zur Minimierung der Kosten oder Maximierung der gewünschten Eigenschaften. Die Strukturoptimierung befasst sich mit der iterativen Modifikationen der Parameter innerhalb Gestalt-definierender Randbedingungen wie beispielsweise einer Funktion, um eine optimale geometrische Gestalt zu finden. Topologieoptimierung bestimmt die optimale Materialverteilung durch Erzeugung und Verschmelzung von Hohlräumen. [122] Im Folgenden wird sich auf die Strukturoptimierung beschränkt, da die Parameteroptimierung eine vergleichsweise untergeordnete Rolle spielt und sich die Topologieoptimierung auf ein bestimmtes Portfolio an Zielgrößen beschränkt, die zwar Teile der Anpassungskonstruktion abdecken können aber nicht alle vereinen [122].

Die Optimierung umfasst drei generische Prozessschritte: Gestaltung, Analyse und Optimierung [124]. Die Gestaltung umfasst das Beschreiben der Geometrie sowie das Festlegen variabler Parameter. Die Analyse legt das meist numerische Analysewerkzeug fest. Mit dessen Hilfe wird die Bewertung der Gestalt hinsichtlich vorab definierter Zielgrößen untersucht. Die Optimierung umfasst das Aufstellen des Optimierungsproblems, die Auswahl des

Optimierungsalgorithmus sowie das iterative Abgleichen mit den Zielgrößen. Der generische Optimierungsprozess ist in Bild 19 dargestellt.

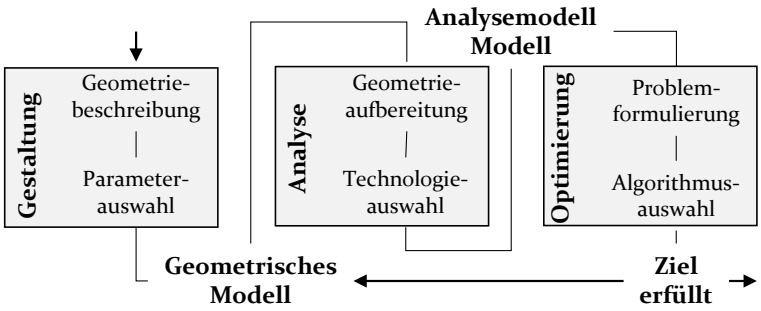


Bild 19: Illustration eines generischen Strukturoptimierungsprozesses in Anlehnung an [122]

Für das Optimierungsproblem wird in dieser Arbeit die Notation nach Marler und Arora verwendet (s. Gleichung 1) [125]. Dort ist das Ziel die Minimierung hinsichtlich k Zielgrößen des Vektors $f(x)$, welcher die Individuen x des Lösungsraums X in $\mathbb{R}$ abbildet[2]. Die Individuen x stellen Gestaltungsmöglichkeiten (Lösungen) da und bestehen aus x_i Merkmalen. Diese werden durch Restriktionen limitiert. Dabei wird in Ungleichheits- und Gleichheitsrestriktionen unterschieden, so dass sich die Restriktionen aus j Ungleichheitsrestriktionen $g_j(x)$ und l Gleichheitsrestriktionen $h_l(x)$ zusammensetzen. Üblicherweise hat jedes Merkmal x_i eine obere Grenze x_i^U und untere Grenze x_i^L. Diese Restriktionen spannen den Raum der möglichen Lösungen (engl. *Feasible Design Space*) auf.

$$\min_{x \in X} f(x) = [f_1(x), f_2(x), ..., f_k(x)]^T \tag{1}$$

$f_k(x)$	$k = 1, 2, ..., m$	Zielfunktion
$g_j(x) \leq 0$	$j = 1, 2, ..., m$	Ungleichheitsrestriktion
$h_l(x) = 0$	$l = 1, 2, ..., e$	Gleichheitsrestriktion
$x_i^L \leq x_i \leq x_i^U$	$i = 1, 2, ..., n$	Grenzen des Lösungsraums

Neben diesen direkten Methoden den Lösungsraum einzuschränken gibt es auch indirekte Methoden, die mit Hilfe sogenannter Pseudo-Zielfunktionen die Optimierung in die gewünschte Regionen des Lösungsraums lenken [126]. Durch das addieren des Strafterms $P(x)$ mit dem optionalen Skalierungsfaktor p, ist es möglich die Minimierung auch ohne feste Restriktionen

[2] Eine Maximierung einer Zielgröße wird in eine Minimierung transformiert indem mit -1 multipliziert wird.

durchzuführen (s. Gleichung 2). Dies ist vor allem dann sinnvoll, wenn die Restriktionen nicht klar definiert werden können oder sollen [127].

$$\hat{f}(x) = f(x) + pP(x) \tag{2}$$

$P(x)$ Strafterm
p Skalierungsfaktor

Im Allgemeinen besteht das Ziel von Mehrzieloptimierung (engl. *Multi-Objective Optimization*, MOO) darin, eine Lösung zu finden, auf die sich die Entscheidungsträger einigen können. Da die einzelnen Ziele meist in Konflikt zueinander stehen, kann in diesen Fällen keine eindeutige Lösung, die für alle Ziele gleichzeitig optimal ist, erreicht werden. Um einen Kompromiss zwischen den Zielgrößen zu finden, ist die Pareto-Optimalität wichtig.

Die Pareto-Optimalität

Aufgrund von Konflikten zwischen einzelnen Zielfunktionen in einem MOO-Problem gibt es in der Regel keine eindeutige Lösung, die für alle Ziele gleichzeitig optimal ist. Daher ist eine andere Definition der Optimalität erforderlich. Eine Lösung x_* ist nur dann Pareto-optimal, wenn es keine andere Lösung $x \in X$ gibt, für die gilt $f(x) \leq f(x*)$ und $f_i(x) < f_i(x*)$ für mindestens ein $i = 1, 2, ..., n$ [125]. Die Menge der Pareto-optimalen Lösungen (oder Punkte) wird als Pareto-optimale Menge bezeichnet. Im Lösungsraum kann diese Menge als Pareto Front visualisiert werden (s. dunkelgraue Lösungen Bild 20). Ausgehend von der Pareto-Front kann den Utopia-Punkt der Zielfunktionen als der Vektor definieren werden, welcher in Bezug auf alle Zielfunktionen optimal ist. Dieser ist in der Regel keine zulässige Lösung, da mehrere Ziele miteinander in Konflikt stehen und nicht gleichzeitig optimal sein können. Bei einer Minimierung stellt er die untere Schranke der Pareto-optimalen Menge dar. Im Gegensatz dazu liefert der Nadir-Punkt die obere Schranke der Pareto-optimalen Menge. Der Utopia-Punkt und der Nadir-Punkt sind in Bild 20 durch ein dunkelgraues bzw. hellgraues Kreuz dargestellt.

Die Skalierungsverfahren

Gemäß Gleichung 1 werden MOO-Probleme durch einen Vektor von Zielfunktionen charakterisiert. Dieser kann durch Kombination seiner einzelnen Komponenten in eine einzige skalare Zielfunktion transformiert werden. Dies wird als Skalierung bezeichnet und bietet den Vorteil, dass die Optimierung eines Ziels sehr gut entwickelt ist und Standard-Optimierungs Werkzeuge verwendet werden können [125]. In den Ingenieurswissenschaften werden

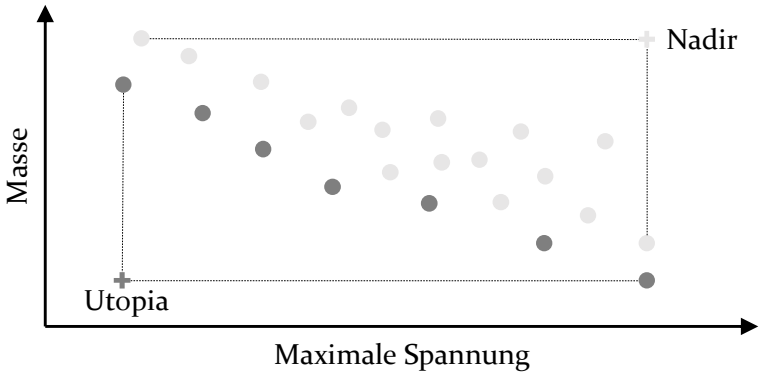


Bild 20: Darstellung der Pareto Front, des Utopia-Punktes sowie des Nadir-Punktes im zweidimensionalen Lösungsraum eines Minimierungsproblems

hier häufig die Methode der gewichteten Summen oder das Verfahren nach CHEBYSHEV angewandt [126]. Bei beiden Methoden werden die Präferenzen eines Entscheidungsträgers a priori, d. h. vor der eigentlichen Optimierung definiert, so dass die relative Bedeutung einer einzelnen Zielfunktion widergespiegelt wird [125].

Bei der Methode der gewichteten Summen, wird der Vektor linear transformiert, indem jede individuelle Zielfunktion f_i mit einem nicht negativen Gewicht t_i multipliziert und anschließend über alle Zielfunktionen summiert wird. Das Optimierungsproblem kann dann wie in Gleichung 3 dargestellt werden.

$$\min_{x \in X} f(x) \sum_{i=1}^{k} t_i f_i(x) \tag{3}$$

$f_i(x)$	$i = 1, 2, \ldots, k$	Zielfunktion
$t_i(x)$	$i = 1, 2, \ldots, k$	Gewichtungsfaktor

Bei der Verwendung der Methode der gewichteten Summen müssen zwei Einschränkungen bedacht werden. Erstens ist es mit diesem Ansatz nicht möglich, nicht-konvexe Teile der Pareto-Front im Zielfunktionsraum zu erhalten [126]. Zweitens garantiert eine konsistente und kontinuierliche Variation der Gewichte keine gleichmäßige Verteilung der Pareto-optimalen Lösungen [125]. Dieser Nachteil ist jedoch weniger schwerwiegend bei der Anwendung auf kontinuierliche, nichtlineare MOO-Probleme, bei denen der Entscheidungsträger Präferenzen formuliert, die die Optimierung auf eine konkrete Region der Pareto-Front konzentrieren [125].

Im Gegensatz dazu ist die Methode nach CHEBYSHEV für allgemeine nichtkonvexe Optimierungsprobleme geeignet. Der Utopia-Punkt $f(x^\star)$ wird

genutzt, um die Abstände der individuellen Zielfunktionen des Vektors $f(x)$ zu berechnen. Die Abstände werden mit nicht negativen Gewichten t_i multipliziert (s. Gleichung 4).

$$\min_{x \in X} \max_{1 \leq i \leq k} \{t_i(f_i(x) - f_i(x^*))\} \tag{4}$$

In der Praxis haben verschiedene Zielfunktionen unterschiedliche Einheiten, daher ist es oft vorteilhaft, die ursprüngliche Zielfunktion in eine gemeinsame numerische Skala zu transformieren, um eine bessere Vergleichbarkeit zwischen den Zielen zu erreichen und die Zielfunktionen dimensionslos zu machen [125]. Dabei darf die Skalierung nicht mit der Normierung verwechselt werden. Die Normierung transformiert eine Zielgröße auf einen definierten Wertebereich. Hier haben die Faktoren der Normierung keinen Bezug zu anderen Zielgrößen. Die Skalierung dient zur Transformation auf eine eindimensionale Zielfunktion und legt die Präferenzen der individuellen Zielfunktionen fest. Wird beispielsweise ein Bauteil auf Gewicht und Steifigkeit optimiert, kann der Anwender entscheiden, welche der individuellen Zielgrößen präferiert werden soll und die Gewichte dementsprechend festlegen. Es wird empfohlen, Normierung und Skalierung streng voneinander zu trennen [125]. Die Modellierung von Präferenzen ist für MOO-Probleme unerlässlich, wird aber in dieser Arbeit nicht untersucht, weil sie dem Anwender und dem spezifischen Anwendungsfall vorbehalten ist. Für weiterführende Informationen zu Skalierungsverfahren wird auf die Literatur verwiesen [128].

Optimierungsansätze für die Automatisierung der Anpassungskonstruktion

Die Übertragung der Mehrzieloptimierung auf die Synthese von Merkmalen wird als Stukturoptimierung bezeichnet. Obwohl frühe Sturkuroptimierungsprobleme auch parametrische Modelle als geometrische Beschreibung des zu optimierendes Objektes genutzt haben [129–131], hat sich das FEM-Netz als geometrische Beschreibung durchgesetzt [122]. Im Rahmen des definierten Netzes beschreibt die Position der Knoten den Lösungsraum der Strukturoptimierung [132]. Dieser muss jedoch weiter eingeschränkt werden, um unbrauchbaren Geometrien entgegenzuwirken. Hierzu können Netzbasierte [133] oder Isogeometrische [134] Restriktionen verwendet werden, um die Gestaltungsfreiheiten des Optimierers auf realisierbare Geometrien einzuschränken. Für die Analyse nutzen kommerzielle Werkzeuge aktuell fast ausschließlich die Isogeometrische Analyse und deren Erweiterungsmöglichkeiten [122]. Für die Optimierung können verschiedenste Algorithmen genutzt werden, wie die Literaturrecherchen [122, 133, 135] zeigen. Kürzlich veröffentlichte Beiträge zeigen gradientenbasierte Technologien [136, 137] sowie

Gradienten-frei Genetische Algorithmen [138–141] und Partikelschwarmoptimierung [142–144]. Dabei werden Geometrien für mehr als eine Anforderung (Zielgröße) gesucht.

Strukturoptimierung bietet sich vor allem für die Automatisierung des Konstruktionsprozesses an, wenn strukturmechanische oder strömungsmechanische FEM als Analysewerkzeug eingesetzt werden sollen. Hier stehen ausgereifte, kommerzielle Softwarelösungen zur Verfügung. Müssen nicht FEM-spezifische Zielgrößen angepasst werden, ist die Strukturoptimierung nur bedingt geeignet, da die Abstimmung von Geometrischer Beschreibung, Analyse- und Optimierungsverfahren sehr herausfordernd ist. Außerdem bringt die Strukturoptimierung die Einschränkung mit sich, dass die vorgeschlagenen Lösungen auf Netzgeometrien basieren. Sie müssen für ein parametrisches Modell zurückgeführt werden. Diese Aufgabe ist nicht eindeutig und benötigt die Unterstützung durch die Produktentwickelnden [122]. Obwohl die Isogeometrische Analyse dieses Problem für flächige, kontinuierliche Bauteile weitestgehend löst, sind Feature-basierte Geometrien nicht reversibel [145]. Schließlich kann abhängig vom verwendeten Optimierungsalgorithmus der Weg zur vorgeschlagenen Geometrie schlecht oder gar nicht nachvollzogen werden.

2.3.6 Die Wissensbasierte Produktentwicklung

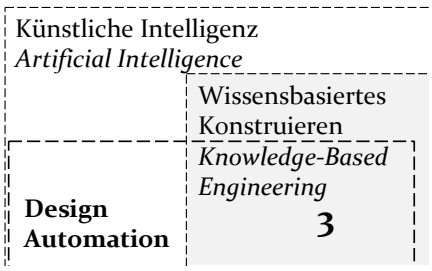


Bild 21: Ausschnitt der Forschungslandschaft für die Wissensbasierte Produktentwicklung

Neben der Strukturoptimierung ist die Wissensbasierte Produktentwicklung ein Treiber der Automatisierung, welcher Methoden der objekt-orientierten Programmierung (OOP), der künstlichen Intelligenz und der Rechnerunterstützung umfasst [3, 15]. Obwohl die Produktentwicklung maßgeblich auf den Erfahrungen und dem Wissen der beteiligten Personen aufbaut, wird explizit von Wissensbasierter Produktentwicklung gesprochen, wenn eine Rechnerunterstützung mittels Wissensbasierter Systeme (engl. *Knowledge-based System*, KBS) vorliegt [146–148]. Ziel des KBE ist die gedankliche Durchdringung des Konstruktionsprozesses, damit dieser im Rechner in Teilen abgebildet werden kann und für die verbleibenden Teilschritte eine Unterstützung der Anwendenden ermöglicht wird [57]. Dies erfordert den Umgang mit organisationalem Wissen, welches in einer Wissensbasis repräsentiert werden muss [149]. Das Wissensmanagement

(engl. Knowledge Management) bietet Methoden und Konzepten an, um das Wissen wie herkömmliche Produktionsfaktoren zu managen. Es unterteilt sich in den Wissensaustausch und die Wissenssicherung [150]. Der Wissensaustausch umfasst die Wissensakquisition – einem zweiphasigen Prozess aus Wissenserhebung und Wissensinterpretation – sowie die Wissensbereitstellung [11]. In dieser Arbeit wird nur die Wissensbereitstellung genauer beleuchtet, da die Automatisierung auf der Wissensbasis aufsetzt und diese für zukünftige Entwicklungszyklen nutzt. Dafür ist ein Verständnis der Begriffe Wissen und Wissensrepräsentation im Kontext der Produktentwicklung erforderlich.

Das Wissen im Kontext der Produktentwicklung

Die Automatisierung des Konstruktionsprozesses nutzt Daten, Informationen und Wissen als Hilfsmittel zur gewollten Reduktion der menschlichen Interaktion. Im Wissensmanagement und der Wissensbasierten Produktentwicklung werden die drei Begriffe nach der Taxonomie des Wissens von voneinander streng getrennt [151–153]. Hierfür hat sich die Taxonomie des Wissens in Anlehnung an die Wissenstreppe nach NORTH etabliert (siehe Bild 22). Daten werden durch Syntax verknüpfte Zeichen beschrieben und stellen objektive Fakten dar. Durch eine Semantik werden die Daten in einen Kontext gebracht und bilden Informationen ab. Wissen entsteht durch gezielte Vernetzung von Informationen [151]. Für die Automatisierung ist diese Unterteilung relevant, da für die Verarbeitung von Daten andere Technologien (bspw. Maschinelles Lernen vgl. Unterabschnitt 2.3.7) notwendig sind, wie für die Verarbeitung von Wissen (bspw. KBE).

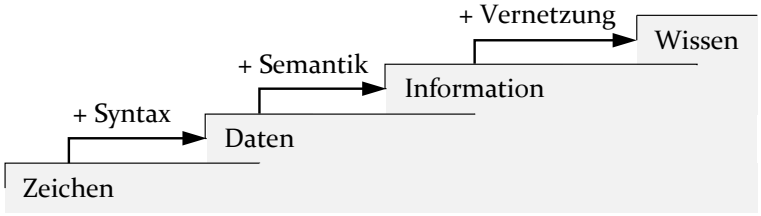


Bild 22: Die untersten vier Stufen der Wissenstreppe nach NORTH [151]

Wissen kann auf verschiedene Arten repräsentiert werden. Die Wissensrepräsentation bestimmt die Ausdrucksfähigkeit der Wissensbasis und hat damit einen starken Einfluss auf die Wissensverarbeitung [154, 155]. Für die Produktentwicklung werden Regeln, Zwangsbedingungen und Klassen als Repräsentationsarten genannt [153]. Außerdem kommen aggregierte Repräsentationsformen wie SysML Modelle und semantische Netze (Wissensgraphen) zum Einsatz. Regeln (engl. *If-Then Rules*) setzen sich aus einer Bedin-

gung und einer Aktion in Form einer Wenn-Dann-Beziehung zusammen [156]. Damit sind Regeln bidirektional und aufgrund der einfachen Formulierung besonders verbreitet [157] Zwangsbedingungen (engl. *Constraints*) beschreiben logische oder mathematische Zusammenhänge und schränken so den Lösungsraum ein [153]. Klassen (engl. *Frames*) stellen Wissen durch die Zuordnung spezifischer Eigenschaften als Attribute dar. Sie sind vergleichbar mit Klassen der Objekt-orientierten Programmierung. Instanzen – konkrete Ausprägung – eines Objektes beinhalten Werte wie Zahlen oder Bezeichnungen für die Attribute. Mit Hilfe von Relationen zwischen den Klassen, wie die in Unterabschnitt 2.3.4 gezeigten Eltern-Kind-Beziehungen lassen sich Attribute vererben [158].

In der Praxis kann meist keine der genannten Wissensrepräsentationen vollständig oder allgemeingültig sein und daher ist eine Kombination üblich [63]. Dabei ist es die Aufgabe der Anwendenden, die jeweils richtige Repräsentationsart entsprechend der Problemstellung auszuwählen [155]. Aggregierte Repräsentationsformen setzen sich aus mehreren Repräsentationsarten zusammen [154]. Klassen lassen aufgrund ihres Objekt-orientierten Charakters in Unified Model Language (UML) Diagrammen darstellen und in relationalen Datenbanken umsetzten. Der UML Sprachrahmen ist absichtlich universell gehalten und erlaubt die Anpassung an den jeweiligen Einsatzzweck durch spezifische Sprachprofile [159]. Für die Produktentwicklung erweitert die SysML diese mit zusätzlichen, spezifischen Diagrammen, beispielsweise für die Modellierung von Anforderungen. Diese stellen mit vordefinierten Eigenschaften bereit, um allgemeingültige Regeln oder Zwangsbedingungen mit Klassen zu kombinieren [160, P2]. Beispielsweise können in Anforderungsdiagrammen beliebige *Key-Value* Paare für die Anforderungen modelliert werden. Für den Abgleich mit den Eigenschaften des Produktes ist eine vordefinierte Beziehung (*satisfy*) bereits durch das Diagramm vorgegeben. Sie muss nicht manuell modelliert werden, sondern kann direkt instanziiert werden. Diese Aggregation erleichtert die Modellierung und beugt Fehlern vor.

Im Gegensatz zum relationalen Charakter der SysML basieren semantische Netze auf semantischen Beziehungen, welche sich gut als Netz oder Graph visualisieren lassen [161]. Können die Datenmodelle von Graphen aus Tripeln (beispielsweise: Subjekt, Prädikat, Objekt) aufgebaut werden. Die Inhalte der Triple werden Entitäten genannt. Sie repräsentieren Aussagen (beispielsweise: Auto, hat, Räder). Diese haben den Vorteil, dass sie sowohl für die Anwendenden als auch für den Computer verständlich sind. Darüber hinaus können Entitäten durch standardisierte Namensgebung (bspw. durch URIs – engl. *Uniform Resource Identifier*) eine eindeutige und weltweit gültige Bezeichnung erhalten. Datenmodelle können in dem auf XML aufbauenden *Ressource*

Description Frame (RDF) Format modelliert werden. In RDF modellierte Entitäten haben eine URI und können durch Attribute, welche ebenfalls eine URI benötigen, in Relation gebracht werden. So modellierte Graphen können das Wissen einer Domäne repräsentieren, wenn die Entitäten durch eine Ontologie vernetzt werden. Eine Ontologie beschreibt eine Domäne durch eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung [162]. Die Konzepte der Ontologie bilden die Taxonomie der Domäne und können in der *Web Ontology Language* (OWL) repräsentiert werden [163]. OWL beinhaltet die Logik erster Ordnung und ermöglicht die Verwendung von Reasonern für die Inferenz. Dadurch können sich Zusammenhänge, die sich aus der Logik ergeben, geschlussfolgert werden. Zusätzliche Relationen können durch Axiome dargestellt werden.

Wissensbasierte Systeme für Automatisierung der Anpassungskonstruktion

Der generische Aufbau von Wissensbasierten Systemen wurde in der Literatur mehrfach beschrieben und weiterentwickelt [63, 158]. Er ist in Bild 23 dargestellt. Die zentralen Elemente der KBS für die Automatisierung sind die Wissensbasis, die Inferenzkomponente und die Dialogkomponente. Daneben zählen die Komponenten zur Wissensakquisition, zur Erklärung und für den Kontext ebenfalls zu den Elementen von KBS, welche oftmals auch als optional angesehen werden [63]. Die Wissensbasis umfasst das domänenspezifische Wissen des KBS [63]. Das Wissen selbst liegt dabei in einer entsprechenden Repräsentation für den Rechner verarbeitbar vor. Die Lösung einer Problemstellung findet in der Inferenzkomponente unter der Verarbeitung des Wissens aus der Wissensbasis statt. Sie wird deshalb auch als Problemlösungs-, Schlussfolgerungs- oder Wissensverarbeitungskomponente bezeichnet [63]. Die Problemlösung wird den Anwendenden in der Dialogkomponente dargestellt, in welcher ebenfalls die Eingabe der relevanten Daten stattfindet. Sie definiert die Eingangs- und Zielgrößen des KBS. Deren Umsetzung erfolgt in der Regel als graphisches und interaktives Modul, welches in die Entwicklungsumgebung der Anwendenden integriert sein kann.

Folglich lassen sich KBS hinsichtlich der Wissensrepräsentation (Wissensbasis), Art der Wissensverarbeitung (Inferenzkomponente), den betrachteten Eingangs- und Zielgrößen (Dialogkomponente) charakterisieren. KBS sind in vielen Domänen vertreten. Innerhalb der Produktentwicklung hat sich der Begriff des Assistenzsystems für KBS als Analogie zu Fahrerassistenzsystemen, welche das Ziel haben, den Fahrer beziehungsweise den Anwender von Routinetätigkeiten zu entlasten und zu unterstützen, etabliert [164]. Tabelle 3 fasst eine Reihe von Assistenzsystemen aus dem Kontext der Produktentwicklung zusammen, welche in [157] identifiziert wurden.

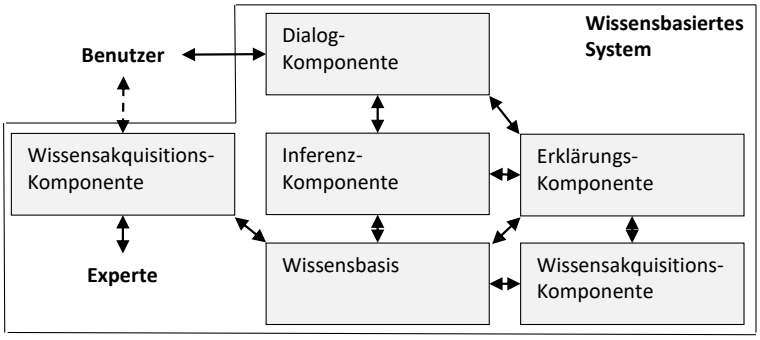


Bild 23: Darstellung des generischen Aufbaus einer KBS nach [63]

Einschränkungen für den Einsatz von KBE zur Automatisierung ergeben sich aus der schlechten Übertragbarkeit und der Umsetzung existierender Standards. Bisherige KBE Anwendungen für die Automatisierung sind speziell für einen Anwendungsfall konzipiert [65]. Anstatt einen strukturellen Rahmen oder eine Methodik zur Entwicklung von KBE Anwendungen zu nutzen, scheint es, dass Entwickelnde ein Problem identifizieren und hierfür eine KBE-Lösung auf der Grundlage eines individuellen Prozesses improvisieren [3, 165]. Methodiken für die Entwicklung von KBE existieren (beispielsweise VDI 5610-2 [153], MOKA [96]), werden allerdings nur sehr selten berücksichtigt [3]. Übertragbarkeit, Wiederverwendbarkeit oder Transfer zu weiteren Anwendungsfällen ist meist nicht vorgesehen [3, 166, 167]. Eine mögliche Begründung hierfür ist, dass ein Großteil der Ansätze Wissen aus konkreten Anwendungsszenarien einer anderen Phase des Produktlebenszykluses, beispielsweise Wissen über die Fertigung eines speziellen Bauteils, zurück in die Produktentwicklung führt. Das Wissen aus dem Produktentwicklungsprozess selbst wird selten berücksichtigt. Hier fehlt eine gemeinsame, deklarative Formalisierung der Konstruktionsaufgaben [65]. Hinsichtlich der Verwendung von Ontologien zur Wissensrepräsentation sind bisher nur wenige Arbeiten veröffentlicht worden. Geeignete Standards für Ontologien im Bereich der Produktentwicklung sind daher ebenfalls ein offenes Thema [168].

Tabelle 3: In [157] identifizierte Assistenzsysteme

AutorIn	Beschreibung
Meerkamm und Finkenwirth [169]	Konstruktionssystem mfk
Storath [170]	Kontextsensitive Wissensbereitstellung
Weber [171]	Entwicklung umweltgerechter Produkte
Schön [172]	Mechatronische Produktentwicklung
Wartzack [11]	Multikriterielle Bewertung alternativer Produktkonzepte
Hochmuth [173]	Rechnerunterstützte Analyse und Optimierung robuster Produkte
Düchting [174]	Freigabe- und Kommunikationsbasiertes System
Fathi u. a. [175]	Entscheidungsunterstützung in der Produktverbesserung
Lutz [35]	Rechnerunterstütztes Auslegen und Konfigurieren
Mayer; Spieckermann und Wenzel [176]	Assistenzwerkzeuge für Simulationsstudien
Spruegel und Wartzack [164]	Assistenzsystem für Finite-Element-Analysen
Kestel; Schneyer; Wartzack u. a. [177]	Automatischer Aufbau von Finite-Element-Analysen
Küstner [13]	Lärmreduzierte Auslegung rotierender Maschinen
Breitsprecher [14]	Selbstlernendes System für die Akquisition von konstruktionsrelevantem Fertigungswissen
Tüchsen [157]	Assistenzsystem für die Produktentwicklung elektrischer Motoren

2.3.7 Das Maschinelles Lernern

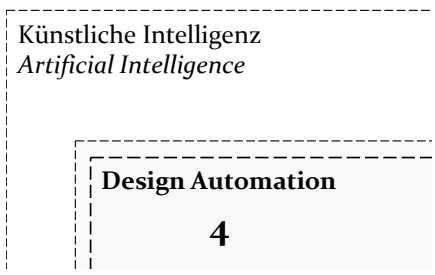


Bild 24: Ausschnitt der Forschungslandschaft für das Maschinelles Lernern

Lernen beschreibt den Vorgang der Verbesserung im Umgang mit zukünftigen Situationen im Vergleich zu Vergangen auf Basis von Beobachtungen. Dazu umfasst das Maschinelle Lernen die Methoden der Datenverarbeitung, der Mustererkennung und der Künstlichen Intelligenz [178]. Im Allgemeinen werden Beobachtungen, auf deren Grundlage gelernt wird, als Feedback bezeichnet [31]. Anhand des Feedbacks lassen sich Maschinelle Lernverfahren in drei Haupttypen unterteilen. Beim nicht überwachten Lernen (synm. unüberwachten Lernen [13, 28]) wird kein explizites Feedback bereitgestellt. Die Beobachtungen werden nicht durch Beschreibungen (engl. label) erklärt. Es werden intrinsische Zusammenhänge zwischen den Beobachtungen gelernt. Die Methoden des nicht überwachten Lernens umfassen das Clustern, die Anomalieerkennung, die Assoziation und die Visualisierung [178]. Beim überwachten Lernen wird das Feedback explizit beschrieben, indem die Beobachtungen in Paaren aus Eingabe (engl. *Features*) und Ausgabe (engl. *Targets*)

dargestellt werden [31]. Die Methoden des überwachten Lernens lassen sich hinsichtlich der Ausgabe in Klassifikation und Regression unterteilen. Wird die Ausgabe durch nominale Werte einer endlichen Menge (bspw. Kategorien) beschrieben, handelt es sich um eine Klassifikation. Reale Werte hingegen bedürfen einer Regression. [178] Das halb-überwachte Lernen ist eine Mischform, bei der mit Hilfe einer kleinen Menge explizit beschriebenen Beobachtungen Rückschlüsse auf eine größere Menge von nicht beschrieben Beobachtungen gelernt werden [31]. Das verstärkende Lernen ist der dritte eigenständige Haupttyp. Auf ihn wird detailliert in Abschnitt 2.4 eingegangen. Abseits des verstärkenden Lernens können die einzelnen Methoden, deren Verfahren und deren Validierungsmöglichkeiten in der Literatur nachvollzogen werden [13, 31, 179]. Wie für die anderen Technologien der Automatisierung des Konstruktionsprozesses werden in diesem Abschnitt die bereits gezeigten Umsetzungsmöglichkeiten vorgestellt. Anschließend werden mögliche Gründe für den Einsatz von Maschinellen Lernverfahren genannt und die momentanen Limitationen hinsichtlich Übertragbarkeit und Skalierbarkeit erörtert. Abschließend wird die Rolle des Wissens im Kontext der Maschinellen Lernprozesse analysiert.

Maschinelles Lernern zur Automatisierung der Anpassungskonstruktion

Hinsichtlich des Einsatzes zur Automatisierung des Konstruktionsprozesses werden vorrangig überwachte Lernprozesse verwendet, da Aussagen über die Zusammenhänge zwischen Eigenschaften und Merkmalen gefunden werden sollen [28]. Je nach Aktivität können die Maschinellen Lernprozesse dort in Analyse- und Synthesewerkzeuge unterschieden werden (s. Bild 25). Maschinelle Lernprozesse als Analysewerkzeuge unterstützen oder beschleunigen die Absicherung des virtuellen Produktes, indem Simulationen durch Metamodelle approximiert oder mit deren Hilfe die Simulation gezielt angeleitet werden [180]. Ein Metamodell beschreibt ein Modell aus Modellen [14]. Es bildet den Raum zwischen den bereits abgesicherten Versuchspunkten unter Angabe einer Unsicherheit ab [180]. Der Raum zwischen den Versuchspunkten wird auch als *Response Surface* bezeichnet [180, 181]. Wie bereits gezeigt, können auch Optimierungsalgorithmen für die Erstellung von Metamodellen verwendet werden. Bei der Auswahl von Maschinellen Lernprozessen zur Erstellung von Metamodellen muss beachtet werden, dass Verfahren, wie beispielsweise die lineare oder polynomiale Regression [12], welche einen vordefinierten Zusammenhang zwischen *Features* und *Targets* voraussetzen, nur gute Vorhersagemodelle erzeugen können, wenn das zu analysierende Systemverhalten den Annahmen entspricht. Bei der Absicherung von virtuellen Produkten ist der Zusammenhang zwischen den zu definierenden Merk-

malen und den resultierenden Eigenschaften typischerweise nicht bekannt, so dass eine falsche Wahl zu unzureichend genauen Metamodellen führt. Hierfür müssen Verfahren wie beispielsweise Neuronale Netze [12, 182] oder Kriging [183, 184, P3] eingesetzt werden, die sich selbständig an gegebene Datenpunkte anpassen [180]. Weitere Ansätze und Anwendungsszenarien können den Literaturrecherchen entnommen werden [181, 185, P4].

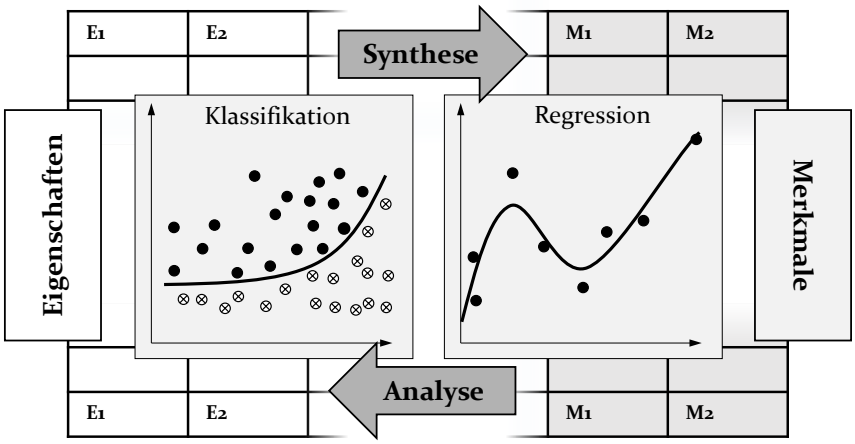


Bild 25: Unterteilung der überwachten, maschinellen Lernprozesse in Analyse- und Synthesewerkzeuge [179]

Synthesewerkzeuge auf Basis eines Maschinellen Lernprozesses sind vergleichsweise selten [186]. Die Vorhersage von Produktmerkmalen konnte mit Hilfe von Regressionsmodellen (Support Vector Regression [187–189], Gaußprozessen [190, 191]) und Neuronalen Netzen [25, 192, 193] für sehr spezifische Einzelfälle wie B-Säulen oder Turbinenschaufeln gezeigt werden. Parallel zur Produktentwicklung wurde die Synthese von 3D Objekten in der Grafik und Bildverarbeitung erforscht. Wu u. a. zeigen wie mit Hilfe von zwei konkurrierenden, tiefen, gefalteten Neuronalen Netzen (*Generative Adversarial Network* nach [195]) die Nachahmung von beliebigen 3D Objekten gelernt werden kann [194]. Bei der Recherche für diese Arbeit konnten keine Ansätze dieser Technologien im Bereich des Konstruktionsprozesses gefunden werden. Die Verknüpfung dieser rein geometrischen Technologien mit Expertenwissen aus der Produktentwicklung stellt nach den Erkenntnissen dieser Arbeit eine potentielle Forschungslücke dar.

Der Einsatz Maschineller Lernverfahren hat mit der Verfügbarkeit von Daten, deren Infrastruktur – Vernetzung und Speicherung, sowie der steigenden Rechnerleistung stark zugenommen [27]. Russell und Norvig nennen drei Gründe für den Einsatz von Maschinellem Lernen. Erstens eignet sich Maschi-

nelles Lernen für Aufgabenstellungen, in denen nicht alle möglichen Situationen vorhergesehen werden können. Außerdem können Maschinelle Lernprozesse Veränderungen in der gelernten Umgebung mit abbilden. Demnach stellen sie sich dynamisch auf neue Situationen ein. Zuletzt können Maschinelle Lernprozesse auch dann eingesetzt werden, wenn keine Vorstellung zur Lösung der Aufgabe existiert. [31] Limitationen für Maschinelle Lernprozesse ergeben sich selten aus den Möglichkeiten der Verfahren, sondern viel mehr aus der Qualität der bereitstehenden Daten [13, 27]. Die Automatisierungsmöglichkeiten für Maschinelles Lernen werden durch benötigte Datenmengen, hohe Datendimensionalität, heterogene Datenaspekte und geringe Datenqualität eingeschränkt [27]. Im Bezug auf Merkmale ist die hohe Datendimensionalität besonders herausfordernd [21, 26, 196]. Für Analyse- oder Synthesewerkzeuge bestimmt die Anzahl der Merkmale die Dimensionalität der Eingangs- oder Ausgangsgrößen. Folglich hat sie einen direkten Einfluss auf das Lernverfahren. Wird beispielsweise ein Neuronales Netz verwendet muss die Architektur auf die entsprechende Größe angepasst werden. Je aufwändiger das Lernverfahren gestaltet werden muss, desto mehr Stellgrößen innerhalb des Verfahrens müssen trainiert werden. Am Beispiel des Neuronalen Netzes müssen zusätzliche Kantengewichte approximiert werden. Je mehr Stellgrößen angepasst werden, desto mehr Daten werden zum Training benötigt. Dieses sehr alte Phänomen wird als Curse-of-Dimensonality hinsichtlich der verschiedenen Verfahren diskutiert [22].

Der Fluch der Dimensionalität

Der sogenannte *Curse-of-Dimensionality* wird auf die Einleitung von BELLMANS Buch „Dynamic Programming“ zurückgeführt [94]. Dort wird festgehalten, dass das Finden einer optimalen Lösung, in einem Gleichungssystem mit der Anzahl der Gleichungen schwieriger wird. Diese auf den ersten Blick sehr logische Formulierung wurde seitdem auf viele Ansätze übertragen [22]. Für das Maschinelle Lernen bedeutet, dass das die Anzahl der in Relation zubringenden *Features* (Eingaben) und *Targets* (Ausgaben) nicht unabhängig von der Anzahl der Stichproben des zugrundeliegenden Datensatz betrachtet werden kann. Im Allgemeinen bestimmt die Anzahl der *Features* die Dimension des Lösungsraums. Je größer der Lösungsraum ist, desto mehr Stichproben sind wahrscheinlich notwendig, um diesen ausreichend darzustellen. Im Kontext des Maschinellen Lernens wird dann von einem spärlich besetzten Lösungsraum gesprochen. Dieser muss nicht zwangsläufig problematisch sein. Sind einfach Modelle, im Extremfall eine lineare Regression, geeignet die Zusammenhänge zwischen Ein- und Ausgängen zu inferieren, reichen wenige Stichproben aus, da nur zwei Parameter gelernt werden müssen. Sind jedoch komplizierte Modelle, wie Künstliche Neuronale Netze (vgl. Unterab-

schnitt 2.4.3) notwendig, weil der Lösungsraum mathematische aufwendiger beschrieben werden muss, kann ein spärlich besetzter Lösungsraum die Wahrscheinlichkeit auf eine gute Prognosegüte des Modells stark limitieren [22, 197].

Neben den Veröffentlichungen, die die Datenmenge und -qualität direkt als Limitation aufführen [13, 21, 26, 27, 196], soll die Problematik für die Automatisierung der Anpassungskonstruktion kurz an einem Gedankenspiel verdeutlicht werden. Wird ein Modell zur Analyse oder Synthese im Konstruktionsprozess verwendet, bestimmt die Anzahl der betrachten Merkmale die Anzahl der Ein- oder Ausgänge des Modells. Nur in spezifischen Fällen können einfache Modelle, wie lineare Regression eingesetzt werden. Sollen beispielsweise diskrete Parameter, wie Materialien oder Blechdicken, im Modell berücksichtigen, ist davon auszugehen, dass Unstetigkeiten auftreten und Lineare oder Polynomiale Regression nicht mehr ausreichen. Daher muss im Allgemeinen davon ausgegangen werden, dass die Vorhersage des Zusammenhangs zwischen Merkmalen und Eigenschaften ein kompliziertes Modell erfordert. Hier sollten aus den im vorherigen Absatz genannten Gründen spärliche Lösungsräume vermieden werden. Diese treten auf, wenn die Anzahl der vergleichbaren Produkte gering in Relation zur Anzahl der Merkmale ist. Vergleichbar bedeutet, dass die Merkmale und Eigenschaften nur in ihren Ausprägungen variieren. Das Hinzufügen eines Merkmals hat zur Folge, dass die Dimension des Lösungsraum (Analyse) oder die Anzahl der Zielgrößen (Synthese) wächst. Die Frage wie viele Stichproben pro Dimension des Lösungsraum zur Verfügung gestellt werden müssen, kann pauschal nicht beantwortet werden, weil sowohl vom Anwendungsfall, als auch vom Modell abhängig ist. In Summe führt dies zum Dilemma, dass je mehr Merkmale und Eigenschaften gleichzeitig betrachtet werden, desto mehr Strichproben müssen für die Modellierung der Zusammenhänge zur Verfügung stehen. Damit skalieren rein datengetriebene Ansätze schlecht mit immer komplizierten Produkten.

Die Abgrenzung zwischen der Verarbeitung von Wissen und Daten

Diese Arbeit differenziert zwischen Maschinellem Lernen und Wissensbasierter Produktentwicklung hinsichtlich des Umgangs mit Wissen. Wissensbasierte Produktentwicklung sammelt, speichert und verarbeitet Wissen, um daraus Schlussfolgerungen abzuleiten, die dann zur Entscheidungsfindung und schließlich zur Automatisierung verwendet werden können. Dort gibt es eine wissensverarbeitende Komponente. Beim Maschinellen Lernen wird der Begriff Wissen in drei verschiedenen Aspekten verwendet. Die erste Rolle des Wissens beim Umgang mit maschinellen Lernprozessen ist das A-Priori Wissen, welches bereits vor dem Training im Lernprozesses veran-

kert ist [31]. Beispielsweise die Gesetze der Logik oder eine Kernelfunktion. Dieses Wissen wird manuell von einem Experten bei der Auslegung des Maschinellen Lernprozesses implementiert. Des weiteren können Wissensrepräsentationen abgefragt werden, um Beobachtungen zu erhalten. Nach der gezeigten Definition des Wissens benötigt Wissen sowohl Kontext als auch Vernetzung [151]. Kontext kann (teilweise) aufrecht erhalten werden, in dem modellbasierte Ansätze zur Anreicherung der Beobachtungen mit deklarativen Informationen verwendet werden. Die Vernetzung wird durch die Abfrage der Wissensrepräsentation jedoch aufgehoben, da nur die Ergebnisse der Abfrage und nicht das Wissensnetz in den Lernprozess eingehen. Schließlich gibt es die Auffassung, dass ein trainierter Lernprozess wiederum Wissen darstellt, weil er die Informationen der einzelnen Beobachtungen vernetzt [12, 198].

2.3.8 Zusammenfassung Automatisierung

Im vorangegangenem Abschnitt wurden der Begriff Automatisierung, dessen Limitationen bei der Verwendung von Informations- und Kommunikationstechnologien sowie die Technologien der Rechnerunterstützten Produktentwicklung, der Optimierung, des Wissensbasierten Konstruierens und des Maschinellen Lernens hinsichtlich der Automatisierung der Anpassungskonstruktion analysiert. Für die Anpassungskonstruktion stellt die Rechnerunterstützung im Sinne des CAD Systems für die anderen Technologien häufig das Produktmodell und die Dialogkomponente zu den Anwendenden zur Verfügung. Des weiteren können durch Rechnerunterstütze Produktentwicklung Teile von Produkten aus intelligenten Katalogen automatisch generiert werden. Dies bietet sich vor allem für die höher standardisierten Teile des Anlagenbaus an. Für die detaillierte Synthese und Analyse von einzelnen Bauteilen muss auf die Technologien der Optimierung, des KBE und des ML zurückgegriffen werden. Diesen ist gemein, dass durch sie spezifische Anwendungsfälle schon weit automatisiert werden können, jedoch die Übertragbarkeit auf folgende Anwendungsfälle sehr limitiert ist. Beim Einsatz von datengetrieben Technologien muss zusätzlich beachtet werden, dass ausreichend Daten in entsprechender Qualität zur Verfügung stehen.

2.4 Das Selbstverstärkende Maschinelle Lernen

In diesem Abschnitt werden die relevanten Grundlagen des Selbstverstärkenden (Maschinellen) Lernens für diese Arbeit in Anlehnung an Sutton und Barto vorgestellt [32]. Zunächst werden Entscheidungsprobleme nach MARKOV (engl. *Markov Decision Process*, MDP) erläutert, da sie die im Folgenden verwendete Möglichkeit darstellen, Selbstverstärkendes Lernen zu

formulieren (s. Unterabschnitt 2.4.1). MDPs bilden dynamische Probleme in Zuständen ab. Hauptaugenmerk liegt dabei auf der Bellman-Gleichung und der TD-Methode. Erste drückt die Beziehung zwischen dem momentanen und den nachfolgenden Zuständen aus (s. Gleichung 9). Zweite schätzt nachfolgende Zustände ab, ohne das Ende des Lernprozesses abzuwarten. Um die Bellman-Gleichung zu lösen, muss die Wahrscheinlichkeitsfunktion für die Zustandsänderungen bekannt sein oder gelernt werden. Zum Erlernen müssen je nach Größe des Problems Künstliche Neuronale Netze verwendet werden. Deren Anpassung ist elementarer Bestandteil des Selbstverstärkenden Lernens und muss daher in Unterabschnitt 2.4.3 erläutert werden. Abschließend wird in Unterabschnitt 2.4.5 ein Einblick in den aktuellen Stand der Forschung des Selbstverstärkenden Lernens in der Produktentwicklung gegeben.

2.4.1 Die Entscheidungsprobleme nach Markov

Beim Selbstverstärkenden Lernen wird der Umgang mit einer Umgebung gelernt. Daher ist es nicht verkehrt im ersten Moment an einen Roboter zu denken, der menschliches Verhalten imitieren soll. Hierzu ist es notwendig, dass der Roboter – im Folgenden der Agent – weiß was er ändern kann – die Umgebung – und wie – die Aktionen. Der Lernprozess schätzt für einen bestimmten Zustand der Umgebung die Wahrscheinlichkeit aller Aktion im Hinblick auf die mögliche Verbesserung des Zustandes ab. MDP sind die gebräuchlichste Möglichkeit die Interaktionen zwischen dem Agenten und der Umgebung zu formulieren. Dazu wird der Lernprozess in diskreten Zeitschritten erfasst. Bild 26 veranschaulicht in Anlehnung an Sutton und Barto die Dynamik einer Agenten-Umwelt-Interaktion als sequentielles Entscheidungsproblem [32]. Der Agent findet die Umgebung in einem Zustand (im Englischen *State*) vor. Er wählt eine Aktion und übergibt diese an die Umgebung. Die Umgebung wird durch die Aktion auf den nächsten Zustand gebracht (im Englischen *Next State*). Zusätzlich erhält der Agent das Feedback – die Belohnung (im Englischen *Reward*). Ein MDP formalisiert solche sequentiellen Entscheidungsprobleme jeden Zeitschritt t durch ein Tupel aus:

- dem Zustand $s_t \in S$, wobei S die Menge aller möglichen Zustände ist,
- der Aktion $a_t \in A$, wobei A die Menge aller möglichen Aktionen ist,
- der Belohnung $r_t \in \mathbb{R}$,
- der Wahrscheinlichkeitsfunktion $P^a_{ss'} \rightarrow [0, 1]$,
- und dem Discount Faktor $\gamma; 0 \leq \gamma \leq 1$ [32].

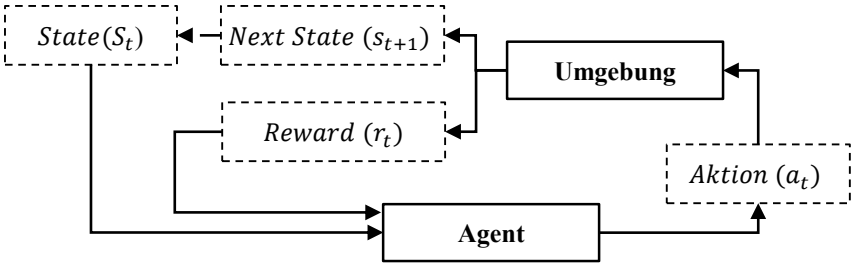


Bild 26: Darstellung der Interaktion zwischen Agent und Umwelt als Grundlage des Selbstverstärkenden Lernens in Anlehnung an [32]

Ziel des Agenten ist es, die zukünftigen Belohnungen zu maximieren. Hierfür gibt die Wahrscheinlichkeitsfunktion $P_{ss'}^{a}$ an, wie groß die Wahrscheinlichkeit des Übergangs vom Zustand s in den Zustand s' durch die Aktion a ist. Für alle Zustände ergibt sich eine Abbildung, die als Strategie (engl. *Policy* Π) des Agenten bezeichnet wird. Ist die Strategie bekannt, kann von jedem Zustand aus eine passende Aktion gewählt werden. Da dies selten der Fall ist, muss die Strategie gelernt werden. Hierfür sind drei mathematische Grundlagen erforderlich – das *Reward Discounting*, die Markov-Eigenschaft und die Bellman-Gleichung.

Das Prinzip des Discountings

Das Ziel ist zukünftige Belohnungen zu maximieren. Dazu wird die Summe über alle erhaltenen Belohnungen gebildet. Damit diese Summe einen Grenzwert hat, werden zukünftige Belohnungen mit dem Discount Faktor γ gewichtet (s. Gleichung 5). Der Discount Faktor γ bestimmt den Wert zukünftiger Belohnungen für den aktuellen Zeitschritt. Eine in der Zukunft erhaltene Belohnung wird durch den Faktor γ^{k-1}, der zwischen 0 und 1 liegt, gewichtet. Folglich ist der Wert der Belohnung um diesen Faktor geringer, als sie wäre, wenn sie zum aktuellen Zeitschritt erhalten worden wäre [32]. Wenn $\gamma; 0 \leq \gamma \leq 1$ ist, hat die Rendite einen endlichen Wert. Wenn γ sich der 1 nähert, berücksichtigt der Agent zukünftige Belohnungen stärker. Wenn $\gamma = 0$ ist, konzentriert sich der Agent nur auf die Maximierung der unmittelbaren Belohnungen.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1} \qquad (5)$$

G_t	$t = 1, 2, \ldots$	Summe aller zukünftigen Belohnungen
r_t	$t = 1, 2, \ldots$	Belohnung zum Zeitpunkt t
γ		Discount Faktor

Wichtig sind dabei zwei Erkenntnisse. Zum einen ist die Summe der zukünftigen Belohnungen endlich, weil γ^k für $k \to \infty$ gegen 0 geht. Zum anderen lässt sich der Unterschied zwischen den zukünftigen Belohnungen der Zeitpunkte t und $t+1$ durch γ und die unmittelbare Belohnung R_{t+1} berechnen. Dies wird relevant, wenn der Wert eines Zustandes für den Agenten bestimmt werden soll.

Die Markov-Eigenschaft

Die Markov-Eigenschaft bezieht sich auf die Umgebung und ihre Zustände. Ihr Zweck ist es zukünftige Zustände nur abhängig vom momentanen Zustand zu machen. Dazu sollte ein Zustand $s \in S$ alle Informationen, die dem Agenten zur Verfügung stehen, kompakt zusammenfassen. Zustände, die nur die notwendigen unmittelbaren Informationen für den Übergang vom aktuellen in den nächsten Zustand enthalten, werden als Markov-Zustände bezeichnet. Folglich ist die Zukunft unabhängig von der Vergangenheit, da der aktuelle Zustand s_t alle relevanten Informationen der Vergangenheit erfasst. Wenn S und A endlich sind, handelt es sich um einen endlichen MDP, bei dem für jeden Zustand $s \in S$ und jede Aktion $a \in A$ die Wahrscheinlichkeit, dass s in den Zustand s' übergeht durch die Wahrscheinlichkeitsfunktion $P^a_{ss'}$ in Gleichung 6 ausgedrückt werden kann.

$$P^a_{ss'} = \mathbb{P}\left[s' | s_t = s, a_t = a\right] \tag{6}$$

$R^a_{ss'}$ drückt den Erwartungswert für die Belohnung r_{t+1} beim Übergang von s nach s' durch die Aktion a aus (s. Gleichung 7).

$$R^a_{ss'} = \mathbb{E}\left[r_{t+1} | s_t = s, a_t = a, s_{t+1}\right] \tag{7}$$

Wichtig ist zu erkennen, dass $P^a_{ss'}$ und $R^a_{ss'}$ nur von t bzw. $t+1$ abhängig sind und nicht von älteren (vergangenen) Zuständen. Wenn beide bekannt sind, ist das Verhalten des MDP vollständig bestimmt. Dies ist in der Realität selten der Fall.

Die Bellman-Gleichung

Algorithmen des Selbstverstärkenden Lernens beinhalten normalerweise eine Funktion, die den Wert (im Englischen *value*) von Zuständen (oder Zustands-Aktions-Paaren) hinsichtlich der Maximierung zukünftiger Belohnungen abschätzt. Wenn dies der Fall ist, werden die Lernstrategien hinsichtlich dieser Funktion unter dem Begriff *value-based* zusammengefasst. Diese Arbeit basiert auf Zustands-Aktions-Paaren. Wertfunktionen auf Basis von

reinen Zuständen sind für diese Arbeit nicht relevant und sind der Literatur zu entnehmen [32]. Üblicherweise wird zum Erlernen der Wertfunktion von Zustands-Aktions-Paaren der im anschließenden Abschnitt vorgestellte Q-Learning Algorithmus nach WATKINS & DAYAN verwendet [199]. Daher werden Werte für Zustands-Aktions-Paare Q-Values genannt.

Q-Values werden bezüglich der verwendeten Strategie definiert, da die erhaltene Belohnung von den durchgeführten Aktionen abhängt. So ist der Wert $Q^{\pi}(s,a)$ eines Zustands s und der Aktion a unter einer Strategie Π der erwartete Ertrag, wenn der Agent im Zustand s_0 beginnt und anschließend Π folgt. Dieser ergibt sich nach Gleichung 8.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right] \quad (8)$$

Nach Bellman erfüllt Gleichung 8 die zentrale Eigenschaft, welche bei Selbstverstärkendem Lernen genutzt wird, um die Beziehung zwischen dem Wert des momentanen Zustands und seiner möglichen Nachfolger beschreibt [94]. Für jede Politik Π gilt die Bellman-Gleichung in Gleichung 9 zwischen dem Q-Value $Q^{\pi}(s,a)$ und seinem möglichen Nachfolgern $Q^{\pi}(s',a')$

$$Q^{\pi}(s,a) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma Q^{\pi}(s',a')\right] \quad (9)$$

Bellman-Gleichung drückt die Beziehung zwischen dem Wert eines Q-Value $Q^{\pi}(s,a)$ und dessen möglichen Nachfolgern $Q^{\pi}(s',a')$ durch den Discount Faktor und die erwarteten Belohnungen auf dem Weg dorthin aus. Hierfür bildet Gleichung 9 den Durchschnitt über allen möglichen Zustände-Aktions-Paare und gewichtet jedes mit ihrer Eintrittswahrscheinlichkeit über $P^{a}_{ss'}$ die erwartete zukünftige Belohnung $R^{a}_{ss'}$.

Auf der Grundlage der Bellman-Gleichung kann eine Politik Π^* dann und nur dann optimal, wenn keine andere Strategie Π existiert, so dass $Q^{\pi}(s,a)^* \leq Q^{\pi}(s,a) \forall s \in S, a \in A$ gilt. Dies drückt die Optimalitätsbedingung nach Bellman in Gleichung 10 aus.

$$Q^{\pi}(s,a)^* = R^{a}_{ss'} + \gamma \sum_{s' \in S} P^{a}_{ss'} \max_{a' \in A} Q^{\pi}(s',a') \quad (10)$$

Wichtig ist, dass Gleichung 10 ein nichtlineares Gleichungssystem bildet, dessen Dimensionalität abhängig von der Anzahl der Zustände und Aktionen ist. Es kann eindeutig gelöst werden, wenn die Dynamik des MDPs ($R^a_{ss'}$ und $P^a_{ss'}$), welche in Gleichung 6 und Gleichung 7 dargelegt wurden, beschrieben werden kann. Bei realen Aufgaben sind jedoch die Modelle für die Zustandsübergangswahrscheinlichkeit $P^a_{ss'}$ die erwarteten zukünftigen Belohnungen $R^a_{ss'}$ nicht verfügbar. Daher muss die Strategie aus Erfahrungen mit der Umgebung gelernt werden. Hierzu wird im Folgenden das Q-Learning nach WATKINS & DAYAN vorgestellt.

2.4.2 Das Q-Learning

Gesucht ist die Strategie, die die Optimalitätsbedingung nach BELLMAN in Gleichung 10 erfüllt. Pragmatisch ausgedrückt, muss die Strategie solange angepasst werden, bis keine bessere gefunden werden kann. Damit die Strategie vollständig ausgewertet werden kann, muss ihr üblicherweise gefolgt werden, bis ein Endzustand erreicht ist (s. Gleichung 5). Dies ist vergleichbar mit einem Schachspiel, bei dem sich der Wert der einzelnen Züge erst am Ende der Partie ergibt. Da dies den Lernaufwand immens machen würde, basiert das Lernen auf der TD-Methode. TD steht für *Temporal Difference* und ist nach SUTTON & BARTO das entscheidende Konzept des Selbstverstärkenden Lernens, weil es den Lernaufwand auf die Berechnung des nächsten Schrittes reduziert [32]. Bei TD-Methoden wird nur bis zum nächsten Zustand s_{t+1} gewartet und dieser direkt ausgewertet. Dabei werden die zukünftigen Belohnungen lediglich auf Basis der bisherigen Erfahrungen abgeschätzt. Mehrere Beispiele für Annäherung von mit TD-Methoden abgeschätzten Belohnungen zu berechneten Belohnungen können Kapitel 6 in [32] entnommen werden. Für diese Arbeit reicht es festzuhalten, dass die Abschätzung des nächsten Zustandes ausreicht, um diese zu verbessern und über lange Sicht die Strategie des Agenten anzupassen. Die Anpassung erfolgt nach dem durch WATKINS & DAYAN vorgestellten Q-Learning in Gleichung 11 [199].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \qquad (11)$$

Es reicht demnach den Q-Value für die nächsten Zustände $Q(s_{t+1}, a)$ abzuschätzen und die aktuelle Belohnung r_{t+1} zu kennen, um die Strategie des Lerners zu verbessern. Um $Q(s_{t+1}, a)$ abzuschätzen können, werden je nach Komplexität des zugrundeliegenden MPD verschiedene Algorithmen in Betracht gezogen. Die Komplexität ergibt sich aus der Menge der Zustände S und Aktionen A. In Anlehnung an die Mehrzieloptimierung werden diese auch Zustands- bzw. Lösungsraum und Aktionsraum genannt. Im Fall von kleinen,

endlichen Mengen für die Zustände S und Aktionen A sind tabellarische Algorithmen vorzuziehen, da sie garantiert zu optimalen Lösungen konvergieren [32]. Im Fall von großen oder kontinuierlichen Zustands- und Aktionsräumen verwenden, werden tiefe Künstliche Neuronale Netze eingesetzt, um die Lösung zu approximieren [200]. Im Folgenden werden die notwendigen Prinzipien für die Anpassung von tiefen, Künstlichen Neuronalen Netzen vorgestellt, da für die Produktentwicklung von sehr großen Lösungsräumen ausgegangen werden muss.

2.4.3 Das Künstliche Neuronale Netz

Das künstliche neuronale Netz (KNN) soll die Funktionsweise des menschlichen Gehirns technisch nachbilden. Wie in Unterabschnitt 2.3.7 können sie zum Maschinellen Lernen zur Inferenz, d. h. zur logischen Abbildung von Eingangsgrößen in eine definierte Form von Zielgrößen, eingesetzt werden [31]. KNN werden aus Neuronen aufgebaut. Diese werden üblicherweise in Schichten vernetzt. Ein Neuron besteht aus drei Funktionen. Einer Eingangsfunktion, einer Aktivierungsfunktion und einer Ausgangsfunktion. Das

ɜELL &

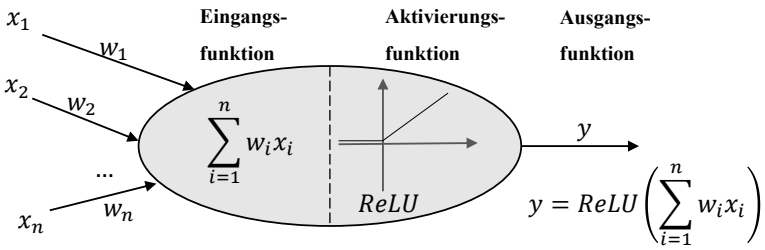


Bild 27: Darstellung eines Neurons in Anlehnung an [31]

Je nach Art der Vernetzung erhält ein Neuron Eingangswerte x_i, die mit den entsprechenden Kantengewichten w_i multipliziert und in der Eingangsfunktion aufsummiert werden. Zusätzlich kann dort ein skalarer Bias addiert werden. Ab einem definierten Schwellenwert schaltet die Aktivierungsfunktion das Neuron aktiv und gibt die Summe über die Aktivierungsfunktion an die Ausgangsfunktion weiter. Diese berechnet den Ausgangswert y und gibt diesen an die nächste Schicht ab. Je nach Aufgabe und Anwendung (bspw. Regression, binäre Klassifikation oder Mehrklassenklassifikation) können unterschiedliche Aktivierungs- und Ausgangsfunktionen eingesetzt werden. Diese können der Literatur entnommen werden [201]. Für diese Arbeit wird ausschließlich eine sogenannte ReLU (*Rectified Linear Unit*) verwendet, die

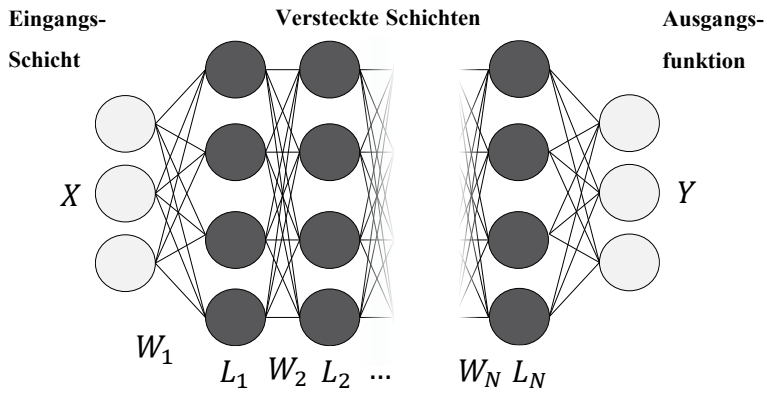


Bild 28: Darstellung Künstlichen Neuronalen Netzes

Durch die Verkettung von Neuronen zu Netzen, ist es möglich beliebig Funktionen zu approximieren. Dabei werden Neuronen häufig in Schichten angeordnet, um verschiedenen Gruppen von Neuronen verschiedene Eigenschaften zu übergeben. Üblicherweise gibt es eine Eingangsschicht, die die Eingangswerte empfängt und eine Ausgangsschicht, die die Vorhersagen – Ausgangswerte – widerspiegelt. Dazwischen liegt eine definierte Anzahl verdeckter Schichten. Für diese Arbeit sind lediglich vollständig verknüpfte Netze, die ausschließlich Vorhersagen im *Feed-Forward* Modus treffen, relevant. Eine solche *fully-connected feed forward* Architektur ist in Bild 28 dargestellt.

Die Backpropagation

Die Ausgangswerte des KNN sind von den Kantengewichten, den Schwellwerten und der Netzarchitektur (hier Anzahl der Knoten pro Schicht und Anzahl der versteckten Schichten) abhängig. Schwellwerte und Netzarchitektur bleiben fest, während die Kantengewichte durch Maschinelles Lernen an die Relation zwischen Aus- und Eingängen trainiert werden. Hierzu wird das Netz mit vorgegebenen Eingangswerten einmal vorwärts Durchlaufen und die Ausgangswerte berechnet. Diese berechneten Ausgangswerte werden mit den wahren Werten verglichen und der Fehler (im Englischen *loss*) für die Netzanpassung durch die Backpropagation genutzt (s. Bild 29).

Die Backpropagation erlaubt es die Gradienten für die Kantengewichte zu bestimmen [202]. In Gleichung 12 ergibt sich das Gewicht nach der Anpassung w^{t+1} aus der Differenz des Gewichtes vor der Anpassung w^t und dem

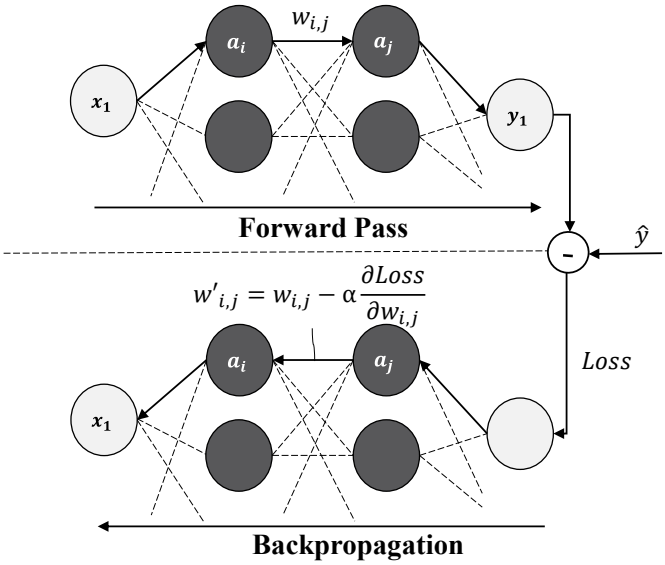


Bild 29: Darstellung des Forward Pass zur Berechnung der Ausgänge y und der Backpropagation zur Anpassung der Gewichte von w nach w' durch Berechnung des Loss aus den wahren Werten $\hat{y}$

Gradienten des Fehlers bezüglich des Gewichts. Die Differenz wird zusätzlich über die Lernrate α gesteuert.

$$w^{t+1} = w^t - \alpha \frac{\delta Loss}{\delta w} \tag{12}$$

w^t	$t = 1, 2, \ldots$	Gewicht zum Zeitpunkt t
α		Lernrate
$Loss$		Fehler

2.4.4 Das Deep Q-Learning

Die Verbindung aus KNNs und Q-Learning durch MNIH et al. wird als ein wesentlicher Durchbruch des Selbstverstärkenden Lernens betrachtet [200]. Beim Deep Q-Learning wird die Aktualisierungsregel nach WATKINS & DAYAN aus Gleichung 11 durch zwei tiefe KNNs approximiert. Da diese Q-Values bestimmen, werden sie DQNs (Deep Q-Learning Networks) genannt. Das sogenannte Target DQN wertet den nächsten Zustand s_{t+1} aus und wählt den höchsten Ausgangswert $maxQ(s_{t+1})$ bezüglich der möglichen Aktionen aus. Nach Gleichung 5 ergibt sich zusammen mit der aktuellen Belohnung r_{t+1} und dem Discount Faktor γ die Zielvorgabe. Mit deren Hilfe wird der Loss

aus Zielvorgabe und den aktuellen Ausgangswerten des zweiten DQNs (Evaluation DQN) bestimmt. Dieser Loss wird für die Anpassung der Gewichte des Evaluation DQN nach Gleichung 13 genutzt.

$$w^{t+1} = w^t - \alpha \frac{\delta(r_{t+1} - maxQ(s_{t+1}))^2}{\delta Q(s_t)} \tag{13}$$

2.4.5 Das Selbstverstärkende Lernen in der Produktentwicklung

Während das Selbstverstärkende Lernen bereits auf verschiedene Probleme in den Bereichen Robotik und Fertigung angewandt wurde, ist die Zahl der wissenschaftlichen Beiträge im Bereich der Produktentwicklung sehr gering. Die meisten Beiträge konzentrieren sich auf die numerische Strömungsmechanik (*Computational Fluid Dynamics*, CFD) und bauen auf [203] auf. RABAULT et al. beschrieben 2019 einen innovativen Ansatz zur Durchführung einer aktiven Strömungssteuerung in einer 2D-Simulation unter Verwendung eines Ansatzes der proximalen Strategieoptimierung durch Selbstverstärkendes Lernen [203]. Die Umgebung ist die 2D-Simulation der Strömung. Deren Zustände werden durch Geschwindigkeitsvektoren an mehreren Punkten in der CFD-Simulation dargestellt. Der Agent führt Aktionen zur Steuerung der Massenströme von zwei Düsen aus, um die Wirbelgasse zu stabilisieren. Beispielsweise, um den Massenstrom einer Düse zu erhöhen. RABAULT et al. beschleunigten diesen Ansatz durch paralleles Training des Agenten in einem Experiment mit mehreren CFD Umgebungen [204]. TANG et al. haben diese Umgebung durch verschiedene Reynoldszahlen erweitert und dabei die Generalisierungsfähigkeit des Selbstverstärkenden Lernens bei der aktiven Strömungssteuerung untersucht [205]. Die Veröffentlichungen [203–205] bilden die Grundlage für [206], in der VIQUERAT et al. die erste Optimierung von Formparametern durch Selbstverstärkendens Lernen vorstellen. In diesem Fall handelt es sich um ein tropfenförmigen Objekt, das mit Hilfe von Bezier-Kurven beschrieben wird. Diese vier Veröffentlichungen bilden eine der zwei Gruppen, die Selbstverstärkendes Lernen für die Produktentwicklung einsetzen. In diese Gruppe wird noch [207] eingenordet, in der die genannten Autoren ihre bisherige Forschung im Bereich des Selbstverstärkenden Lernens für CFD zusammenfassen. Neben RABAULT et al. haben YONEKURA et al. einen ähnlichen Rahmen für CFD-Umgebungen vorgestellt [208] . Im Bereich CFD schlugen YAN et al. einen Ansatz vor, der durch eine Kombination aus Selbstverstärkendem Lernen und Transferlernen die Positionierung der Flügel einer symmetrischen Rakete übernimmt, mit dem Ziel den Luftwiderstand der Rakete zu optimieren [209].

Die zweite Gruppe, die Selbstverstärkendes Lernen in der Produktentwicklung einsetzt, konzentrierte sich auf sehr einfache geometrische Objekte. Im Bauingenieurwesen werden 1D-Stab-Strukturen oder 2D-Fachwerken Strukturen, wie Rahmen oder Fachwerke, optimiert. HAYASHI et al. stellten Selbstverstärkendes Lernen für die binäre Topologieoptimierung von Fachwerken vor, um das Gesamtvolumen der Struktur unter Berücksichtigung von Spannungs- und Verschiebungsbedingungen zu minimieren [210]. HAYASHI et al. verwendeten einen gleichen Ansatz des Selbstverstärkenden Lernens, um das Gesamtvolumen von Fachwerkquerschnitten zu minimieren [211]. Im Bereich des Generative Design präsentierte SUN et al. einen Ansatz des Selbstverstärkenden Lernens, der in der Lage ist, die Anzahl der geometrischen Elemente, z. B. quadratische Platten in 2D, schrittweise zu reduzieren und gleichzeitig die Nachgiebigkeit für gegebene Randbedingungen zu berücksichtigen [212]. Dabei ist anzumerken, dass bei diesen Ansätzen das Verhalten der Finite-Elemente-Methode analytisch berechnet werden kann, ohne eine FEM-Software einzusetzen.

Neben diesen beiden Clustern wurde Selbstverstärkendes Lernen in der Fertigung und Produktion eingesetzt. Ein Ansatz, der geometrische Parameter berücksichtigt wurde von STOCKER et al. zur Sortierung von Schüttgut vorgeschlagen [213]. Sie nutzen Selbstverstärkendes Lernen, um einen Vibrationswendelförderer zu verbessern. Ihr Konzept geniert einen Entwurf, der die Orientierung von Komponenten durch Q-Learning hinsichtlich der Effizienz des Förderers bestimmt. Dabei ist der Lösungsraum so klein, dass eine Tabelle zur Abschätzung der Q-Values genutzt werden kann und kein DQN eingesetzt werden muss.

2.4.6 Zusammenfassung Selbstverstärkendes Lernen

Das Selbstverstärkende Lernen bildet die dritte, eigenständige Kategorie des Maschinellen Lernens. Die Grundlage bildet nicht ein Datensatz, sondern eine dynamische Interaktion von einem Agenten mit einer Umgebung, die durch Zustände beschrieben wird. Durch die Formalisierung dieser Umgebung mit Hilf des MDP lässt sich der Umgang mit der Umgebung auf die Übergangswahrscheinlichkeit von momentanen in einen zukünftigen Zustand zurückführen. Einfach ausgedrückt auf die Frage, wie wahrscheinlich ist eine Aktion, die zum nächsten Zustand führt, wenn sich der Agent im momentanen Zustand befindet. Da diese jedoch meist unbekannt ist, wird sie als Strategie gelernt. Eine Strategie gibt für jeden Zustand die Übergangswahrscheinlichkeiten für die nächsten Zustände an. Elementarer Bestandteil dieses Lernprozesses ist die TD-Methode. Ohne das Konzept der TD-Methode ist der Wert eines Übergangs zum nächsten Zustandes im Bezug auf die Zielgrößen

erst bekannt, wenn einer Strategie solange gefolgt wird, bis der Endzustand erreicht wird. Die TD-Methode schneidet die Berechnung des Wertes bereits nach dem nächsten Zustand ab und benutzt nur diesen zur Anpassung der Strategie. Da selbst die Berechnung des Wertes für den Übergang zum nächsten Zustandes einer großen Umgebungen sehr rechenintensiv sein kann, werden KNNs genutzt, um diesen abzuschätzen.

Diese Kombination aus der Bestimmung des Wertes eines Übergangs vom momentanen zum nächsten Zustand durch eine Aktion[3] und dem Einsatz von tiefen KNNs hat 2015 durch die Veröffentlichung von MNIH et al. zu einer regelrechten Welle an Konzepten für den Einsatz des Selbstverstärkendes Lernen geführt [200]. In der Produktentwicklung ist die Anzahl der Veröffentlichungen, die Selbstverstärkendes Lernen einsetzen, sehr gering. Keines der bisher vorgestellten Konzepte beinhaltet Merkmale von realen Produkten. Wenn geometrische Elemente in die Lernumgebung mit einbezogen wurden, wurden mathematische Beschreibungen von Oberflächenkurven, Stab- oder Balkenelementen genutzt. Diese bieten den Vorteil, dass analytische Lösungen für die Eigenschaften generiert werden können, haben jedoch keinen Bezug zu realen Produkten. Daher fehlt es an Ansätzen, die Technologie des Selbstverstärkenden Lernens den Produktentwickelnden zu eröffnen.

[3] Q-Learning nach WATKINS & DAYAN [199]

3 Ableitung des Handlungsbedarfs und der Forschungsfragen

Im vorangegangen Kapitel wurden die wissenschaftlichen Grundlagen im Kontext dieser Arbeit vorgestellt. Hierbei wurde der thematische Rahmen dargelegt und der Stand der Forschung und Technik in jedem Abschnitt abschließend zusammengefasst. In diesem Kapitel werden zunächst die wissenschaftlichen Grundlagen diskutiert (Abschnitt 3.1), um den Forschungsbedarf (Abschnitt 3.2) und anschließend die wissenschaftliche Zielsetzung und die Forschungsfragen abzuleiten (Abschnitt 3.3).

3.1 Fazit aus den wissenschaftlichen Grundlagen

Als Kontext dieser Arbeit wurde in Abschnitt 2.2 die Anpassungskonstruktion vorgestellt und als Konstruktionsart aus verschiedenen Perspektiven analysiert. An dieser Stelle muss erneut der Unterschied zwischen präskriptiver Theorie und gelebter Praxis unterstrichen werden. Aufgrund unterschiedlicher Ausgangspositionen und Rahmenbedingungen der Unternehmen, muss die konkrete Einordnung in Konstruktionsarten in der Praxis individuell erfolgen. Um einen wissenschaftlichen, generell übertragbaren Beitrag zu liefern, baut diese Arbeit auf der methodischen Vorstellung der Anpassungskonstruktion auf. Ausschlaggebend für die Anpassungskonstruktion ist dabei die Bearbeitungstiefe, d. h. welche Aktivitäten neu durchgeführt werden müssen, und wie groß der Einfluss des Vorgängers, d. h. welche Ergebnisse übernommen werden, ist. Aus der Literatur wurde abgeleitet, dass für diese Arbeit eine Anpassungskonstruktion vorliegt, wenn ein Vorgänger existiert, dessen Prinzipielle Lösung – Modulare Struktur und Lösungsprinzipen – übernommen wird. Die Aktivitäten definieren die quantitative Gestalt (wie beispielsweise geometrische Maße) aus der qualitativen Gestalt (wie beispielsweise der Definition der Wirkflächenpaare) durch Synthese der Merkmalsausprägungen, analysieren die sich ergebenden Eigenschaften und gleichen diese mit an die Anpassungskonstruktion gestellten Anforderungen ab. Auf die von PAHL & BEITZ herausgestellte Ausnahme, dass eine Anpassungskonstruktion auch dann vorliegt, wenn eine Komponente des Produktes neu konstruiert werden muss, kann nicht eingegangen werden, da Neukonstruktion im Allgemeinen aufgrund der kreativen Aspekte technologisch noch nicht berücksichtigt werden kann und soll [10].

Mit dieser Charakterisierung der Anpassungskonstruktion wurde der Stand der Forschung hinsichtlich der Automatisierung in Abschnitt 2.3 analysiert. Da der Begriff Anpassungskonstruktion selten explizit genannt wird, wurden

Beiträge betrachtet, die Merkmale sowie deren Ausprägungen als Ein- oder Ausgangsgrößen (Zielgrößen) berücksichtigen. Hieraus lässt sich bereits ableiten, dass die Anpassungskonstruktion bisher nicht allgemein für die Automatisierung betrachtet wurde, sondern nur spezifische Anwendungsfälle, die einer Anpassungskonstruktion entsprechen würden. Dieser starke Bezug auf spezifische Anwendungsfälle wurde im vergangen Jahrzehnt mehrfach kritisch analysiert und als Ursache für die mangelhafte Übertragbarkeit und damit schwierige Verbreitung von Automatisierungswerkzeugen identifiziert [3, 15, 65]. Die Grundlagen für diese Werkzeuge aus der Rechnerunterstützen Konstruktion, der (Struktur-)Optimierung, der Wissensbasierten Konstruktion und des Maschinellen Lernens wurden in Abschnitt 2.3 vorgestellt. Diese sind nicht streng voneinander getrennt zu betrachten, sondern bilden die Basis für verschiedene Automatisierungswerkzeuge. Beispielsweise kann ein Maschineller Lernprozess Versuchsdaten in einem Vorhersagemodell verarbeiten, um Fertigungsparameter abzuschätzen. Das trainierte Vorhersagemodell kann als zusätzliches Wissen einem Wissensbasierten System zur Verfügung gestellt werden. Diese datengetriebenen Ansätze können auch dann eingesetzt werden, wenn die Beziehungen, beispielsweise Konstruktionsregeln, zu aufwendig zu modellieren oder vorab unbekannt sind. Daher wurde diese Möglichkeiten des Maschinellen Lernens am Lehrstuhl für Konstruktionstechnik der Friedrich-Alexander Universität Erlangen-Nürnberg an verschiedenen Anwendungsszenarien untersucht und dabei konnten große Potentiale unter anderem für die Vorhersage von Beschichtungsparameter, Fertigungsparameter in der Blechmassivumformung [12] und der Herstellung von Clinchverbindungen [214] sowie Merkmalen von E-Motoren [P3] gezeigt werden. Vorhersagemodelle schätzen dabei den Versuchsraum zwischen den Stichproben unter Angabe einer Unsicherheit durch Antwortflächen ab.

Bei datengetriebenen Ansätzen muss jedoch berücksichtigt werden, dass der Anzahl der gleichzeitig betrachteten Variablen eine ausreichend große Stichproben bereit gestellt werden kann. Da anderenfalls anzunehmen ist, dass die Unsicherheit der Vorhersagen stark steigt (vgl. Unterabschnitt 2.3.7 auf Seite 47) [94]. Vor allem bei kompliziert Produkten mit einer großen Anzahl abhängiger und unabhängiger Konstruktionsvariablen wird eine große Anzahl von Stichproben benötigt [21]. Diese eingeschränkte Skalierbarkeit limitiert den Einsatz von maschinell-lernenden Ansätzen in der Produktentwicklung [215]. Insbesondere wenn Deep-Learning-Architekturen erforderlich sind, um die Zusammenhänge zwischen den Konstruktionsvariablen abschätzen zu können [25, 26, 216]. Darüber ist die Beschaffung der benötigten Daten in der Praxis häufig schwer möglich [27].

Dem gegenüber steht das Selbstverstärkende Lernen. Hier wird nicht auf Grundlage von Daten gelernt, sondern von Erfahrungen, die während des

Trainings gemacht werden. Die Potentiale des Selbstverstärkenden Lernens konnten bereits in [P5] gezeigt werden. Der Aufbau eines Selbstverstärkenden Lernprozess gilt jedoch im Vergleich zu herkömmlichen überwachten und nicht-überwachten Lernprozessen als schwierig, da neben den Hyperparametern des Modells zusätzlich die Explorations-Strategie und die Synchronisation der tiefen Neuronalen Netze vor dem Training festgelegt werden muss. Außerdem muss beachtet werden, dass die gesammelten Erfahrungen des Agenten mit der Umgebung Einfluss auf zukünftige Entscheidungen, welche wiederum weitere Erfahrungen generieren, haben. Folglich kann das Lernverhalten leichter instabil werden, als bei einem festen Datensatz. Die Anzahl der Veröffentlichungen des Selbstverstärkenden Lernens in der Produktentwicklung ist sehr gering. Bisherige Ansätze berücksichtigen geometrische Merkmale höchstens in mathematischen Gleichungen zur analytischen Lösung. Ein Ansatz für reale Produkte fehlt bisher.

3.2 Ableitung des Forschungsbedarfs

Der Forschungsbedarf dieser Arbeit motiviert sich aus der unzureichenden Übertragbarkeit der gelernten Lösungsstrategie auf ähnliche Anwendungsfälle und Skalierbarkeit auf komplexere Produkte von Automatisierungswerkzeugen im Rahmen der Anpassungskonstruktion. Insbesondere wurde folgende Herausforderungen identifiziert:

- Die Forschungslandschaft wird von Anwendungsfall-spezifischen Ansätzen für die automatisierte Anpassung einzelner Produkte beherrscht. Nur in wenigen Fällen ist der Entwicklungsprozess protokolliert und damit reproduzierbar. In sehr seltenen Fällen wird ein methodisches Vorgehen wiederverwendet. Folglich muss meist der komplette Entwicklungsprozess des Automatisierungswerkzeugs für einen neuen Anwendungsfall erneut durchlaufen werden. Es fehlt an einem gemeinsamen, generellen Verständnis des Anpassungsprozesses.
- Datengetriebene Ansätze können genutzt werden, wenn die Zusammenhänge zwischen Ein- und Ausgangsgrößen nicht beschrieben werden können oder der Aufwand einer manuellen Modellierung zu hoch ist. Dabei gehen die meisten Ansätze von einem ausreichend großen Datenbestand aus. Dieser muss in der Praxis nicht zwangsläufig vorliegen. Besonders kritisch ist der Fall eines komplexen Produktes mit vielen Merkmalen und einer kurzen Produkthistorie. Diese Fälle führen zu einem sehr spärlich besetzten Lösungsraum mit einer geringen Chance ein gutes Vorhersagemodell zu trainieren. Für diese Fälle fehlt es an einem daten-unabhängigen Werkzeug.

Aus den genannten Aspekten lässt sich zusammenfassend die wissenschaftliche Zielsetzung dieser Arbeit ableiten:

Das wissenschaftliche Ziel dieser Arbeit ist ein Konzept zum Einsatz des Selbstverstärkenden Lernens zur Abschätzung von Merkmalsausprägungen, insbesondere um eine Automatisierung der Anpassungskonstruktion durch Maschinelles Lernen zu ermöglichen, obwohl kein ausreichende Produkthistorie vorliegt. Außerdem ist es notwendig dieses Konzept in ein allgemeines Modell der Anpassungskonstruktion einzupassen und nach einem methodischen Vorgehen aufzubauen.

3.3 Ableitung der Forschungsfragen

Zur Operationalisierung der wissenschaftlichen Zielsetzung dieser Arbeit werden im Folgenden drei übergeordnete Forschungsfragen aufgestellt:

1. Es fehlt ein rechner-verarbeitbares Modell der Anpassungskonstruktion, das eine Anwendung von anderen Automatisierungswerkzeugen ermöglicht.
   Wie muss die Anpassungskonstruktion und deren Konstruktionsaufgaben modelliert werden, um als rechner-verarbeitbares Modell die Automatisierung zu ermöglichen?
2. Selbstverstärkendes Lernen ist innovativ und anspruchsvoll. Es gibt kein Konzept für die Anwendung in der Produktentwicklung und daher auch keine Anleitung für Produktentwickelnde. Deren Einbindung ist zwangsläufig notwendig, da deren Expertenwissen in die Überführung von Konstruktionsaufgaben in Lernumgebungen einfließen muss.
   Wie können Produktentwickelnde die Automatisierung der Anpassungskonstruktion durch Selbstverstärkendes Lernen einsetzen und evaluieren?

   Diese Frage lässt sich wie folgt unterteilen:

   - Welche Stellgrößen müssen die Produktentwickelnden beim Einsatz des Selbstverstärkenden Lernens kennen und beachten?
   - Welche Indikatoren können herangezogen werden, um das Lernverhalten und den Aufwand des Selbstverstärkenden Lernens zu untersuchen?
3. Selbstverstärkendes Lernen ist in zweierlei Hinsicht aufwändig. Einerseits ist die Anzahl der Stellgrößen hoch und sie beeinflussen sich gegenseitig. Anderseits muss der Agent Erfahrungen mit der Lernumgebung machen. Diese können je nach Kolonisationsaufgabe kosten-intensiv sein. Daher ist der Einsatz von Selbstverstärkendem Lernen vorab abzuwägen.
   Welche Kriterien sollten Konstruktionsaufgaben erfüllen, damit eine Automatisierung durch Selbstverstärkendes Lernen gerechtfertigt ist?

4 Aufbau der Arbeit

Aus der ursprünglichen Idee die Anpassungskonstruktion durch datengetriebene Ansätze, wie beispielsweise überwachtes Lernen zu automatisieren, ist der Aufbau und die Struktur dieser Arbeit erhalten geblieben. Obwohl die Automatisierung im folgendem Konzept nicht mehr aus der Auswertung von Daten, sondern auf Erfahrungen beruht, definiert der *Cross Industry Standard for Data-Mining* (CRISP-DM) nach CHAPMAN et al. die folgenden Abschnitte dieser Arbeit [217]. Der CRISP-DM dient dabei als generisches Referenzmodell, welches auf das Selbstverstärkende Lernen angepasst wurde,

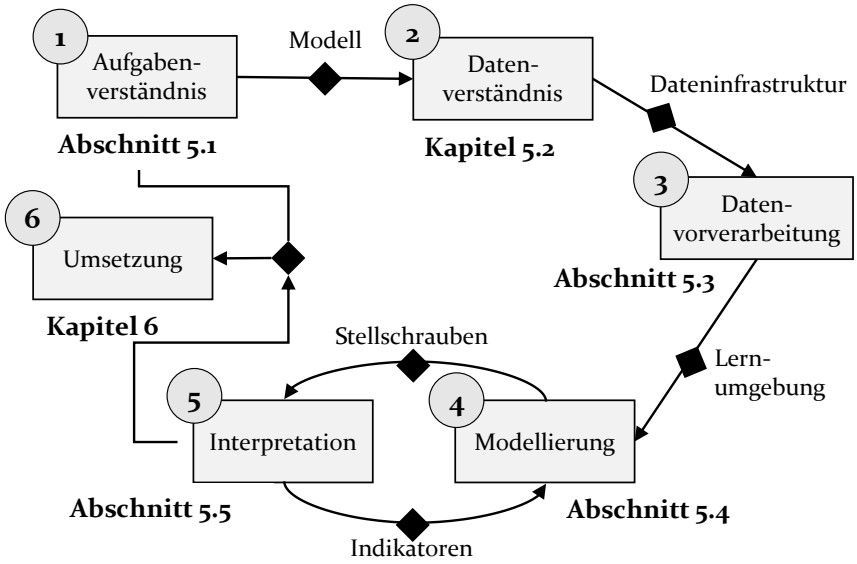


Bild 30: Aufbau dieser Arbeit in Anlehnung an den CRISP-DM [217]

① In der Phase des *Business Understanding* (Aufgaben- und Prozessverständnis) werden die Projektziele aus Perspektive der Anwendenden spezifiziert und IKT Sichtweisen überführt. Anwendende sind im folgenden Produktentwickelnden. Für das generische Vorgehen ist das Projektziel mit dem Ziel dieser Arbeit – die Anpassungskonstruktion zu automatisieren – identisch. Die erarbeitete Zusammenfassung der Anpassungskonstruktion in Abschnitt 2.2 (vgl. Seite 9) dient als Vorlage, die durch die Limitationen der Informations- und Kommunikationstechnologien (vgl. Unterabschnitt 2.3.2, Seite 23) in ein

rechner-verarbeitbares Modell in Abschnitt 5.1 übersetzt wird. Dieses soll die erste Forschungsfrage beantworten und bildet das erste Zwischenergebnis, das an die Phase des *Data Understandings* übergeben wird.

② Das *Data Understanding* (Datenverständnis) dient bei der Anwendung von daten-getrieben Werkzeugen zur Analyse der vorhanden Daten. Dabei wird beispielsweise geprüft, ob eine notwendige Zielgröße aus Sicht der Dateninfrastruktur zugänglich ist. Analog dazu wird beim Einsatz des Selbstverstärkenden Lernens geprüft, ob alle notwendigen Datenquellen über Schnittstellen (im Folgenden als API (*Application Programming Interface*) abgekürzt) zugänglich sind. In der Regel handelt es sich hierbei um Autorensysteme, die sich in Synthese (Anforderungen → Merkmalen, Merkmalsausprägungen) oder Analyse (Merkmalen, Merkmalsausprägungen → Eigenschaften) einordnen lassen. Ein System zur Synthese ist beispielsweise das CAD-System, da es eingesetzt wird, um Merkmale und deren Ausprägungen in einem virtuellen Produktmodell zu definieren. Ein FEM System wird beispielsweise zur Analyse eingesetzt, da es die Eigenschaften des virtuellen Produktmodells durch Lastfälle abschätzen kann. Diese Phase wird allgemein durch den Aufbau des Konzepts für das Selbstverstärkende Lernen zur Automatisierung der Anpassungskonstruktion erfüllt (Abschnitt 5.2). Allerdings ist diese Phase spezifisch für jeden Anwendungsfall und wird in Kapitel 6 noch einmal explizit aufgegriffen.

③ In der *Data Preparation* (Datenvorverarbeitung, in Abschnitt 5.3) wird die Lernumgebung des Selbstverstärkenden Lernens aus dem Modell einer Konstruktionsaufgabe aufgebaut. Diese interagiert mit dem Agenten durch Zustände, Aktionen, und Belohnungen. Produktentwickelnden müssen in diese drei Aspekte ihr Expertenwissen bezüglich der Konstruktionsaufgabe einbringen, da diese drei die Erfahrungen des Agenten mit Konstruktionsaufgaben widerspiegeln. Die Gesamtheit der gesammelten Erfahrungen ergibt die Historie des Lerners. Diese kann als Datentabelle interpretiert werden. Daher kommt das Erstellen der Lernumgebung der klassischen *Data Preparation* gleich. Das Ergebnis ist die Konstruktionsaufgabe als Lernumgebung für den Agenten.

④ & ⑤ Das *Modeling* (Modellierung) und die *Evaluation* (Interpretation und Auswertung) sind zwei sehr eng verknüpfte, iterative Phasen. Die Modellierung wird in Abschnitt 5.4 beschreiben und beinhaltet das Aufsetzen des Selbstverstärkenden Lerners auf die Lernumgebung. Dies dient zur Identifikation und Zusammenfassung der Stellschrauben für Produktentwickelnde (vgl. Forschungsfrage 2.1). Die Evaluation hingegen befasst sich mit der Bewertung des Lernverhaltens. Abschnitt 5.5 erarbeitet Zielgrößen, die den zweiten Teil der Forschungsfrage 2 beantworten sollen.

⑥ Die Phase des *Deployment* (Umsetzung) richtet sich an die Umsetzung des Konzeptes in Kapitel 6 und an dessen konkreten Einsatz. Konzept und Umsetzung werden anschließend genutzt, um die Machbarkeit, den Aufwand und die Übertragbarkeit des Selbstverstärkenden Lernens (vgl. Kapitel 7) zu untersuchen. Hierzu werden drei Anwendungsfälle vorgestellt, die sich an die Anpassung von Fahrradkomponenten an individuelle, körperliche Voraussetzungen richten. An diesen werden drei Untersuchungen durchgeführt.

1. Selbstverstärkendes Lernen wird eingesetzt, um unabhängig vom Ausgangszustand einen gewünschten Zielzustand (Merkmalskombination) zu finden, der die Anforderungen der Konstruktionsaufgabe erfüllt. Dabei wird das Lernverhalten und der Trainingsaufwand beobachtet. Hierbei werden die Indikatoren, die den Produktentwickelnden durch das Konzept bereit gestellt werden, erörtert, um deren Evaluationsmöglichkeiten darzustellen. Für die Untersuchung wird ein Konfiguration der Hyperparameter als Benchmark definiert.
2. Ausgehend von diesem Benchmark wird der Einfluss der Hyperparameter untersucht, da diese die Stellschrauben der Produktentwickelnden darstellen und deren Einfluss auf das Lernverhalten und den Trainingsaufwand analysiert werden soll. Um dabei die gegenseitigen Einflüsse der Hyperparameter zu berücksichtigen, werden Gruppen hinsichtlich der Aufgabe der Hyperparameter gebildet. Beispielsweise werden die Hyperparameter der Exploration-Strategie gemeinsam untersucht.
3. Ein elementarer Vorteil gegenüber anderen Automatisierungswerkzeugen ergibt sich aus der Möglichkeit der Übertragbarkeit eines bereits angelernten Agenten auf eine nachfolgende, ähnliche Konstruktionsaufgabe. Dies ist im Bezug der Anpassungskonstruktion besonders wichtig, da davon auszugehen ist, dass nachfolgende Produktgenerationen durch weitere Anpassungskonstruktionen realisiert werden. Die dritte Untersuchung zeigt den Einfluss auf das Lernverhalten und den Trainingsaufwand beim Einsatz eines vortrainierten Agenten im Vergleich zu einem nicht vortrainierten Agenten, bei deren Anwendung auf eine nachfolgende Anpassungskonstruktion.

Die gefunden Erkenntnisse werden in Kapitel 8 diskutiert und mit anderen Automatisierungswerkzeugen der Forschungslandschaft aus Unterabschnitt 2.3.3 verglichen, um Forschungsfrage 3 zu beantworten. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick in Kapitel 9 der die gefunden Einschränkungen aufgreift und nächste Schritte für das Selbstverstärkende Lernen in der Produktentwicklung aufzeigt.

5 Das Konzept des Selbstverstärkenden Lernens zur Automatisierung der Anpassungskonstruktion

Dieses Kapitel beschreibt das Konzept des Selbstverstärkenden Lernens zur Automatisierung der Anpassungskonstruktion. Das Vorgehen ist nach dem CRISP-DM (vgl. Kapitel 4 S. 65) strukturiert. Die folgenden Abschnitte erarbeiten die Zwischenergebnisse des Konzepts. Abschnitt 5.1 konzipiert ein computer-verarbeitbares Modell für die Anpassungskonstruktion. Abschnitt 5.2 stellt den Aufbau des Selbstverstärkenen Lernens für die Anpassungskonstruktion vor und definiert die Anforderungen an Schnittstellen zu den Autorensystemen, die als Synthese oder Analyse Werkzeug verwendet werden sollen. Abschnitt 5.3 betrachtet das Vorgehen, welches notwendig ist, die im Modell definierten Konstruktionsaufgaben in Lernumgebungen umzusetzen. Abschnitt 5.4 beschreibt die Details des Agenten und identifiziert die Stellschrauben anhand derer Produktentwickelnde Einfluss auf das Lernverhalten nehmen können. Um das Lernverhalten spezifisch für die Anpassungskonstruktion beurteilen zu können, stellt Abschnitt 5.5 Indikatoren für das Lernverhalten vor. Abschließend wird in Abschnitt 5.6 das Konzept für die Übertragung auf nachfolgende Produktgenerationen betrachtet.

5.1 Das Computer-verarbeitbare Modell der Anpassungskonstruktion

Dieser Abschnitt beschreibt eine Möglichkeit den Prozess der Anpassungskonstruktion zu modellieren. Ergebnis ist eine Wissensrepräsentation, die:

1. für Anwendende verständlich und leicht zu modellieren ist,
2. für den Computer verarbeitbar ist,
3. und in andere Wissensrepräsentationen integrierbar ist.

Vorlage ist die durch den Autor vorgestellte Wissensrepräsentation für die *Mass Customization* [P6]. Diese wird hier für die Anpassungskonstruktion angepasst. Ausgangspunkt und damit Zielsetzung ist die Analyse des Stands der Technik der Anpassungskonstruktion (vgl. Abschnitt 2.2 S. 19). Zunächst wird das Modell vorgestellt. Anschließend dessen Überführung in eine Wissensrepräsentation.

5.1.1 Das Modell der Anpassungskonstruktion

Das Modell beruht auf der Erkenntnis, dass die Aktivitäten des Anpassungsprozesses für die Automatisierung individuell betrachtet werden müssen. So kann jede Aktivität als Problem oder Aufgabe charakterisiert werden. Probleme setzen unter anderem kreatives Denken voraus. Dies kann nach dem momentanen Stand nicht bei der Automatisierung berücksichtigt werden. Da so ausschließlich Aufgaben automatisiert werden können, werden Aktivitäten im Folgenden als Konstruktionsaufgaben bezeichnet. Das Modell soll für jede einzelne Konstruktionsaufgabe einer Anpassungskonstruktion folgende Fragen beantworten können:

- Was beinhaltet die Anpassung?
- Warum wird die Anpassung vorgenommen?
- Wann wird die Anpassung vorgenommen?

Damit jede dieser Fragen beantwortet werden kann, beinhaltet das Modell eine hierarchische Achse und eine chronologische Achse.

Auf der chronologischen Achse wird die zeitliche Abfolge der Konstruktionsaufgaben repräsentiert. Dabei stehen die Konzepte Zustand und Transition im Vordergrund. Ein Zustand enthält die Ausprägungen für alle Merkmale eines Produktes zu einem definierten Zeitpunkt. Im Allgemeinen beschreibt eine Transition die Änderungen zwischen zwei Zuständen. Ein Zustand der Anpassungskonstruktion beschreibt die quantitative Gestalt (wie beispielsweise die geometrischen Ausprägungen, vgl. Bild 7 auf S. 15) des Produkts. Folglich sind die Merkmale aller Zustände identisch. Deren Ausprägungen ändern sich mit den Transitionen. Sie können geometrische, materialspezifi-

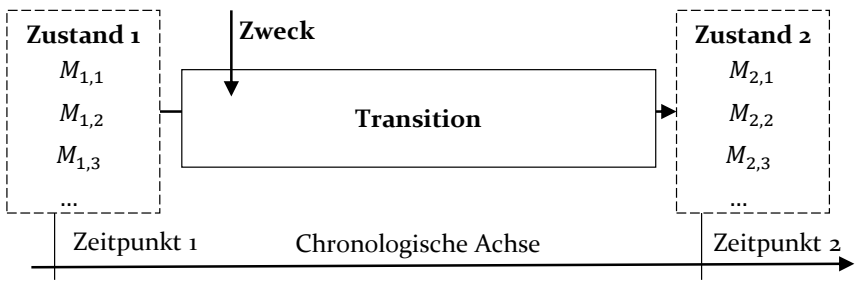


Bild 31: Chronologische Perspektive auf eine Transition und zwei Zustände

Eine Transition hat einen Zweck und kann entweder eine Konstruktionsaufgabe oder weiterer, untergeordneter Konstruktionsprozess sein. Der Zweck (im Englischen Design Rationale) soll die Frage nach dem Warum der Anpassung beantworten. Er lässt sich am besten als Container visualisieren, in dem eine theoretisch beliebige Anzahl Anforderungen abgelegt werden können. Die Anforderung wird als Zweck der Änderung verstanden und stellt dar, warum eine Änderung notwendig ist. So kann modelliert werden, dass sich eine Transition auf mehrere Anforderungen beziehen und gleichzeitig eine

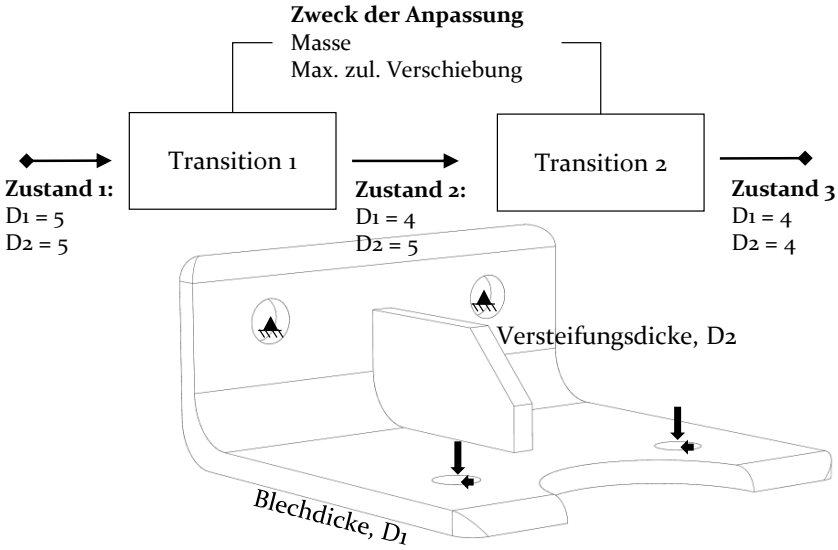


Bild 32: Ausschnitt aus dem exemplarischen Anpassungsprozess eines Flanschhalters unter Berücksichtigung der Masse und der maximalen, zulässigen Verschiebung

Bei der Anpassung eines Flanschhalters an einen neuen Einbaufall werden unter anderem Merkmale, wie die Blechdicke und die Dicke der Versteifung angepasst. Deren Ausprägungen sind auf den Kompromiss zwischen Leichtbau und Widerstand gegen die angreifenden Kräfte zurückführen (s. Bild 32). Die Anforderung sind folglich eine Restriktion der maximalen Masse und eine maximale, zulässige Verschiebung, um zu große Spannungen auf den Schweißnähten der Rohrleitung, die der Flanschhalter im eingebauten Zustand befestigen soll, zu vermeiden. Der Halter wird aus einem zugekauften Blech gebogen und anschließend wird die Versteifung eingeschweißt. Da in diesem fiktiven Beispiel die Blechdicken als Zukaufteil in mm-Schritten vorliegen, bilden sie den Flaschenhals der Anpassung und werden zuerst

dimensioniert. Anschließend werden die Eigenschaften in einer Simulation abgeschätzt und wenn notwendig die Versteifung dimensioniert. Auf der hierarchischen Achse werden die Beziehungen zwischen Anforderungen, Merkmalen und Eigenschaften modelliert. Sie wird in mehreren Leveln modelliert. Auf dem höchsten Level gibt es genau eine Transition zwischen dem Ausgangszustand und dem Zielzustand der Anpassungskonstruktion. Als Zweck werden alle Anforderungen an diese eine Transition übergeben (s. Bild 33). Da in den meisten Fällen der Prozess der Anpassungskonstruktion mehr als eine Konstruktionsaufgabe beinhaltet, muss die Reihenfolge und die Zuordnung der Anforderungen zu den Konstruktionsaufgaben modelliert werden. Diese Zuordnung erfolgt über das Konzept des Konstruktionsprozesses. Jeder Konstruktionsprozess beinhaltet ein eigenes untergeordnetes Level (s. Bild 33). Er hat genau einen Container für Anforderungen als Eingang und eine beliebige Anzahl an Containern als Ausgänge. Die Reihenfolge der Ausgänge muss beachtet werden, da sie die Reihenfolge der untergeordneten Transitionen bestimmt. Für jeden Ausgang wird genau eine Transition auf dem untergeordneten Level definiert. Die Ausgangs Container enthalten die jeweiligen Anforderungen für die untergeordneten Transitionen. Der

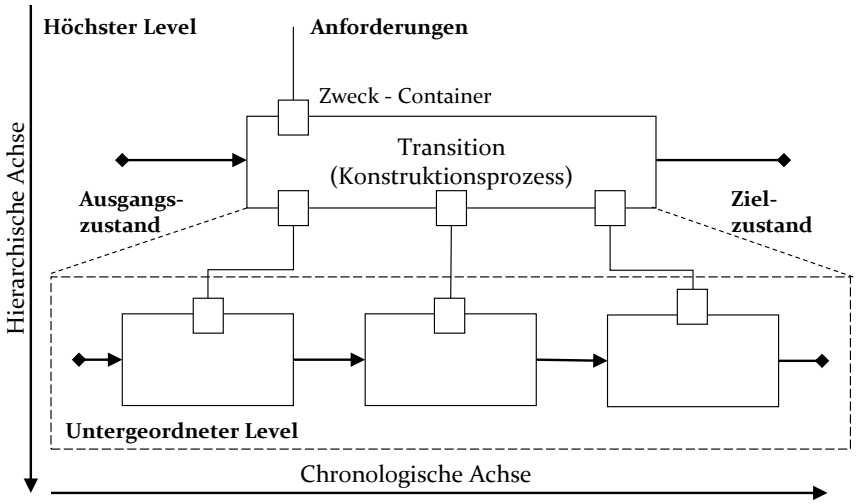


Bild 33: Konzept des Konstruktionsprozesses mit Darstellung des Prozesses der Anpassungskonstruktion (höchster Level) und einem untergeordneten Level mit drei Transitionen

Im Gegensatz zum Konzept des Konstruktionsprozesses modelliert die Konstruktionsaufgabe die Relation zwischen Merkmalen und Anforderungen (s. Bild 31). Folglich hat sie nur einen Container für Anforderungen als Eingang. Die Merkmale werden über die Zustände modelliert. Lediglich die Änderung der Merkmalsausprägung wird durch die Konstruktionsaufgabe repräsentiert. In Anlehnung an die Taxonomie der Automatisierung beinhaltet eine Konstruktionsaufgabe Handeln und Entscheiden. Handeln bedeutet für eine Konstruktionsaufgabe Merkmale zu synthetisieren oder zu analysieren. Entscheiden hingegen ist das Abgleichen der analysierten Eigenschaften mit den

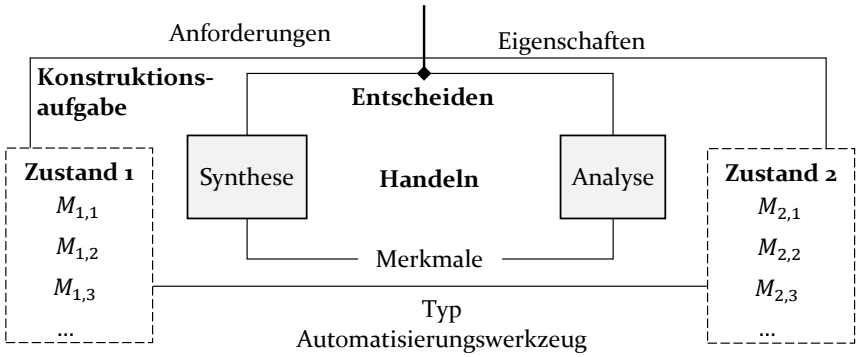


Bild 34: Konzept einer Konstruktionsaufgabe

Rückblickend für das Beispiel, wird ein Modell für die Anpassung des Flanschhalters erstellt. Dieses ist in Bild 35 visualisiert. Um vom Ausgangszustand den Zielzustand zu erreichen, muss mindestens eine Transition existieren. Deren Zweck sind die Anforderungen. Da diese Transition nicht gleichzeitig zwei Konstruktionsaufgaben darstellen kann, handelt es sich um einen Konstruktionsprozess. Dieser hat zwei Ausgänge. Beide beinhalten beide Anforderungen. Für jeden Ausgang wird auf dem untergeordneten Level in der definierten Reihenfolge eine weitere Transition angelegt. Beide sind Konstruktionsaufgaben, da sie Relationen zwischen Merkmalen und Anforderungen modellieren. Dieses Konzept der Konstruktionsaufgabe definiert, welche Merkmale, aus welchem Grund und zu welchem Zeitpunkt angepasst werden und beantwortet damit die drei Eingangs aufgestellten Fragen. Bisher wurde jedoch offen gelassen, wie die Anforderungen zu den Merkmalen in Relation gebracht werden. Hierfür hat die Konstruktionsaufgabe einen Typ und gegebenenfalls ein Automatisierungswerkzeug. Der Typ beschreibt den Automatisierungsgrad nach der Taxonomie nach ENDSLEY (vgl.Tabelle 2 auf Seite 23):

- Typ 0: eine manuelle Konstruktionsaufgabe benötigt zum Handeln und Entscheiden die Anwendenden – es gibt kein Automatisierungswerkzeug.
- Typ 1: das Automatisierungswerkzeug liefert zusätzliche Informationen – die Anwendenden Handeln und Entscheiden selbst.
- Typ 2: das Automatisierungswerkzeug handelt selbst und generiert eine Auswahl an Vorschlägen aus denen die Anwendenden wählen.
- Typ 3: das Automatisierungswerkzeug handelt und entscheidet eigenständig – die Anwendenden können eingreifen.
- Typ 4: das Automatisierungswerkzeug handelt und entscheidet eigenständig.

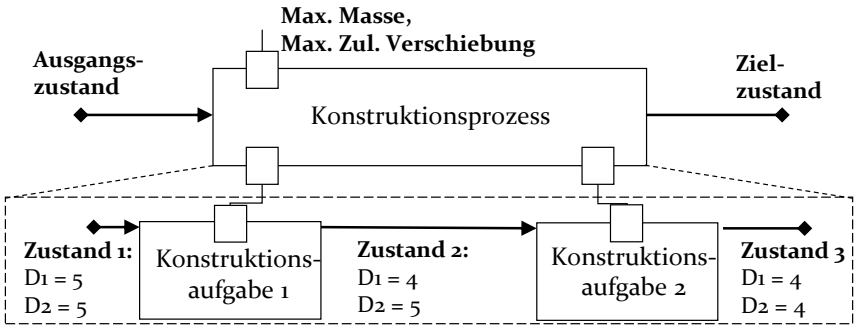


Bild 35: Modell der Anpassungskonstruktion zum Flanschhalterbeispiel

5.1.2 Die Wissensrepräsentation der Anpassungskonstruktion

Das in Abschnitt 5.1 vorgestellte Modell soll für die Anwendenden gut interpretierbar und einfach zu erstellen sein. Damit Automatisierungswerkzeug in die Konstruktionsaufgaben integriert werden können, muss es für den Computer verarbeitbar vorliegen. Es wird eine Wissensrepräsentation für die Domäne der Anpassungskonstruktion aufgebaut. Vorab sollte der Begriff Wissensrepräsentation in diesem Zusammenhang kurz reflektiert werden. Daten sind beispielsweise eine Zahl in Verbindung mit einer Einheit. Diese werden zu Informationen durch Semantik. Im Beispiel handelt es sich um ein Merkmal. Durch die Vernetzung der Informationen mit Hilfe des nachfolgend vorgestellten Konzeptes wird das Wissen über den Anpassungsprozess

repräsentiert. Prinzipiell können verschiedene Wissensrepräsentationen eingesetzt werden. In dieser Arbeit wird ein semantisches Netz mit einer stark limitierten Anzahl an Konzepten verwendet, damit es in andere semantische Netze integrierbar ist. Da bisher kein standardisiertes Vokabular für die Domäne der Produktentwicklung existiert, orientieren sich die Konzepte an den in Abschnitt 2.2 gefunden Begriffen. Über diese Konzepte kann die Wissensrepräsentation der Anpassungskonstruktion in andere semantische Netze integriert werden. Die zwei Konzepte für Zustand und Transition werden durch Axiome detailliert. Sie bilden die Terminologie dieser Wissensrepräsentation.

Tabelle 4 zeigt die Axiome für die Erweiterung des Zustands, welcher aus einer beliebigen Menge an Merkmalen besteht, um das Produkt zu jedem Zeitpunkt abbilden zu können. Einen eindeutigen Namen und damit einen Zeitpunkt erhält der Zustand erst, wenn er instanziiert wird. Jedes Merkmal besteht aus einem Wert und einer Einheit. Des weiteren werden Zustände in Ausgangs- und Zielzustände unterschieden. Dabei kann ein Ausgangszustand nicht der Zielzustand sein und umgekehrt. Diese dienen als eindeutige Start- bzw. Endpunkte des Anpassungsprozesses. Schließlich gibt es noch Konzepte für den Anfangs- und Endzustand einer Transition. Über sie wird die Reihenfolge modelliert.

Tabelle 4: Erarbeitete Axiome des Zustands aus [P6]:

Erweiternde Konzepte des Zustands	**Axiome**
Zustand	hat $n \geq 1 \in \mathbb{N}$ Merkmale.
Ausgangszustand	ist ein Zustand; disjunkt Zielzustand.
Zielzustand	ist ein Zustand; disjunkt Ausgangszustand.
Anfangszustand	ist ein Zustand; disjunkt Zielzustand.
Endzustand	ist ein Zustand; disjunkt Ausgangszustand.
Merkmal	hat einen Wert; hat eine Einheit.

Tabelle 5 zeigt die Axiome für die Erweiterung der Transition. Jeder Transition hat einen Anfangszustand und einen Endzustand, welche die Reihenfolge modellieren. Des weiteren hat jede Transition mindestens einen Eingang und gegebenenfalls Ausgänge für Anforderungen, um die Hierarchie repräsentieren zu können. Es wird wiederum der Zweck, als Container für eine beliebige Anzahl an Anforderungen verwendet. Ein Zweck besteht aus mindestens einer Anforderung, welche wiederum einen Wert und eine Einheit hat. Je nach dem ob eine Transition Ausgänge hat oder nicht ist sie entweder ein Konstruktionsprozess, der einen untergeordneten Prozess mit definierter Reihenfolge modelliert oder eine Konstruktionsaufgabe. Die Konstruktionsaufgabe hat

einen Typ für den Automatisierungsgrad und ein Automatisierungswerkzeug. Dieses wird weiter durch die Repräsentation der Merkmale, Zwecke sowie einer Applikation definiert. Applikation ist hier als Platzhalter für jede Art Automatisierungswerkzeug, das Anforderungen als Eingangsgrößen hat, Merkmale als Ausgangsgrößen hat und deren Relationen bestimmen kann, gedacht. Diese Lücke wird im folgenden mit Hilfe des Selbstverstärkenden Lernens geschlossen, dennoch ist es möglich jede andere Art hier einzusetzen. Dies ermöglicht den Einsatz des Modells und der Wissensrepräsentation für Mehrzieloptimierung, Wissensbasierte Systeme oder datengetriebene Ansätze des Maschinelles Lernen.

Tabelle 5: Erarbeitete Axiome der Transition aus [P6]:

Erweiternde Konzepte der Transition	**Axiom**
Transition	hat einen Anfangszustand; hat einen Endzustand; hat $r_{in} \geq 1 \in \mathbb{N}$ Zweckeingänge; hat $r_{out} \geq 0 \in \mathbb{N}_0$ Zweckausgänge.
Zweck	hat $r \geq 1 \in \mathbb{N}$ Anforderungen.
Zweckeingang	ist ein Zweck.
Zweckausgang	ist ein Zweck.
Anforderung	hat einen Wert; hat eine Einheit.
Konstruktionsprozess	ist eine Transition; hat $r_{out} > 1$ Output Rationale.
Konstruktionsaufgabe	ist eine Transition; hat $r_{out} = 0$ Output Rationale; hat einen Typ; hat ein Automatisierungswerkzeug
Automatisierungswerkzeug	hat einen Typ; hat $\hat{r} \geq 1 \in \mathbb{N}$ Zwecke; hat $\hat{n} \geq 1 \in \mathbb{N}$ Merkmale; hat eine Applikation.

5.1.3 Zusammenfassung des computer-verarbeitbaren Modells der Anpassungskonstruktion

Damit digitale Technologien zur Automatisierung des Handelns und Entscheidens eingesetzt werden können, müssen alle relevanten Daten, Information und Wissen in einer computer-verarbeitenbaren Form vorliegen. Da es in den meisten Fällen nicht möglich ist, den Anpassungsprozess durch

ein Werkzeug zu automatisieren, wurde hier ein Modell vorgestellt, dass den Prozess der Anpassungskonstruktion modelliert und in Konstruktionsaufgaben unterteilt. Das Modell wurde so konzipiert, das zu jeder Konstruktionsaufgabe Aussagen über die Anpassung, dessen Begründung und dessen Zeitpunkt getroffen werden können. Das Modell wurde in ein semantisches Netz als Wissensrepräsentation überführt. Dies ermöglicht einerseits die computer-verarbeitenbare Abfrage des Modells. Andererseits kann die Wissensrepräsentation so mit Hilfe von Zuweisungsvorgaben (im Englischen Mapping Schema, siehe Ontology Alignment in [P6]) an andere semantische Netze gekoppelt werden. Im folgenden wird das Selbstverstärkende Lernen zur Automatisierung einer Konstruktionsaufgabe vorgestellt.

5.2 Der Aufbau des Selbstverstärkenden Lernprozesses für die Anpassungskonstruktion

Dieser Abschnitt beschreibt das strukturelle Konzept des Selbstverstärkenden Lernens als Automatisierungswerkzeug für die Anpassungskonstruktion und erfüllt damit die Phase des *Data Understandings* des CRISP-DM. Es wird unterschieden in die internen Interaktionen der Komponenten und den externen Interaktionen mit den Anwendenden. Dazu wird in Unterabschnitt 5.2.1 zunächst der modulare Aufbau der Komponenten erklärt werden. Dieser ist zur besseren Orientierung bereits vorab in Bild 36 dargestellt. Dort ist links der Aufbau und rechts die Schnittstellen zu den Anwendenden dargestellt.

Der Selbstverstärkende Lernprozess enthält Stellschrauben die spezifisch für den Anwendungsfall eingestellt werden müssen. Dies obliegt den Produktentwickelnden die Selbstverstärkendes Lernen als Automatisierungswerkzeug einsetzen möchten, weil keine Datengrundlage für einen Maschinellen Lernprozess existiert oder, wie am Ende dieses Kapitels in Abschnitt 5.6 gezeigt, gelernte Zusammenhänge für nachfolgende Produkt wiederverwendet werden sollen. Bevor in den folgenden Abschnitten jeder Punkt der Anwender-Interaktion beleuchtet wird, wird vorab in Unterabschnitt 5.2.2 deren Relevanz für den gesamten Lernprozess erläutert.

5.2.1 Die Komponenten und ihr Zusammenspiel

Bild 36 zeigt, dass sich der Kern des Automatisierungswerkzeugs an den durch SUTTON & BARTO vorgestellten strukturellen Aufbau, welcher die strenge Unterteilung in einen Agenten und eine Lernumgebung vorsieht, orientiert [32]. Dieser wird für die Automatisierung der Anpassungskonstruktion von einem eigenen Konzept umfasst. Im Kern dieser Arbeit steht der Umgang von Produktentwickelnden mit dem Selbstverstärkenden Lernen. Um die Fakto-

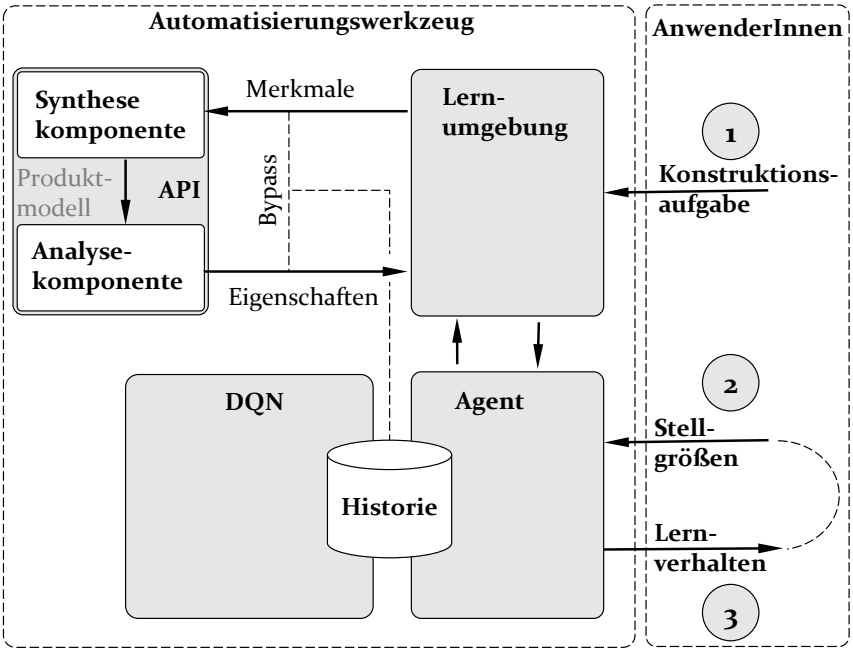


Bild 36: Darstellung des strukturellen Aufbaus der Selbstverstärkenden Lernens als Automatisierungswerkzeug aus Sicht der Anwendenden

ren, die Produktentwickelnde beeinflussen können und sollen zu verstehen, muss vorab das Konzept erläutert werden.

Die **Lernumgebung** beschreibt die Konstruktionsaufgabe, die der Agent erlernen soll. Dazu zählt der abgegrenzte Lösungsraum, die Anforderungen an das angepasste Produkt, gegebenenfalls deren Präferenzen, sowie weitere Randbedingungen, die der Agent berücksichtigen soll. Der Lösungsraum definiert, die Merkmalsausprägungen, die für die Anpassung in Betracht gezogen werden sollen. Für geometrische Merkmale wird er meist durch den Bauraum, die Wirkflächen oder andere Randbedingungen vorgegeben. Ein Hypothetischer Lösungsraum ist in Bild 37 dargestellt. Damit jede Instanz des Lösungsraums - jedes mögliche Produkt - modelliert und hinsichtlich der Eigenschaften analysiert werden kann, wird die Lernumgebung durch die **API** Komponente erweitert. Dieses steuert die Synthese- (beispielsweise eine CAD-Umgebung) und die Analysekomponente (beispielsweise eine FEM-Umgebung). Beide sind nicht Teil des Werkzeugs, sondern werden nur von diesem gesteuert. Sie erhalten die Merkmale und Anweisungen, setzen diese in einem Produktmodell um und geben die Eigenschaften an die Lernumgebung zurück. Der Weg über die API kann über den Bypass abgekürzt werden, um zu verhindern, dass bereits bekannte Merkmalskombinationen

erneut simuliert werden. Bevor die Merkmale an die API übergeben werden, werden sie mit der Historie, welche unter anderem alle bisherigen Merkmalskombinationen enthält, verglichen. Nur neue Merkmalskombinationen müssen neu berechnet werden. Für bereits Bekannte können die Erfahrungen aus der Historie verwendet werden. Die resultierenden, abgeschätzten, einzelnen Eigenschaften werden in der Lernumgebung den Anforderungen gegenübergestellt. Die Lernumgebung spiegelt die Erfüllung der Anforderungen in individuellen Belohnungen (*r* für *reward*) wider. Die genaue

ehr-
hen
iese
ıme
gen

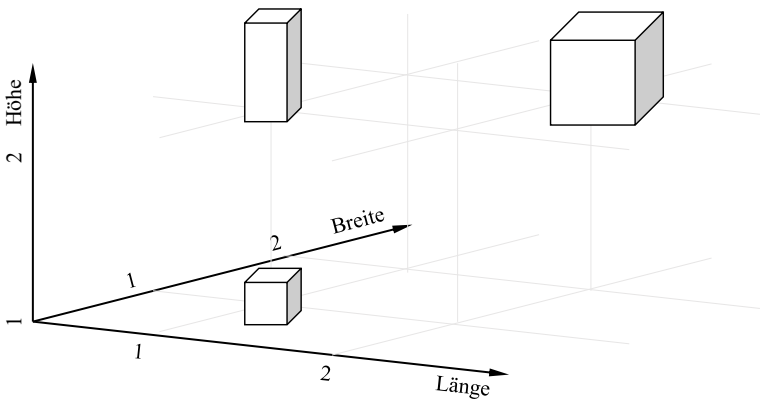


Bild 37: Darstellung eines hypothetischen Lösungsraums für drei Merkmale und drei exemplarische Zustände

Der **Agent** ist die ausführende Komponente. Sein Ziel ist es, für die Konstruktionsaufgabe der Lernumgebung eine Strategie zu finden, die vom unerwünschten Ausgangszustand zum gewünschten Zielzustand führt. Eine Strategie ist eine Folge von Zuständen (*s* für *state*) und Aktionen (*a* für *action*). Analog zum in Abschnitt 5.1 vorgestellten Modell beschreibt ein Zustand alle Merkmale zu einem definierten Zeitpunkt t. So ist s_t der Zustand zum Zeitpunkt t. Eine Aktion beschreibt alle Änderungen, die der Agent zum Zeitpunkt *t* vorschlägt. In dieser Arbeit kann der Agent jedes Merkmal individuell vergrößern, verkleinern oder beibehalten. Theoretisch können jedoch beliebige Änderungsmöglichkeiten als Regeln definiert werden, auch solche, bei denen sich die Merkmale gegenseitig beeinflussen, wie beispielsweise Stufensprünge. Eine Strategie lässt sich visualisieren in dem die Folge der

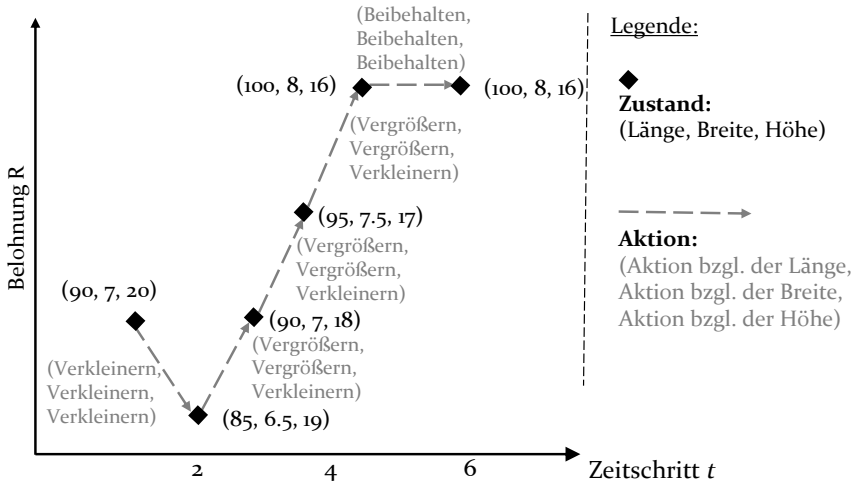


Bild 38: Darstellung einer hypothetischen Strategie als Folge von Zuständen und Aktionen über 6 Zeitschritte

Eine Strategie ist dann besonders gut, wenn mit möglichst geringem Aufwand eine möglichst hohe Belohnung erzielt wird. Der Aufwand entsteht im Wesentlichen[1] aus dem Sammeln von Erfahrungen mit der Lernumgebung. Eine Erfahrung setzt sich zusammen aus dem aktuellen Zustand (s_t) der gewählten Aktion (a_t), der erhaltenen Belohnung (r_t), sowie dem dadurch erreichten Zustand ($s_t + 1$)[2]. Erfahrungen werden generiert, indem der Agent mit der Lernumgebung interagiert. Dies geschieht in einer Schleifen für jeden Zeitschritt. Umgangssprachlich wird dieser Prozess als *Play Loop* bezeichnet, da der Agent die Umgebung bespielen darf. Er ist in Bild 39 dargestellt und enthält die fünf durchnummerierten Schritte.

In ① (Bild 39) sieht der Agent den Zustand der Lernumgebung s_t und wählt eine Aktion a_t. Diese kann zufällig (explorativ) oder unter Ausnutzung des DQNs (im Englischen *to exploit*) getroffen werden. Die Interaktion mit den DQN wird anschließend erläutert. Die Aktion a_t enthält für jedes Merkmal des Zustandes s_t eine Anweisung, ob das Merkmal zu vergrößern, zu verkleinern oder gleich zu lassen ist. Folglich wird in ② die Lernumgebung auf den Zustand s_{t+1} gebracht und dieser an die API übergeben. Die API in ③

[1] Dies wird anhand der Demonstratoren in Kapitel 7 verdeutlicht werden.

[2] s_{t+1} lässt sich aus s_t und a_t berechnen, jedoch wird hier die von MNIH et al. empfohlene Nomenklatur verwendet, welche die Berechnung der DQNs vereinfacht verwendet [32].

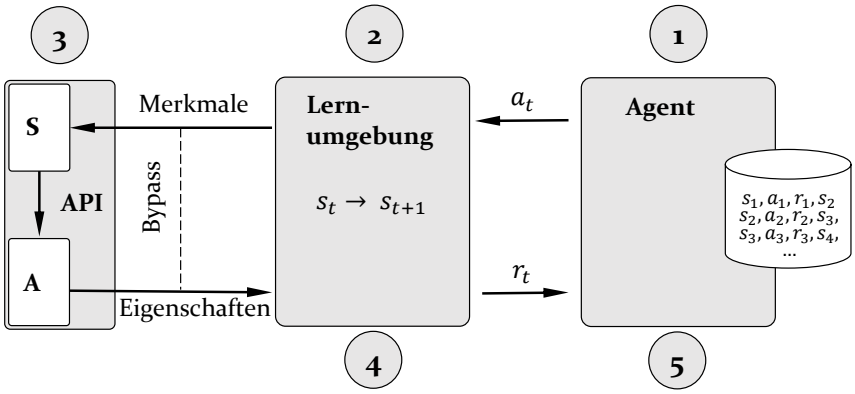


Bild 39: Darstellung der Schleife zur Generierung von Erfahrungs-Tuples (s_t, a_t, r_t, s_{t+1})

generiert mit den neuen Merkmalen ein Produktmodell, analysiert die Eigenschaften und sendet diese zurück an die Lernumgebung. Die Lernumgebung berechnet in ④ aus den Eigenschaften die individuellen und die gesamte Belohnung (r_t) für die Aktion a_t auf dem Zustand s_t. Das Tupel (s_t, a_t, r_t, s_{t+1}) wird in der Historie abgelegt ⑤ und der Agent beginnt die Schleife erneut mit dem Unterschied, dass sich die Lernumgebung nun in s_{t+1} befindet. Ein Exploit benutzt zur Bestimmung der nächsten Aktion die Vorhersagen eines Künstlichen Neuronalen Netzes. Diese werden im **DQN** Modul berechnet, in Anlehnung an den verwendeten Q-Learning Algorithmus nach MNIH et al. [200]. Dazu enthält das Modul zwei Tiefe, Künstliche Neuronale Netze mit identischer Architektur. Beide haben die gleiche Anzahl Schichten (N_S) und die gleiche Anzahl Knoten pro Schicht (N_K). Die Knoten sind vollständig verknüpft. Vorhersagen des Netzes werden im *Forward Feed* Modus getroffen. Es gibt keine Rückkopplung. Auf der Eingangsschicht hat das DQN ein Neuron für jedes Merkmal, das angepasst werden soll. Dementsprechend ist die Dimension der Eingangsschicht gleich der des Zustands-Vektors s_t. Auf der Ausgangsschicht hat das DQN ein Neuron für jede Kombination aus Merkmal und Aktion. Die Werte auf der Ausgangssicht sind die namensgebenden *Q-Values*. Einer dieser Werte gibt, an wie groß die Erwartung auf eine hohe Belohnung ist. Beim Ausnutzen des DQNs zum Vorschlagen einer Aktion (Exploit), wird immer die Aktion mit dem höchsten Q-Value gewählt. Wird beispielsweise ein rechteckiger Balken mit drei Merkmalen (Länge, Höhe, Breite) angepasst. Hat das DQN 3 Eingangsneuronen und 27 Ausgangsneuronen, weil jedes Merkmal vergrößert, verkleinert oder gleich gelassen werden kann. So trifft der erste Q-Value eine Ausgabe darüber, wie hoch die Erwartungen auf eine hohe Belohnung bei der Vergrößerung aller drei Merkmale

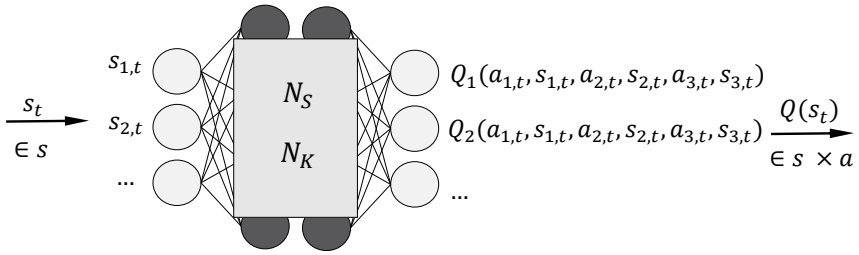


Bild 40: Darstellung des tiefen, künstlichen neuronalen Netzes zum Erlernen der Strategie

Dies führt zur entscheidenden Frage, wie der Agent besser werden kann - die Konstruktionsaufgabe erlernt. Dazu müssen die Gewichte der Tiefen, Künstlichen Neuronalen Netze so angepasst werden, dass für einen beliebigen Zustand s_t die bestmögliche Aktion den höchsten Q-Value erhält. Die Anpassung eines Neuronalen Netzes wird üblicherweise durch die Backpropagation eines Fehlers (engl. *Loss*) vorgenommen. Beim überwachten Lernen ergibt sich der Loss aus dem Abstand der Vorhersagen des Netzes zu den wahren Werten des Datensatzes. Beim Selbstverstärkenden Lernen existieren keine Daten, deren Werte für die Berechnung des Losses verwendet werden können. Daher wird über die Bellman Gleichung (vgl. Gleichung 9 auf Seite 53) der Loss mit Hilfe des bestmöglichen Q-Values des nächsten Zustandes approximiert. Die Anpassung ist die zweite Schleife des Konzepts, welche das Prinzip des Q-Learnings nach Mnih u. a. realisiert und in Bild 41 dargestellt ist [200].

Um die Netze anzupassen, wird ein Erfahrungs-Batch aus der Historie verwendet, indem eine definierte Anzahl an Erfahrungen zufällig ausgewählt wird. So wird der Einfluss einzelner Erfahrungen reduziert und eine Überanpassung auf diese erschwert. Zunächst wird der Loss berechnet (s. ① in Bild 41). Dazu werden die aktuellen Zustände (s_t) an das sogenannte *Evaluation DQN* und die nächsten Zustände s_{t+1} an das *Target DQN* übergeben. Zu Beginn sind beide Netze identisch. Die Vorhersagen unterscheiden sich jedoch, da unterschiedliche Zustände übergeben wurden. Aus der Differenz der Vorhersagen beider Netze und der erhaltenen Belohnungen (r_t) ergibt sich der Loss (vgl. Gleichung 11 auf Seite 54 oder in Bild 41). Dabei ist zu beachten, dass aus den Vorhersagen der nächsten Zustände das Maximum ($maxQ(s_{t+1})$) gewählt wird. Dieser Fehler wird in ② (Bild 41) für die Backprobpagation verwendet und das Evaluation DQN dementsprechend angepasst. Da zunächst nur das Evaluation DQN angepasst wird, driften die Gewichte des Evalutation DQN und des Target DQN auseinander. Daher wird in regelmäßigen Abständen

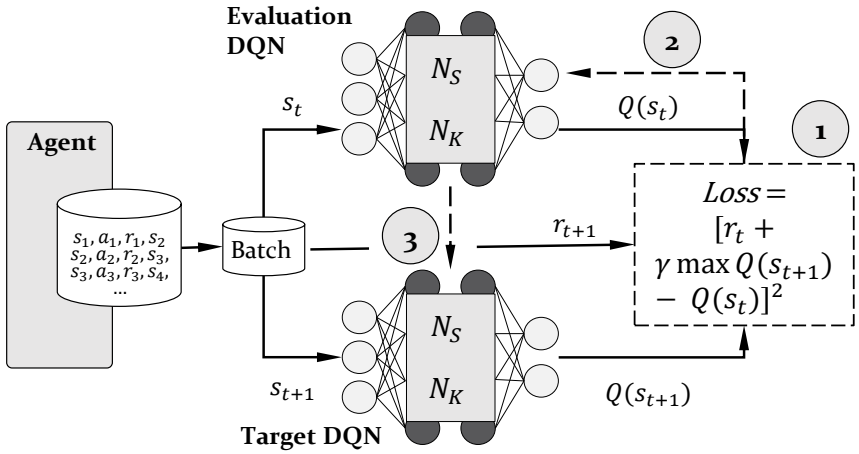


Bild 41: Der Lernprozess des Selbstverstärkenden Lernens auf Basis des Q-Learning Algorithmus nach MNIH et al. [200]

die Gewichte des Evaluation DQN in das Target DQN geklont ③ (Bild 41), um das Target DQN aktuell zu halten.

Der Fehler und damit die Netzanpassung wird folglich über bereits gemachte Erfahrungen determiniert. Daraus lässt sich schließen, dass der Selbstverstärkende Lerner eine Strategie zur besten Lösung finden wird, solange das Tiefe, Künstliche Neuronale Netz über die mathematischen Möglichkeiten verfügt den Lösungsraum zu approximieren, ausreichend viele, unterschiedliche Erfahrungen gemacht wurden und ausreichend lange trainiert wird. Der zweite Punkt unterstreicht die Relevanz des effizienten Trainings, da jede neue Erfahrung einen neuen Durchlauf über Analyse- und Synthese-Komponente und damit Mehraufwand bedeutet. Jede Konstruktionsaufgabe ist individuell. Daher muss der Selbstverstärkende Lerner, um den Lösungsraum entsprechend abbilden zu können, auf die Konstruktionsaufgabe durch die Anwendenden eingestellt werden. Die Relevanz der Nutzerinteraktion wird im nächsten Abschnitt noch einmal verdeutlicht bevor die Eingriffsmöglichkeiten in Abschnitt 5.4 und deren Indikatoren in Abschnitt 5.5 vorgestellt werden.

5.2.2 Die Relevanz der Nutzerinteraktion

Rückblickend auf Bild 36 auf Seite 78 haben Anwendende drei Interaktionsmöglichkeiten mit dem Selbstverstärkenden Lernprozess. ① beschreibt die Übergabe der Konstruktionsaufgabe an Lerner. Diese ist uni-direktional, d. h. die Eingaben gehen von den Anwendenden an den Selbstverstärkenden Lerner. Die Eingaben definieren die Konstruktionsaufgabe und werden in

Abschnitt 5.3 detailliert beschrieben. ② und ③ in Bild 36 beziehen sich auf das Lernverhalten des Agenten und den benötigten Trainingsaufwand. In ② werden Hyperparameter durch die Produktentwickelnde vorgegeben. Diese müssen vor dem Training definiert werden. Sie werden während des Trainings nicht angepasst und werden häufig als Hyperparameter bezeichnet. Bevor diese in Abschnitt 5.4 detailliert beschrieben werden, wird deren Relevanz für das Selbstverstärkende Lernen erläutert. Im Vergleich zu anderen datengetriebenen Ansätzen, wie beispielsweise dem überwachten Lernen mit einem Künstlichen Neuronalen Netz als Vorhersagemodell, haben die Hyperparameter einen direkten Einfluss auf die Generierung der Erfahrungen mit der Lernumgebung. Dieser Punkt ist kritisch und unterscheidet Selbstverstärkendes Lernen von anderen Lernverfahren. Beim überwachten Lernen führen schlecht eingestellte Hyperparameter mutmaßlich zu einer schlechten Prognosequalität. Der Datensatz bleibt davon jedoch unberührt. Beim Selbstverstärkenden Lernen führen schlecht eingestellte Hyperparameter zu unzureichenden Erfahrungen mit dem Lösungsraum. Diese Erfahrungen dienen als Grundlage für die Netzanpassung und folglich auch zur Auswahl der nächsten Aktionen, die wiederum dafür verantwortlich sind, welche Erfahrungen zukünftig gemacht werden können. Das bedeutet schlussendlich, dass bei unpassenden Hyperparametern Teile des Lösungsraums nicht vom Agenten berücksichtige werden und vermeintlich gute Produkte nicht in Betracht gezogen werden können. Dies soll nachfolgend anhand von einem Beispiel verdeutlicht werden.

Das Labyrinth als exemplarische Lernumgebung

Ein Labyrinth eignet sich zur Vergegenwärtigung von einfachen Lernaufgaben. Das Labyrinth stellt dabei die Lernumgebung. Ein Beispiel ist in Bild 42 dargestellt. Zustände in dieser Lernumgebung werden durch die Position des Agenten im Labyrinth (Schwarzer Punkt in Bild 42) ausgedrückt. Hier könnten beispielsweise X- und Y-Koordinaten zur Beschreibung des Zustandes verwendet werden. Der in Bild 42 gezeigte Zustand könnte eindeutig durch $S(X = 6, Y = 5)$ beschrieben werden. Alle möglichen Zustände bilden den Lösungsraum der Lernumgebung. Das in Bild 42 gezeigte Labyrinth umfasst eine Fläche von 80 Feldern. Davon sind 39 durch Mauern blockiert und für den Agenten nicht erreichbar. Damit hat der Lösungsraum, in dem sich der Agent bewegen kann 41 mögliche Zustände. Gewünschte Zustände sind versteckte Schätze im Labyrinth. Diese sind in den Ecken oben und unten links positioniert und werden durch +10 gekennzeichnet. Sie sollen vom Agenten erreicht werden. Im Gegenzug dazu sind Sackgassen mit -10 gekennzeichnet.

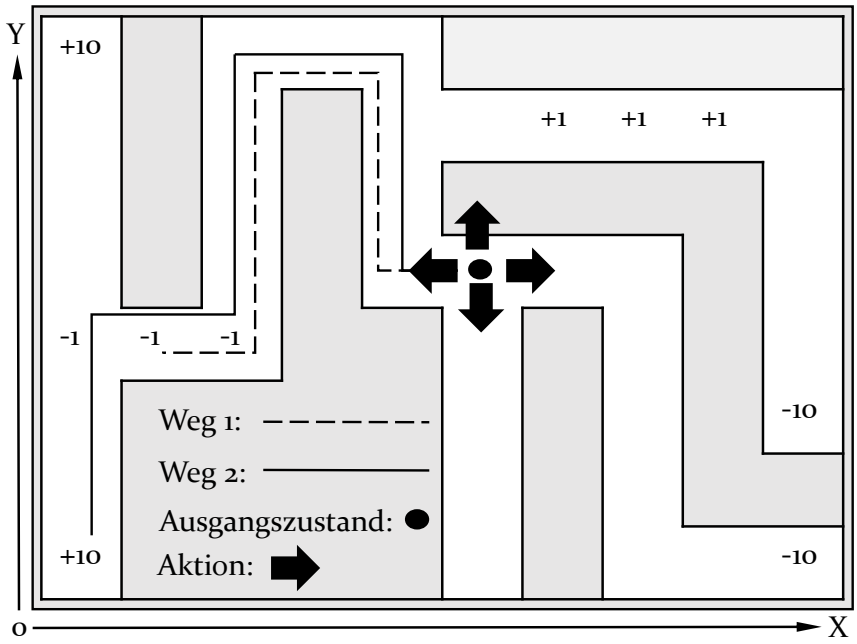


Bild 42: Darstellung des Lösungsraums als Labyrinth in dem Selbstverstärkende Lerner gute Zustände finden sollen

Der Agent bewegt sich schrittweise, d. h. er kann einmal pro Zeitschritt seine Position ändern. Die Änderung der Position ist die Aktion. Der Aktionsraum besteht aus den Möglichkeiten: hoch, runter, links und rechts. Nach einer Aktion erhält der Agent seine Belohnung für die Aktion. Diese sind in Bild 42 durch die Wert im Labyrinth gekennzeichnet. Läuft der Agent gegen eine Wand, wird die Aktion nicht ausgeführt und der Agent erhält die gleiche Belohnung erneut. Nach einer definierten Anzahl an Zeitschritten wird der Agent wieder in die Mitte – auf die Ausgangsposition zurückgesetzt, um zu verhindern, dass der Agent in einer Ecke stecken bleibt.

Das Ziel der Lernaufgabe ist, dass der Agent auf Basis seiner Postion lernt, welche Aktion er ausführen muss, um zu einem Schatz zu gelangen. Der Wert einer Aktion wird durch die Q-Values (s. Gleichung 8 auf Seite 53) ausgedrückt. Jeder Zustand hat vier Q-Values. Einen für jede Aktion, beispielsweise $Q(S(X = 6, Y = 5), Aktion = hoch)$. Folglich muss der Agent für die 39 möglichen Zustände jeweils vier Q-Values lernen. Theoretisch können diese Werte in einer Tabelle mit 156 Einträgen gesammelt werden. Da dies jedoch unübersichtlich ist, wird zu deren Abschätzung das DQN verwendet. Das DQN erhält auf den Eingangsneuronen den Zustand. Hier sind es folglich zwei Neuronen. Der Eingangszustand gelangt über die verdeckten Schichten

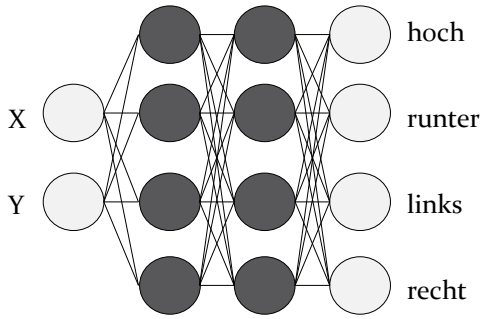


Bild 43: Darstellung eines DQNs mit zwei Eingangsneuronen für die X- und Y-Koordinate, zwei verdeckten Schichten mit vier Neuronen und vier Ausgangsneuronen für die vier Q-Values der Aktionen

Unter Lernen wird folglich das Anpassen der Gewichte des Neuronalen Netzes mit Hilfe von Gleichung 11 auf Seite 54 verstanden. Hierfür sind Erfahrungen mit der Lernumgebung notwendig. Sie bestehen aus dem momentanen Zustand, der ausgeführt Aktion, dem sich daraus ergebenden, nächsten Zustand und der dadurch erhaltenen Belohnung. Ein Beispiel für das Labyrinth ist: $S_t(X = 6, Y = 5)$, links, $S_{t+1}(X = 6, Y = 5)$, 0. Für die Anpassung der Gewicht werden Erfahrungen benötigt, die die Lernumgebung ausreichend gut charakterisieren. Beispielsweise weiß der Agent nicht, dass er sich auf dem Weg zum Schatz befindet, wenn er diesen noch nie erreicht hat. Im schlechtesten Fall erreicht der Agent das Feld vor dem Schatz und läuft dort solange gegen die Wand, bis er wieder auf die Startposition zurückgesetzt wird. Der Wert der Aktionen, die der Agent bis zum Feld vor dem Schatz ausgeführt hat, sollten eigentlich einen hohen Wert für den Agenten haben. Dieser wird nicht erfasst und daher nicht gelernt, weil die entscheidende Belohnung nicht erreicht wurde. Wichtig ist zu verstehen, dass die gesammelten Erfahrungen die Anpassung des DQNs bestimmen. Diese wiederum bestimmen die Erfahrungen, die der Agent in Zukunft machen wird. Am Beispiel des Labyrinthes lässt sich außerdem verdeutlichen, wie schlecht gewählt Hyperparameter dazuführen können, dass die Lernumgebung nicht ausstreichend gut erfasst werden kann. Hierzu sind in Bild 42 sind zwei Wege von der Ausgangsposition gekennzeichnet. Weg 1 führt nicht zu einer guten Postion, weil er sich durch die negativen Belohnungen der horizontalen Schritte abbringen lässt. Weg 2 führt über die negativen Belohnungen bis zu einer der gewünschten Postionen, in der der Agent +10 Punkte als Belohnung erhält. Entscheidend dafür ist unter anderem der Discount Faktor, der bestimmt, wie weitsichtig der Agent ist. Er ist einer der Hyperparameter, die von den Anwendenden vor

dem Training festgelegt werden muss. Ist der Agent nicht weitsichtig genug, kann es vorkommen, dass der Weg über die schlechten Zustände nicht akzeptiert wird und der dahinterliegende gewünschte Zustand nicht gefunden wird. Dies bedeutet, dass er nicht als Erfahrung in der Historie ist und daher auch nicht bei der Anpassung der DQNs berücksichtigt werden kann. Die gegenseitige Abhängigkeit von Lernverhalten und Erfahrungen ist ein Grund dafür, dass das Selbstverstärkendes Lernen vergleichsweise kompliziert und instabil sein kann. Daher ist es besonders wichtig, dass Anwendende die Stellgrößen (Hyperparameter) kennen, einstellen und deren Auswirkungen auf das Lernverhalten interpretieren können. Daher konzentriert sich das Konzept auf die Einstellung der Hyperparameter in Abschnitt 5.4 und deren Auswirkungen auf das Lernverhalten in Abschnitt 5.5.

5.2.3 Zusammenfassung des Zusammenspiels der Komponenten

Im vorangegangenen Abschnitt wurde die Aufteilung des Selbstverstärkenden Lernens in Komponenten für die Automatisierung der Anpassungskonstruktion und deren Interaktionen vorgestellt. Mit Hilfe der vorgestellten Strukturierung werden die Funktionen in Module gekapselt und die Nutzerinteraktionen klar definiert. Im den folgenden Abschnitten wird das Selbstverstärkende Lernen für die Automatisierung der Anpassungskonstruktion vorgestellt. Dabei kann es helfen, die Anpassungskonstruktion auf das Labyrinth zu übertragen. Merkmalskombinationen sind Koordinaten und bestimmen den Zustand. Sie werden an die Eingangsneuronen des DQNs übergeben. Werden mehr als zwei Merkmale angepasst, wird das Labyrinth dementsprechend mehrdimensional. Anpassungen sind Aktionen. Sie ändern die Merkmale und führen zu einem neuen Zustand. Sie werden durch die Q-Values des DQNs vorgeschlagen. Die äußere Umrandung des Labyrinthes kann der Bauraum sein, der die Merkmale beschränkt. Wände sind Restriktionen, die beispielsweise aus der Fertigung herreichen. Gewünschte Merkmalskombinationen sind Schätze im Labyrinth, die der Agent finden soll. Ihr Wert ergibt sich aus der Erfüllung der Anforderungen.

5.3 Die Überführung von Konstruktionsaufgaben in Lernumgebungen

Die Konstruktionsaufgabe soll die Relationen zwischen den an sie gestellten Anforderungen und anzupassenden Merkmalen darstellen. Folglich müssen die möglichen Merkmale als Lösungsraum und die Anforderungen als Zielfunktionen vorgegeben werden.

5.3.1 Die Darstellung der Merkmale als Parameter Grid

Die anzupassenden Merkmale bilden den Zustand (s_t). Alle erlaubten Merkmalskombinationen bilden den Lösungsraum. Dieser wird im sogenannten *Parameter Grid* tabellarisch zusammengefasst. Die Zeilen bilden die Merkmale, die vom Selbstverstärkenden Lerner bei der Anpassung des Produkts berücksichtigt werden sollen. Die Spalten bilden Informationen über den Bauraum und sonstige Restriktionen wie beispielsweise Fertigungsgenauigkeiten. Dazu wird für jedes Merkmal eine untere und eine obere Grenze, sowie die Schrittweite angegeben. Die Schrittweite beschreibt die kleinste Änderungsmöglichkeit, die der Selbstverstärkende Lerner noch berücksichtigen soll. Sie ist notwendig, damit diskrete Zustände über die Aktionen erreicht werden. Darüber hinaus kann der Ausgang definiert werden. Falls dieser nicht definiert wird, wird er mittig zwischen die obere und untere Grenze gesetzt. Ein exemplarisches Parameter Grid für einen Balken mit den Merkmalen Länge, Breite und Höhe ist in Tabelle 6 dargestellt.

Tabelle 6: Paramter Grid für einen Balken mit drei Merkmalen (Länge, Breite und Höhe):

Merkmal	Untere Grenze	Obere Grenze	Ausgangszustand	Schrittweite
Länge	60,0	120,0	90,0	1,0
Breite	10,0	20,0	15,0	1,0
Höhe	10,0	20,0	15,0	1,0

5.3.2 Die Darstellung der Anforderungen als Belohnungsfunktionen

Neben den Merkmalen müssen die Anforderungen an den Selbstverstärkenden Lerner übergeben werden. In Anlehnung an Ehrlenspiel und Meerkamm werden 4 Typen von Anforderungen unterschieden [9]: Festforderungen, Intervallforderungen, Mindestforderungen und Maximalforderungen. Diese werden in individuelle Belohnungen mit Hilfe von Belohnungsfunktionen überführt. Dieses Vorgehen kann mit der Beschreibung von einzelnen Zielfunktionen bei der Mehrzieloptimierung verglichen werden. In dieser Arbeit werden lineare Belohnungsfunktionen verwendet. Produkte, die eine Anforderung nicht erfüllen, werden durch negative Strafterme bewertet (vgl. Gleichung 2). Um die Auswirkung der Bestrafung zu erhöhen, werden die Strafterme mit dem Faktor 2 skaliert. Folglich wird die 0 als Belohnung zu einem Grenzwert, der signalisiert, dass die individuelle Anforderung gerade noch erfüllt wird. Stellvertretend für die Belohnungsfunktionen ist in Bild 44 eine Belohungsfunktion, die für eine Eigenschaften E einer Belohnung R zuweist, für eine Maximalanfoderung dargestellt. Der kritische Wert

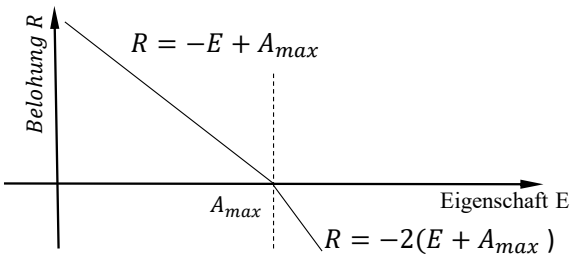


Bild 44: Darstellung der Belohnungsfunktion für eine Maximalanforderung

Für die vier Anforderungstypen sind sie in Tabelle 7 dargestellt und beschrieben. Anwendende müssen für jede Anforderung, die bei der Konstruktionsaufgabe berücksichtigt werden soll, den Typ der Anforderung sowie die jeweiligen kritischen Werte angeben.

5.3.3 Zusammenfassung der Überführung der Konstruktionsaufgaben

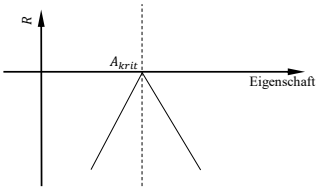
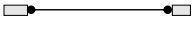
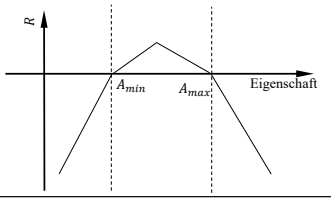
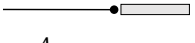
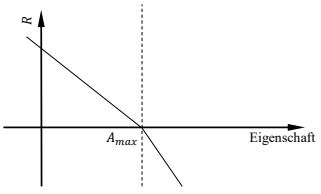
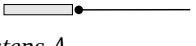
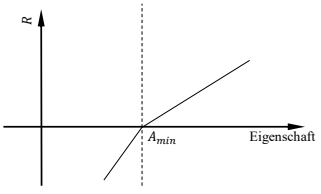
Im vorangegangenen Abschnitt wurde vorgestellt, wie Konstruktionsaufgaben in Lernumgebungen überführt werden können. Da die Lernumgebung die Informationen abbildet, die der Agent während des Trainings sieht, muss hier das Wissen der Produktentwickelnden formalisiert werden. Neben festen Randbedingungen wie Bauräume und Stufensprünge müssen vor allem die Anforderungen in Belohnungen übersetzt werden. Hierfür wurden in dieser Arbeit für jede Anforderungsart (Fest-, Intervall-, Maximal und Minimalanforderungen) eine Umsetzungsmöglichkeit vorgestellt. Diese Vorschläge werden in Kapitel 6 realisiert und deren Performanz in Kapitel 7 untersucht.

5.4 Die Stellschrauben für Produktentwickelnde

Das Aufsetzen des Selbstverstärkenden Lernens für eine Konstruktionsaufgabe entspricht der Modellierungsphase des CRISP-DM. Um das Lernverhalten des Selbstverstärkenden Lerners einstellen zu können, müssen Anwendende die entsprechenden Stellschrauben kennen. Da sie vor dem Training des Agenten mit der Umgebung festgelegt und anschließend nicht angepasst werden, sind sie übergeordnete Parameter – sogenannte Hyperparameter. Die Einstellung von Hyperparametern ist für jeden Maschinellen Lernprozess herausfordernd und muss meist spezifisch für den jeweiligen Anwendungsfall erfolgen [218, 219]. Im Falle des Selbstverstärkenden Lernens muss zusätzlich die Eigenheit berücksichtigt werden, dass die Erfahrungen, die mit der Umgebung im

Laufe des Trainings gemacht werden, ebenfalls von den Hyperparametern abhängig sind. Daher werden in diesem Abschnitt die Hyperparameter in vier Gruppen eingeteilt und vorgestellt. Zuerst werden die Stellschrauben, welche die zeitliche Abfolge des Trainings managen, beschrieben. Als Zweites werden die Hyperparameter, welche den Umgang des Agenten mit der Lernumgebung definieren, vorgestellt. Es folgen die Parameter der Netzarchitektur, die die mathematischen Möglichkeiten des Agenten definieren. Anschließend werden die Parameter, die sich auf die Verarbeitung des Losses für die Netzanpassung des Evaluation DQNs beziehen, vorgestellt. Abschließend werden alle Parameter tabellarisch zusammengefasst, um eine Übersicht zu erhalten und den ersten Teil der 2. Forschungsfrage zu beantworten. Deren Auswirkungen auf das Lernverhalten werden hier beschrieben und können mit den Indikatoren aus Abschnitt 5.5 beurteilt werden.

Tabelle 7: Belohnungsfunktionen für die vier Anforderungstypen:

Typ	**Darstellung**	**Eingabe**	**Belohungsfunktion**
Festforderungen *„Eigenschaft muss A_{krit} entsprechen“*		Fest, A_{krit}	
Intervallforderungen *„Eigenschaft muss mindestens A_{min} und höchstens A_{max} entsprechen“*		Intervall, A_{min}, A_{max}	
Maximalforderungen *„Eigenschaft muss höchstens A_{max} entsprechen“*		Max, A_{max}	
Mindestforderungen *„Eigenschaft muss mindestens A_{max} entsprechen“*		Min, A_{min}	

5.4.1 Die Stellschrauben der zeitlichen Abfolge

Das vorgestellte Konzept des Selbstverstärkenden Lernens arbeitet in diskreten Zeitschritten. Jeder Zeitschritt beschreibt das Durchlaufen der Schleife über Agent und Lernumgebung zur Generation von Erfahrungen (s. Bild 39). Die übrigen Teile des Konzeptes, wie beispielsweise das Training des Evaluation DQN oder die Synchronisierung des Target DQN, müssen mit dieser Routine synchronisiert werden. Zunächst soll der Agent nach einer definierten Anzahl an Zeitschritten N_T wieder zurück zum Ausgangszustand gebracht werden. Dies ist notwendig, da sich der Agent nur relativ zu seinem aktuellen Zustand s_t bewegen kann. Ist der Agent tief in einen unerwünschten Bereich des Lösungsraums vorgedrungen, wird es für ihn schwierig und zeitaufwändig in einen gewünschten Bereich zurückzukehren. Dies kann durch das Labyrinth in Bild 42 auf Seite 85 veranschaulicht werden. Ein Agent, der mehrmals schlechte Entscheidungen getroffen hat, kann sich in einer Ecke des Labyrinthes befinden, aus der er sich ohne eine entsprechende Strategie schwer selbstständig befreien kann. Daher wird der Agent in Episoden aus N_T Zeitschritten trainiert. Ab Anfang jeder Episode wird der Agent auf den Startzustand, den so genannten *inital state* zurückgesetzt. Bei der Wahl von N_T sollte berücksichtigt werden, dass je größer der Lösungsraum ist, desto mehr Zeitschritte sollte der Agent für eine Episode haben, da er möglicherweise weite Wege zu einem gewünschten Zustand zurücklegen muss. Die Gesamtdauer des Trainings wird durch die Anzahl der Episoden N_E festgelegt. Die Anzahl der Episoden kann iterativ erweitert werden, wenn das Evaluation DQN am Ende des Trainings gespeichert und wieder zur Verfügung gestellt werden kann. Dies wird ausführlich in Abschnitt 5.6 beschrieben, da es Teil des Transferlernens für Nachfolgende Anpassungskonstruktionen ist. Die Netzanpassung des Evaluation DQN wird nach dem Beginn des Trainings verzögert. In der Anfangsphase des Trainings ist es wahrscheinlich, dass sich der Agent um den Startpunkt bewegt. Daher wird ein überwiegender Anteil der Erfahrungen, die anfangs in der Historie gespeichert werden, in diesem Bereich beschrieben. Erfolgt die Netzanpassung zu früh ist es wahrscheinlicher, dass eine Strategie für den Bereich um den Startpunkt erlernt wird. Diese ignoriert möglicherweise bessere, weiter entfernte Zustände. Daher bestimmt N_{Start} die Anzahl an Erfahrungen, welche in der Historie vorhanden sein müssen, damit die erste Netzanpassung vorgenommen wird. Dabei ist zu beachten, dass N_{Start} größer sein muss als die später beschriebene Batchgröße N_{Batch}. Die letzte zeitliche Stellschraube ist die Synchronisation des Target DQN. Das Target DQN dient als sogenannte *Grounded Truth*. Es imitiert die wahren Werte durch zukünftige Werte und dient der Netzanpassung zur Berechnung des Losses mit Hilfe der Bellman-Gleichung für Q-Learning (s. Gleichung 11 auf Seite 54). Das Target DQN soll träger sein

als das Evaluation DQN und nur in definierten Abständen - der Synchronisationsrate N_{Sync} - auf den aktuellen Stand des Evaluation DQN gebracht werden. Folglich müssen für die zeitliche Abfolge die Anzahl der Zeitschritte pro Episode N_T, die Anzahl der Episoden N_E, der Start des Trainings N_{Start} und die Synchronisationsrate N_{Start} bestimmt werden.

5.4.2 Die Stellschrauben der Explorations-Strategie

Zu Beginn jedes Zeitschrittes entscheidet der Agent, ob der Lösungsraum durch eine zufällig gewählte Aktion erkundet oder die Vorhersagen des Evaluation DQNs für die Bestimmung der Aktion genutzt werden sollen. Ersteres wird als *Exploration* und zweiteres als *Exploitation* bezeichnet. Die Balance zwischen Exploration und Exploitation ist vor allem entscheidend für die Erkundung des Lösungsraums [220, 221]. Wird der Agent zu sehr an der Exploration gehindert, ist es wahrscheinlicher, dass er den Lösungsraum unzureichend erfasst und gewünschte Zustände nicht gelernt werden können. Je größer der Explorationsanteil wird, desto wahrscheinlicher werden neue Zustände erreicht und somit kann der Agent den Lösungsraum sicherer beschreiben. Allerdings müssen neue Zustände durch die Analyse- und Synthesekomponente berechnet werden. Hierdurch entsteht der wesentliche Berechnungsaufwand, der minimal bleiben soll. Folglich erkundet eine ideale Explorations-Strategie den Lösungsraum gerade soweit, dass gewünschte Zustände erfasst werden können.

Für die Explorations-Strategie wird in diesem Ansatz ein ϵ-Greedy Ansatz gewählt. Dabei wird davon ausgegangen, dass zu Beginn des Trainings der Explorationsanteil groß gegenüber der Exploitation sein sollte, weil der Agent zunächst Erfahrungen mit der Umgebung sammeln soll. Die Wahrscheinlichkeit der Exploration wird dann linear über die Zeitschritte reduziert, damit im Laufe des Trainings der Exploitationsanteil überhand nimmt. Um dieses Verhalten dem Agent zu vermitteln wird der Schwellwert ϵ (die sogenannte Decay Rate, $0 \leq \epsilon \leq 1$) eingeführt. Dieser wird von einem Startwert ϵ_{Start} bis zu einem Endwert ϵ_{Final} linear über $N_{DecayFinal}$ reduziert. Drei unterschiedliche Beispiele für Decay Raten sind

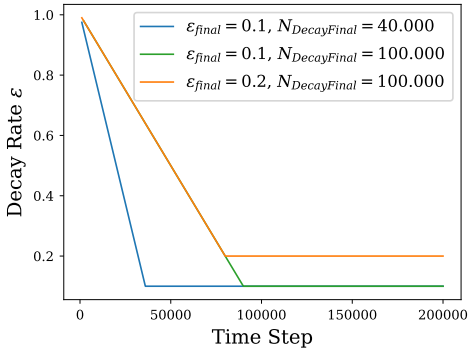


Bild 45: Darstellung drei Decay Raten ϵ, die über die Hyperparameter ϵ_{Final} und $N_{DecayFinal}$ eingestellt werden

in Bild 45 dargestellt. Zu Beginn jedes Zeitschrittes wird eine Pseudozufallszahl zwischen 0 und 1 generiert. Liegt die Zahl über ϵ wird das DQN zur Vorhersage der Aktion genutzt (Exploitation). Andernfalls wird eine zufällige Aktion verwendet (Exploration). ϵ_{Start}, ϵ_{Final} und $N_{DecayFinal}$ sind die Stellschrauben hinsichtlich der Explorations-Strategie. Für ϵ_{Start} empfiehlt sich ein hoher Wert. ϵ_{Final} und $N_{DecayFinal}$ sollten so klein wie möglich, jedoch so groß wie zur Erfassung gewünschter Zustände notwendig, gewählt werden.

5.4.3 Die Stellschrauben der Netzarchitektur

Die Netzarchitektur entscheidet über die mathematischen Möglichkeiten, die das Netz zur Erfassung der Zusammenhänge zwischen Ein- und Ausgangsgrößen hat. Hessel u. a. zeigen, dass enorme Potentiale in komplexer Netzarchitektur liegen [222]. In dieser Arbeit wird ein Werte-basierter (im Englischen *value-based*) Ansatz verfolgt. Der Zustand s_t wird dabei aus einem eindimensionalen Vektor repräsentiert. Die Einträge des Vektors sind Fließkommazahlen, die die Merkmalsausprägung des jeweiligen Merkmals definieren. Die DQNs erhalten jedoch nur den Vektor. Die Bedeutung der Einträge entfällt. Das DQN verarbeitet die übergebene Fließkommazahl am entsprechenden Eingangsneuron ohne zu berücksichtigen, um welches Merkmal es sich dabei handelt. Da die meisten komplexeren Netzarchitekturen mehrdimensionale Eingangsvektoren benötigen, wird in dieser Arbeit ein Tiefes, Neuronales Netz mit N_S versteckten Schichten und N_K Knoten pro Schicht verwendet. Die Knoten sind vollständig verknüpft und Vorhersagen des Netzes werden im *Forward Feed* Modus getroffen. Da die restliche Architektur (Eingangs- und Ausgangsschicht) durch die Dimension des Zustand (s_t) und die Aktionsmöglichkeiten ($s_t \times a_t$) vorgeschrieben werden, sind die Stellschrauben für die Netzarchitektur N_S und N_K. Je größer die Netzarchitektur, desto höher ist die Wahrscheinlichkeit Zusammenhänge zwischen Ein- und Ausgangsgrößen zu erfassen. Jedoch wird dadurch der Trainingsprozess aufwendiger, weil mehr Knoten- und Kantengewichte des DQNs angepasst werden müssen.

5.4.4 Die Stellschrauben der Netzanpassung

Die Netzanpassung ergibt sich aus dem Loss, der Lernrate des Evaluation DQN (α) und der Größe des Erfahrungs-Batches, der für die Anpassung des DQNs, verwendet wird. Der Loss ergibt sich nach der Bellman-Gleichung für Q-Learning (s. Gleichung 11 auf Seite 54) aus den vorhergesagten Q-Values des Evaluation DQN ($Q(s_t)$) für s_t und des Target DQN ($Q(s_{t+1})$) für s_{t+1}, sowie den erhaltenen Belohnungen $r(t)$. Von den Vorhersagen des Target DQN

wird der höchste Q-Value als Richtwert für die Netzanpassung vorgegeben. Dieser wird mit γ der *Discount Rate* ($0 \leq \gamma \leq 1$) multipliziert.

Je höher die Discount Rate, desto größer ist der Einfluss späterer Zustände [32, 223]. Später bezieht sich auf die Anzahl der Zeitschritte, die notwendig sind, um diesen Zustand zu erreichen. Die Discount Rate gibt folglich an, wie weitsichtig der Agent ist und hat einen direkten Einfluss auf den Loss und damit die Netzanpassung. Je größer die Batchgröße N_{Batch}, desto mehr Erfahrungen werden pro Netzanpassung ausgewertet. Der Berechnungsaufwand steigt. Dieser Mehraufwand ist jedoch marginal, da die Erfahrungen nur für die Vorhersagen der Netze und nicht für die Backpropagation verwendet werden. Analog reduziert die Batchgröße den Einfluss einzelner Erfahrungen mit der Lernumgebung auf die Netzanpassung. Je größer die Batchgröße ist, desto geringer ist die Wahrscheinlichkeit der Überanpassung auf eine Erfahrung. Dadurch wird die Netzanpassung harmonischer und weniger sprunghaft. Das hat den zusätzlichen Nutzen, dass die Interpretation der Netzanpassung erleichtert wird. Die Lernrate α ($0 \leq \alpha \leq 1$) wird mit dem Loss multipliziert, um die Netzanpassung pro Durchlauf zu steuern. Kleinere Lernraten führen zu einer langsameren aber stabilen Netzanpassung und können vor allem in Kombination mit einer geringen Discount Rate dazu führen, dass der Agent an einem lokalen Maximum im Lösungsraum hängen bleibt, weil die Anpassung nicht ausreicht, um von diesem lokalen Maximum potentiell bessere lokale Maxima oder das globale Maximum zu erkennen. Große Lernraten führen zu einer schnellen aber potentiell instabilen Netzanpassung. Dabei muss in Kombination mit einer hohen Discount Rate berücksichtigt werden, dass gute Zustände des Lösungsraums übersprungen werden können.

5.4.5 Zusammenfassung der Stellschrauben

Selbstverstärkendes Lernen zeichnet sich durch eine Vielzahl von Stellschrauben, die einen Einfluss auf das Lernverhalten und den Trainingsaufwand haben, aus. In diesem Abschnitt wurden sie in vier Gruppen unterteilt und beschrieben. Sie sind in Tabelle 8 zusammengefasst, um den ersten Teil der 2. Forschungsfrage zu beantworten (vgl. Seite 64). Im folgenden Abschnitt werden die Indikatoren, mit denen sich das Lernverhalten und der Trainingsaufwand beurteilen lassen vorgestellt.

Tabelle 8: Übersicht aller Stellschrauben des Selbstverstärkenden Lernens für Produktentwickelnde:

Name	**Abkürzung**	**Wertebereich**	**Beschreibung**
Zeitschritte Pro Episode	N_T	$\mathbb{N}^+$	Anzahl der Zustände des Agenten pro Episode
Anzahl der Episoden	N_E	$\mathbb{N}^+$	Anzahl der Episoden für das Training
Start des Trainings	N_{Start}	$\mathbb{N}^+$	Zeitschritt nachdem die Anpassung der Gewichte beginnt
Synchronisationsrate	N_{Start}	$\mathbb{N}^+$	Anzahl der Zeitschritte, die zwischen dem Clonen der Gewichte des Target DQNs in das Evaluation DQN liegt
Startwert der Decay Rate	ϵ_{Start}	$0 \leq \epsilon_{Start} \leq 1$	Ausgangswert des Schwellwerts der Exploration-Strategie
Endwert der Decay Rate	ϵ_{Final}	$0 \leq \epsilon_{Final} \leq \epsilon_{Start}$	Endwert des Schwellwerts der Exploration-Strategie
Letzter Zeitschritt der Decay Rate	$N_{decayfinal}$	$\mathbb{N}^+$	Letzter Zeitschritt für die Anpassung des Schwellwerts der Exploration-Strategie
Anzahl der Verstecken Schichten	N_S	$\mathbb{N}^+$	Anzahl der versteckten Schichten in der Netzarchitektur der DQNs
Anzahl der Knoten pro Versteckter Schicht	N_K	$\mathbb{N}^+$	Anzahl der Knoten pro versteckter Schicht in der Netzarchitektur der DQNs
Batchgröße	N_{Batch}	$\mathbb{N}^+; N_{Batch} \leq N_{Start}$	Anzahl der Erfahrungen, die für die Berechnung des Fehlers gleichzeitig verwendet werden
Lernrate	α	$0 \leq \alpha \leq 1$	Faktor des Fehlers bei der Netzanpassung
Discount Faktor	γ	$0 \leq \gamma \leq 1$	Faktor mit dem zukünftige Erfahrungen gewichtet werden

5.5 Indikatoren für das Lernverhalten und den Trainingsaufwand

Im vorangegangen Abschnitt wurde dargelegt, welche Möglichkeiten Produktentwickelnde haben, das Lernverhalten und den Trainingsaufwand zu beeinflussen. Diesen Stellschrauben werden im folgenden Abschnitt Indikatoren, die es ermöglichen gelernte Strategien zu interpretieren, gegenübergestellt. Dies entspricht der Evaluationsphase des CRIPS-DM. Alle Indikatoren werden über die Zeitschritte des Agenten beobachtet.

Zunächst wird der Loss betrachtet. Der Loss ist ein Maß für den Fehler, auf den das DQN angepasst wird. Er wird in den meisten Maschinellen Lernverfahren, beispielsweise der Regressionsanalyse mit einem Neuronalen Netz, verwendet, um das Lernverhalten zu beobachten. Als zweites wird der Reward für die individuellen Anforderungen und deren gewichtete Summe vorgestellt. Diese geben vor allem Aufschluss darüber, ob der Agent Zustände gefunden hat, die die Anforderungen erfüllen. Des weiteren ermittelt der Regret, wie weit der Agent von einer optimalen[3] Lernstrategie entfernt ist. Er wir bevorzugt eingesetzt, um verschiedene Agenten miteinander zu vergleichen. Zuletzt wird der Trainingsaufwand abgeschätzt. Hierzu werden die besuchten Zustände gezählt. Ein neuer Zustand bedeutet ein Aufruf der API und damit der Synthesekomponente, die üblicherweise den Hauptaufwand des Trainings verursacht. Diese Art der Abschätzung hat den Vorteil, das sie Hardware unabhängig ist und den Vergleich mit anderen Automatisierungswerkzeugen ermöglicht. Für das Monitoring von Maschinellen Lernprozessen sollte immer berücksichtigt werden, wie der Computern mit dem „Zufall“ umgeht. Ein Computer kann keine echten Zufallszahlen generieren. Es werden Pseudozufallszahlen mit Hilfe von Hash-Werten generiert. Diese basieren auf den sogenannten *Random Seeds*. Diese sind ebenfalls Zahlen. Über das Festhalten, d. h. vorschreiben, des *Random Seeds* werden nachfolgende Zufallszahlen nachvollziehbar. Das bedeutet, dass bei jedem Durchlauf des Programms die erste zufällig gezogene Zahl gleich ist. Beim Training des Selbstverstärkenden Lernens werden Zufallszahlen für die Initialisierung der DQNs, der Auswahl der Explore-Aktion und der Auswahl der Erfahrungstupel aus der Historie verwendet. Um deren Effekte auf das Monitoring auszuschließen, werden bewusst *Random Seeds* definiert, d. h. die Zufallszahlen determiniert. Dafür werden mehrere *Random Seeds* für das Training verwendet. Innerhalb dieser Arbeit sind es zehn. Alle Indikatoren stellen den Mittelwert über diese zehn Agenten als Linie und das 95zigste Perzentil als farblich passend gekennzeichneten Bereich dar.

[3] Optimalen bedeutet in diesem Fall ideal im Hinblick auf die gesammelten Erfahrungen des Agenten.

5.5.1 Der Loss

Die Überwachung des Loss während des Trainings eines neuronalen Netzes ist die gebräuchlichste Metrik, um Probleme beim Lernverhalten zu identifizieren [219]. Hier ergibt sich der Loss aus der Bellman-Gleichung für Q-Learning (s. Gleichung 11 auf Seite 54) und beschreibt die Differenz zwischen den Vorhersagen des Evaluation DQNs und des Target DQNs über N_{Batch} Erfahrungen mit der Umgebung. Zusammen mit der Lernrate α bestimmt der Loss die Netzanpassung. Er kann in logarithmischer Darstellung betrachtet werden, weil davon auszugehen ist, dass zu Beginn des Trainings sehr große Anpassungen erforderlich sind, die dann im Laufe der Zeitschritte, wenn sich der Agent auf eine Strategie festgelegt hat, sehr klein werden. Der Loss gibt Produktentwickelnden vor allem Aufschluss darüber, ob die Möglichkeiten des DQNs ausreichen, um ausreichend genaue Vorhersagen zu treffen [219].n Bei stark schwankenden oder sogar exponentiellen Anpassungsraten sollte die Netzarchitektur N_S, N_L und die Lernrate α angepasst werden. In Bild 46 ist der Loss von zwei Selbstverstärkenden Lernern exemplarisch dargestellt. Der blaue Loss zeigt das gewünschte Verhalten. Er startet relativ hoch und wird über den zeitlichen Verlauf sehr klein. Das bedeutet, dass das Netz nur noch minimal angepasst wird. Der Lerner hat eine Strategie gefundenen und behält diese bei. In diesem Beispiel wurde keine logarithmische Skala verwendet, um den orangen Loss darstellen zu können. Dieser zeigt ein exponentielles Verhalten. Der Lerner kann hier keine passende Strategie finden. Der Grund ist in diesem Beispiel eine schlecht gewählte Discount Rate. Die rhythmischen Zacken, die vor allem bei der orangen Kurve und im Anfangsbereich der blauen Kurve zu sehen sind, spiegeln die Synchronisation des Target DQN wieder. Während der Loss selbst wichtig ist, um zu überprüfen, ob die DQN-Architektur in der Lage ist, Q-Werte genau vorherzusagen, spiegelt er nicht wider, ob der RL-Agent die passenden Merkmalskombinationen findet. Zu diesem Zweck werden im folgenden Abschnitt die Rewards hinzugezogen.

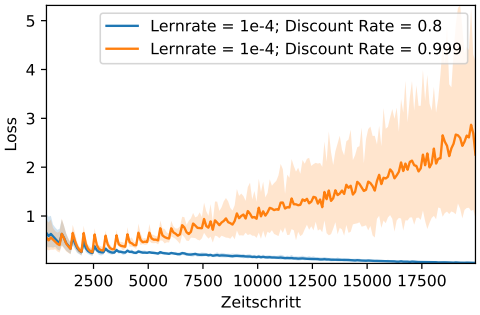


Bild 46: Darstellung des Loss von zwei Lernern mit unterschiedlicher Discount Rate

5.5.2 Der Reward

Der Reward spiegelt wider, wie gut ein Zustand – eine Merkmalskombination – im Bezug auf die Anforderungen ist. Es werden sowohl die individuellen Rewards als auch deren gewichtete Summe als sogenannter *Total Reward* aufgezeichnet. Jeder individuelle Reward repräsentiert eine Anforderung. So kann beurteilt werden, ob jede Anforderung für sich genommen berücksichtigt wird. Die Belohnungsfunktionen (vgl. Unterabschnitt 5.5.2) sind so definiert, dass der Wert 0,0 den kritischen Wert darstellt, an dem die Anforderung gerade noch erfüllt wird. Der Total Reward ist die übergeordnete Zielfunktion des Selbstverstärkenden Lerners. Er ist die gewichtete Summe der individuellen Rewards. In Bild 47 ist der Total Reward für das in Unterabschnitt 5.5.1 gezeigte Beispiel aufgegriffen. Zusätzlich sind in Bild 48 die zwei individuellen Rewards, welche Bild 47 zu Grunde liegen dargestellt.

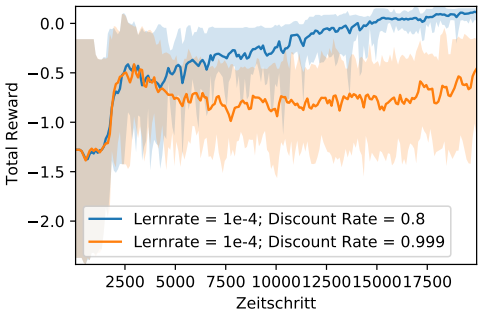


Bild 47: Darstellung des Rewards von zwei Lernern mit unterschiedlicher Discount Rate

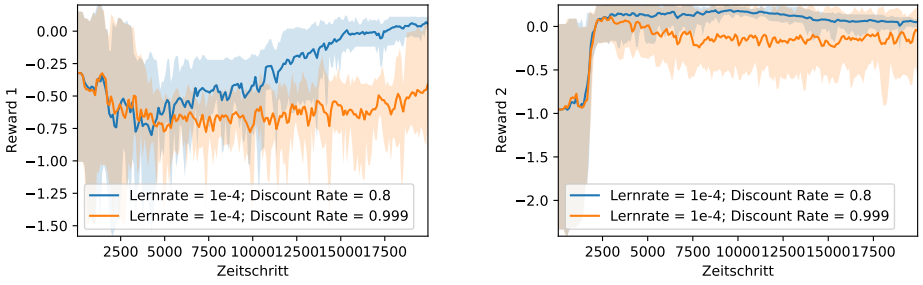


Bild 48: Darstellung der individuellen Rewards zu Bild 47

Wenn der Agent eine Strategie findet, dann konvergieren sowohl die individuellen als auch der Total Reward. Die Konstruktionsaufgabe ist dann gelöst, wenn für alle individuellen Rewards gegen einen Wert größter als 0,0 konvergiert (vgl. blaue Kurve in Bild 47 und Bild 48). In diesem Fall können Anwendenden versuchen den Trainingsaufwand zu reduzieren. Dazu wird die Exploration des Agenten reduziert, indem $N_{DecayFinal}$ reduziert wird. Wenn die Rewards konvergieren, aber ein individueller Reward nicht über 0,0 liegt, bedeutet das, dass diese Anforderung nicht erfüllt ist (vgl. orange Kurve in Bild 47 und Bild 48). Dementsprechend können Anwendenden die Exploration forcieren, indem $N_{DecayFinal}$ oder die Discount Rate γ angepasst

wird. Es kann jedoch nicht ausgeschlossen werden, dass im Lösungsraum keine Merkmalskombination existiert, die alle Anforderungen erfüllt. Wenn die Rewards nicht konvergieren, wird keine Strategie gelernt. Gründe dafür können eine ungeeignete Lernrate α, Discount Rate γ, Netzarchitektur N_S, N_L oder Explorations-Strategie sein. Diese Parameter sind spezifisch für den Anwendungsfall. Im Falle der in Kapitel 7 gezeigten Anwendungsbeispiele wurde versucht, zunächst mit kleinen Lernraten, großer Discount Raten, einem großem DQN und großem Explorationsanteil auf Kosten von erhöhtem Trainingsaufwand den Agenten zur Konvergenz zu bringen.

5.5.3 Der Regret

Der dritte Indikator, der zum Verständnis des Lernverhaltens des Agenten beiträgt, ist der Regret. Die wörtliche Übersetzung „Bedauern“ trifft den Zweck dieser Metrik sehr gut. Er drückt im Rückblick auf die im Verlauf des Trainings getroffenen Entscheidungen aus, wie groß die Differenz der Belohnungen zwischen den getroffenen und den optimalen Entscheidungen ist. Es gibt verschiedene Brechungsmöglichkeiten für den Regret. Dabei sind zwei Faktoren entscheidend. Zum einen wird hier der Regret für jede Aktion berechnet und über die Zeitschritte kumuliert angegeben (sog. *per-action regret*). Zum anderen ist die Berechnungsgrundlage für die optimalen Entscheidungen relevant. Hier wird der beste gefundene Zustand zur Berechnung des Regrets verwendet. Für jeden Zeitschritt ergibt sich der Regret aus der Differenz des besten Rewards, der bisher von allen Agenten für diesen Lösungsraum gefunden wurde, und dem aktuellen Reward. Folglich wird dieser Indikator immer dann ansteigen, wenn der Ausgangszustand nicht der beste Zustand selbst ist, weil der Agent eine Reihe von Aktionen durchführen muss, um den besten Zustand zu erreichen. Des weiteren können mehrere Agenten hinsichtlich ihrer Trainingseffizienz betrachtet werden. In Bild 49 sind zwei Regretkurven für unterschiedliche Agenten dargestellt. Unter der Annahme, dass beide Agenten eine passende Strategie für die Lernaufgabe finden, hat der Agent mit der nierdrigeren Kurve weniger Aktionen benötigt, um diese zu lernen. Die Anzahl der Aktionen gibt jedoch nicht an, wie viele Zustände für die einzigartige Strategie berechnet werden müssen.

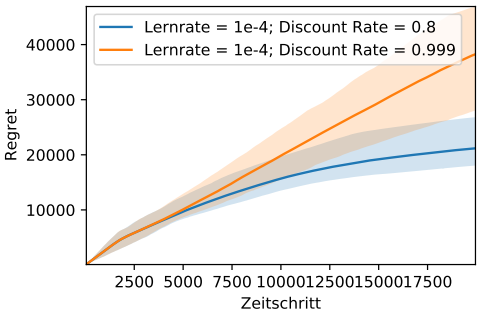


Bild 49: Darstellung des Regrets von zwei Lernern mit unterschiedlicher Discount Rate

Die Anzahl der einzigartigen Zustände ist der entscheidende Indikator des Trainingsaufwandes. Er wird im folgenden Abschnitt vorgestellt.

5.5.4 Der Trainingsaufwand

Zwei Faktoren bestimmen den Rechenaufwand beim Training. Zum einen die Backpropagation des Evaluation DQN und zum anderen die Absicherung der Eigenschaften in der Synthese-Komponente. Der erste Faktor ist Hardware abhängig und lässt sich über die Berechnung auf der GPU oder sogar in einer Cloud sehr gut managen. Der zweite Faktor ist daher für diese Arbeit entscheidend. Der Trainingsaufwand soll die Anzahl der Simulationen widerspiegeln, die notwendig sind, um die Strategie zu erlernen. Einerseits entspricht das dem Vorgehen der Absicherung von Produkten und anderseits ist es möglich, den Selbstverstärkenden Lerner mit anderen Automatisierungswerkzeugen zu vergleichen.

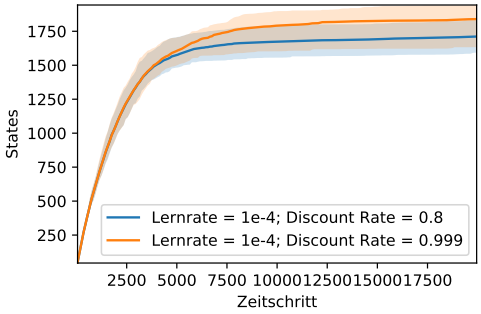


Bild 50: Darstellung des Regrets von zwei Lernern mit unterschiedlicher Discount Rate

Da der Agent sich schrittweise bewegt und zu Beginn jeder Episode an den Initial State zurückgesetzt wird, ist es zielführend vorgeschlagene Zustände mit bereits bekannten Zuständen abzugleichen. Eine bereits bekannte Merkmalskombination wird in der Synthese im Rahmen der numerischen Unsicherheiten identische Eigenschaften erzielen und damit auch die gleiche Belohnung erhalten. Die in Abschnitt 5.2 dargelegte Bypass-Option schließt die Analyse- und Synthese-Komponente kurz, wenn der vorgeschlagene Zustand bereits in der Historie vorhanden ist. Zur Bestimmung des Trainingsaufwands werden folglich die berechneten Zustände über die Zeitschritte verwendet. Eine exemplarische Darstellung dieses Indikators für zwei Lerner ist in Bild 50 dargestellt.

5.5.5 Zusammenfassung der Indikatoren

Das Selbstverstärkende Lernen kann durch eine Vielzahl von Indikatoren überwacht werden. Neben den für das Maschinelle Lernen üblichen Indikatoren, bspw. der Loss, der die Anpassung des tiefen Neuronalen Netzes angibt, werden in dieser Arbeit vor allem die Anzahl der erforderlichen Zustände (Visited States), der Reward und der Regret betrachtet. Die Anzahl der erfor-

derlichen Zustände gibt hardware-unabhängig an, wie viele Simulationen des Produktes zum Erlernen der Lösungsstrategie notwendig sind. Sie ermöglicht den Vergleich mit andern Automatisierungswerkzeugen, wie beispielsweise der Mehrzieloptimierung. Der Reward ist abhängig von den Rewardfunktionen. Diese wurden so konzipiert, dass ein Reward von 0,0 gerade so die Anforderung erfüllt. Dabei wird zwischen individuellen Rewards für einzelne Anforderungen und dem sogenannten Total Reward, welcher sich in dieser Arbeit aus der gewichteten Summe der individuellen Rewards ergibt, unterschieden. Das Ziel des Selbstverstärkenden Lerners ist es eine Merkmalskombination zu finden, deren Rewards für alle individuellen gleichzeitig größer als 0,0 ist, da hier alle Anforderungen gleichzeitig erfüllt werden. Zuletzt ist der Regret ein Hilfsmittel zum Vergleich verschiedener Lernstrategien untereinander und besonders hilfreich, wenn Hyperparameter verglichen werden sollen. Er repräsentiert, den Abstand der gelernten Strategie zur bestmöglichen. Letztere wird rückwirkend berechnet, indem ausgehend vom aktuellen Wissensstand des Agent der Unterschied zwischen den tatsächlich gemachten Erfahrungen und den Erfahrungen, die möglich gewesen wären, wenn sich der Agent immer ideal verhalten hätte.

5.6 Das Konzept für Übertragung auf nachfolgende Produktgenerationen

In den vorangegangen Abschnitten wurde dargelegt, wie sich Selbstverstärkendes Lernen zur Automatisierung einer Konstruktionsaufgabe einsetzen lässt. Nach dem Beschreiben der Konstruktionsaufgabe (s. Abschnitt 5.3) und dem iterativen Anpassen der Stellschrauben (s. Abschnitt 5.4 und Abschnitt 5.5) steht für die Konstruktionsaufgabe eine Strategie zur Verfügung, die vom unerwünschten Ausgangszustand zum gewünschten Zielzustand führt. Die Strategie des Selbstverstärkenden Lernen wird durch das trainierte Evaluation DQN abgebildet. Dessen Gewichte wurden durch das Explorieren der Lernumgebung so angepasst, dass sie den Agenten durch die Konstruktionsaufgabe wie durch ein Labyrinth lenken (vgl. Bild 42 auf S. 85). Die Richtungen im Labyrinth sind die Anpassungsmöglichkeiten (wie beispielsweise das Vergrößern einer Merkmalausprägung) des Agenten. So empfängt das Evaluation DQN einen Zustand (Merkmalskombination) auf den Eingangsneuronen und sagt dafür eine Aktion über die Q-Values auf der Ausgangsschicht vorher. Die Aktion wird an die Lernumgebung übergeben, die durch die Aktion vom momentanen Zustand in den nächsten Zustand übergeht. Im nächsten Durchlauf wird dieser wiederum an das DQN übergeben. Diese Folge von Aktionen und Zuständen bildet einen nachvollziehbaren Weg durch den Lösungsraum. Im Rahmen der Anpassungskonstruktion kön-

nen sich Konstruktionsaufgaben nur eingeschränkt ändern, da die prinzipielle Lösung erhalten bleibt. Beispielsweise gestaltet sich ein Spannungsverlauf ähnlich, wenn die Randbedingungen angepasst werden, aber die kritischen Spannungen an Kerben oder Querschnittsübergängen entstehen, die von der Anpassung der Randbedingung nicht nennenswert beeinflusst werden. Daher ist es häufig sinnvoll, eine gelernte Strategie auf nachfolgende Produktgeneration zu übertragen. Dies hebt das Selbstverstärkende Lernen von

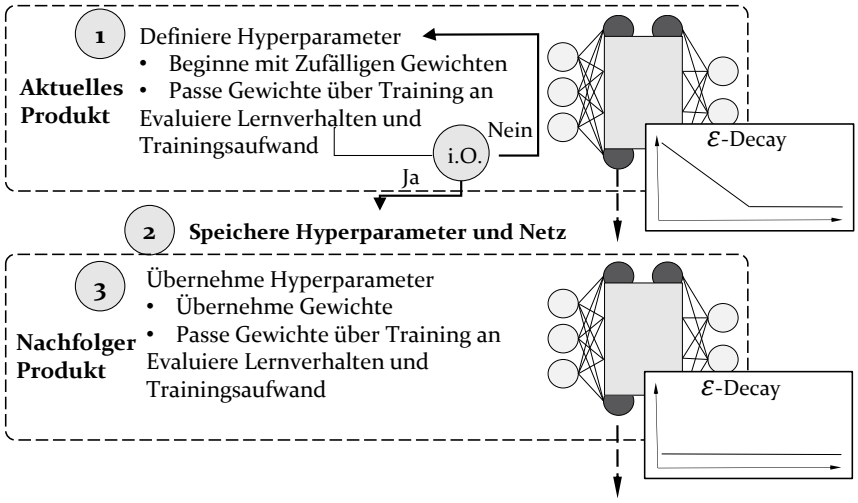


Bild 51: Konzept der Übertragung einer gelernten Strategie auf nachfolgende Produktgenerationen der selben Anpassungsgeneration

Aus der Perspektive der Mehrzieloptimierung können Konstruktionsaufgaben für nachfolgende Produktgenerationen der gleichen Anpassungskonstruktion in zwei Gruppen unterteilt werden. Entweder es werden neue Anforderungen an die Konstruktionsaufgabe gestellt oder deren Randbedingungen ändern sich. Während für neue Anforderungen eine Merkmalskombination auf der bestehenden Pareto-Front gesucht werden kann, muss bei einer Änderung der Randbedingungen die Optimierungsaufgabe neu gelöst werden. Im Falle des Selbstverstärkenden Lernens kann das trainierte Netz gespeichert und für die nächste Produktgeneration als Ausgangspunkt genutzt werden. Anschließend muss die Strategie nicht von Beginn an neu gelernt werden, sondern nur noch auf den neuen, aber vermutlich ähnlichen Lösungsraum kalibriert werden. Dazu wird mit Hilfe stark eingeschränkter Exploration die gelernte Strategie auf dem neuen Lösungsraum angewandt. Über das Training entstehen so Erfahrungen mit der neuen Konstruktionsaufgabe. Mit

deren Hilfe das die Gewichte des Netzes feingetuned werden. Dieses Konzept ist in Bild 51 dargestellt. Bis auf die Explorations-Strategie wird angenommen, dass sich der Lösungsraum ähnlich genug verhält, um die eingestellten Hyperparameter übernehmen zu können. Ist dies nicht der Fall, müssen sie über die Indikatoren ebenfalls kalibriert, d. h. relativ zum übernommenen Wert eingestellt werden. Hinsichtlich der Explorations-Strategie wird die Decay Rate nicht mehr mit den Zeitschritten reduziert, sondern von Beginn an auf dem vorherigen Endwert definiert. So wird die Exploitation des Netzes forciert und ein direktes Feedback, ob die übertragene Strategie auch für den neuen Anwendungsfall funktioniert, generiert.

5.7 Zusammenfassung der vorgestellten Konzepte

Das vorgestellte Konzept deckt einerseits den Bedarf eines computerverarbeitbaren Modells für die Anpassungskonstruktion ab. Auf dessen Grundlage können Automatisierungswerkzeuge direkt zur Anpassungskonstruktion verortet werden. Dies wirkt den schwer übertragbaren, Anwendungsfall-spezifischen Ansätzen, die die Forschungslandschaft dominieren, entgegen. Das Modell ordnet den Prozess der Anpassungskonstruktion hierarchisch und chronologisch auf Basis der Prinzipiellen Lösung. So ist es möglich, den Prozess in Konstruktionsaufgaben zu unterteilen, die einerseits eine zeitliche Reihenfolge und andererseits einen Zweck, im Sinne der zu erfüllenden Anforderungen, beinhalten. Das Konzept der Konstruktionsaufgabe ist prädestiniert für die Automatisierung, da Ein- und Ausgangsgrößen, sowie Randbedingungen definiert werden können. Zur Verarbeitung des Modells durch den Computer wurde die notwendigen Konzepte und Axiome eines semantischen Netzes vorgestellt. Diese wurden so formuliert, dass eine Anknüpfung an andere semantische Netze möglich ist. Anderseits deckt das Konzept den Bedarf nach einem datenunabhängigen Automatisierungswerkzeug für die Anpassungskonstruktion ab. Das Selbstverstärkende Lernen wurde als innovative Technologie für die Produktentwickelnden aufbereitet. Dabei wurden die Stellschrauben, mit den Produktentwickelnden ihr Expertenwissen in den Lernprozess einbringen können, und die Indikatoren, mit denen die Auswirkungen auf das Lernverhalten und den Trainingsaufwand beobachten lassen, in den Vordergrund gestellt. Abschließend wurde gezeigt, wie sich Selbstverstärkendes Lernen von der aktuellen Produktgeneration auf die nächste Übertragen lässt, um das Wissen über die Konstruktionsaufgabe, welches im DQN implizit enthalten ist, wiederverwenden zu können. Im folgenden Kapitel wird die Umsetzung dieser Konzepte dargelegt.

6 Die Umsetzung des Modells und des Automatisierungswerkzeugs

In diesem Kapitel wird die Umsetzung des beschrieben Konzepts für die Modellierung der Anpassungskonstruktion und die Automatisierung der Konstruktionsaufgaben durch Selbstverstärkendes Lernen dargestellt. Die Modellierung erfolgt eigenständig, da auch andere Automatisierungswerkzeuge für die Automatisierung der einzelnen Konstruktionsaufgaben eingesetzt werden können. Der generische Aufbau der Umsetzung ist in Bild 52 skizziert. Dort sind links die eigenständigen Applikationen für die Wissensrepräsentation und das Selbstverstärkende Lernen gezeigt. Sowohl die Modellierungals auch die Automatisierungsapplikationen sind über Schnittstellen (im Englischen Application Programming Interface, API) in die Entwicklungsumgebung der Anwendenden integriert, damit diese nicht verlassen werden muss. In der Entwicklungsumgebung dient ein *Graphical User Interface* (GUI) für den Dialog mit den Anwendenden. Im Folgenden wird die Umsetzung exemplarisch in das CAx System Siemens NX 1899 integriert. Dabei wird als Synthesekomponente die CAD-Umgebung von NX und als Analysekomponente Simcenter Nastran eingesetzt. Die Wissensrepräsentation der in Unterabschnitt 5.1.2 dargelegten Konzepte für den Anpassungsprozess wird in Protégé v. 5.5.0 erstellt. Die Implementierung des Selbstverstärkenden Lernens erfolgt in Python v. 3.8. Die weiteren Spezifikationen werden im Anhang in Tabelle 13 zur Verfügung gestellt.

Die Erläuterung der Umsetzung ist aus Sicht der Anwendenden aufgebaut. Daher wird nach der Nutzerinteraktion und den dazugehörigen GUIs strukturiert. Die Implementierung der einzelnen Applikationen und deren Komponenten wird auf einem generischen Level beschrieben. Für die Umsetzung in Siemens NX kann der *Source Code* im Anhang ab Seite 161 eingesehen werden. Zunächst wird in Abschnitt 6.1 das Vorgehen der Modellerstellung, welches den Anpassungsprozesses in automatisierbare Konstruktionsaufgaben zerlegt, beschrieben (s. GUI M in Bild 52). Anschließend wird in Abschnitt 6.2 das Selbstverstärkende Lernen zur Automatisierung einer Konstruktionsaufgabe eingesetzt. Die drei maßgeblichen Interaktionen (1. Beschreibung der Konstruktionsaufgabe in Unterabschnitt 6.2.2, GUI K, 2. Einstellen der Hyperparameter in Unterabschnitt 6.2.3, GUI H und 3. Evaluation der Indikatoren in Unterabschnitt 6.2.4, GUI I) werden in den jeweiligen Unterkapiteln dargestellt.

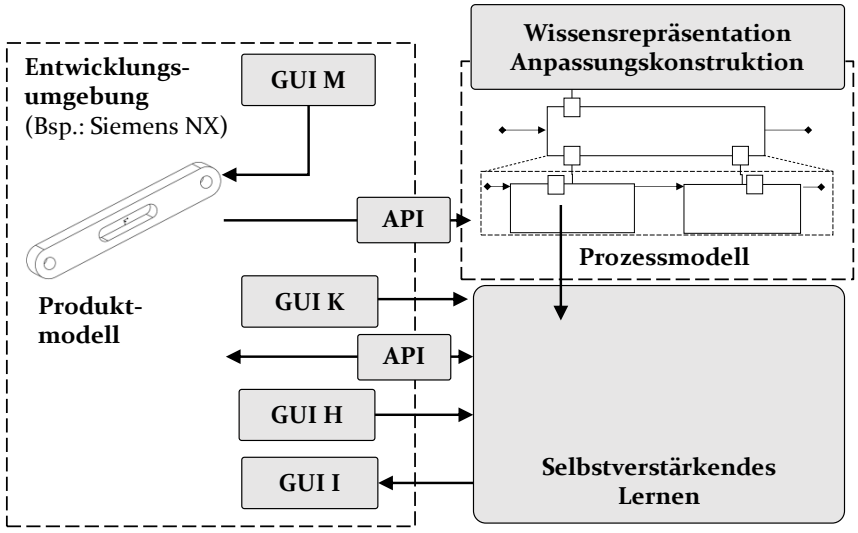
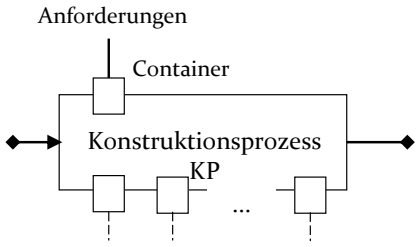


Bild 52: Umsetzung des Selbstverstärkenden Lernens aus Sicht der Anwendenden innerhalb einer Entwicklungsumgebung mit vier GUIs für die einzelnen Nutzerinteraktionen

6.1 Die Modellerstellung

Die Modellerstellung dient der Instanziierung des zu automatisierenden Anpassungsprozesses in der Wissensrepräsentation mit Hilfe der in Unterabschnitt 5.1.2 vorgestellten Konzepte und Axiome für Zustände und Transitionen. Ausgangspunkt sind das Produktmodell des Vorgängerproduktes, dessen Prinzipielle Lösung als fester Meilenstein, der bei der Anpassungskonstruktion nicht geändert werden darf, und neue Anforderungen, die zur Anpassung des Produktes führen. Ziel des Modells ist eine Beschreibung des Anpassungsprozesses durch eine Folge von Konstruktionsaufgaben (s. (a) in Bild 53). Diese bestimmen wann und zu welchem Zweck die Merkmale des Produktes angepasst werden sollen. Daher repräsentieren Konstruktionsaufgaben eine Relation zwischen Anforderungen und Merkmalen. Sie sollen anschließend automatisiert werden. Da üblicherweise mehr als eine Konstruktionsaufgabe notwendig ist, wird der Anpassungsprozess chronologisch und hierarchisch strukturiert. Dazu werden Konstruktionsprozesse verwendet, die die Relationen zwischen den Anforderungen repräsentieren. Wie im vorangegangen Kapitel erörtert, werden Anforderungen in Container ablegt. Diese werden als Zweck bezeichnet. Folglich hat ein Konstruktionsprozess immer einen Eingangscontainer und zwei oder mehr Ausgangscontainer für Anforderungen (s. (b) in Bild 53), damit eine Untergliederung der Anforderungen stattfinden kann. Da sich die einzelnen Anpassungsprozesse und deren Prinzipielle Lösung in den Formaten stark unterscheiden können und



KP: Anforderungen → Anforderungen

(b) Konstruktionsprozess

Bild 53: Darstellung des Unterschieds der Konzepte zwischen Konstruktionsaufgabe und Konstruktionsprozess

6.1.1 Die Applikation für die Modellerstellung

Die Applikation orientiert sich am SeED (*Semantic Integration for Engineering Design*) Ansatz [P6] und besteht aus vier Komponenten, welche in Bild 54 dargestellt sind. Da die Wissensrepräsentation durch eine Ontologie realisiert wird, ist die erste Komponente die sogenannte T-Box und die zweite Komponente die sogenannte A-Box. Die T-Box (T: *Terminology*) enthält die Konzepte und Axiome für Zustände und Transitionen (vgl. Tabelle 4 auf Seite 75 und Tabelle 5 auf Seite 76). Sie bildet das Klassenmodell der Wissensrepräsentationen und ist fest implementiert. Die A-Box (A: *Assertion*) enthält die Instanzen eines konkreten Anpassungsprozesses. Sie werden für jeden Anpassungsprozess auf Basis der Prinzipiellen Lösung und der Reihenfolge der Konstruktionsaufgaben durch die Anwendenden erstellt. Damit diese bei der Erstellung so gut wie möglich unterstützt werden, werden die Konzepte des Modells der Anpassungskonstruktion über ein Annotationsschema und eine GUI in das CAD System integriert. Die dabei entstehende Beschreibung des Anpassungskonstruktionsprozesses in der CAD Umgebgung wird durch die in [P7] beschriebene API automatisch mit der Wissensrepräsentation synchronisiert. Mit Hilfe der GUI M (M: Modellierung) werden Anwendenden schrittweise unterstützt, die Informationen des Anpassungsprozesses anzulegen. Der Prozess der Anwenderinteraktion wird anschließend beschrieben. Zunächst soll die Verarbeitung der Eingaben erläutert werden.

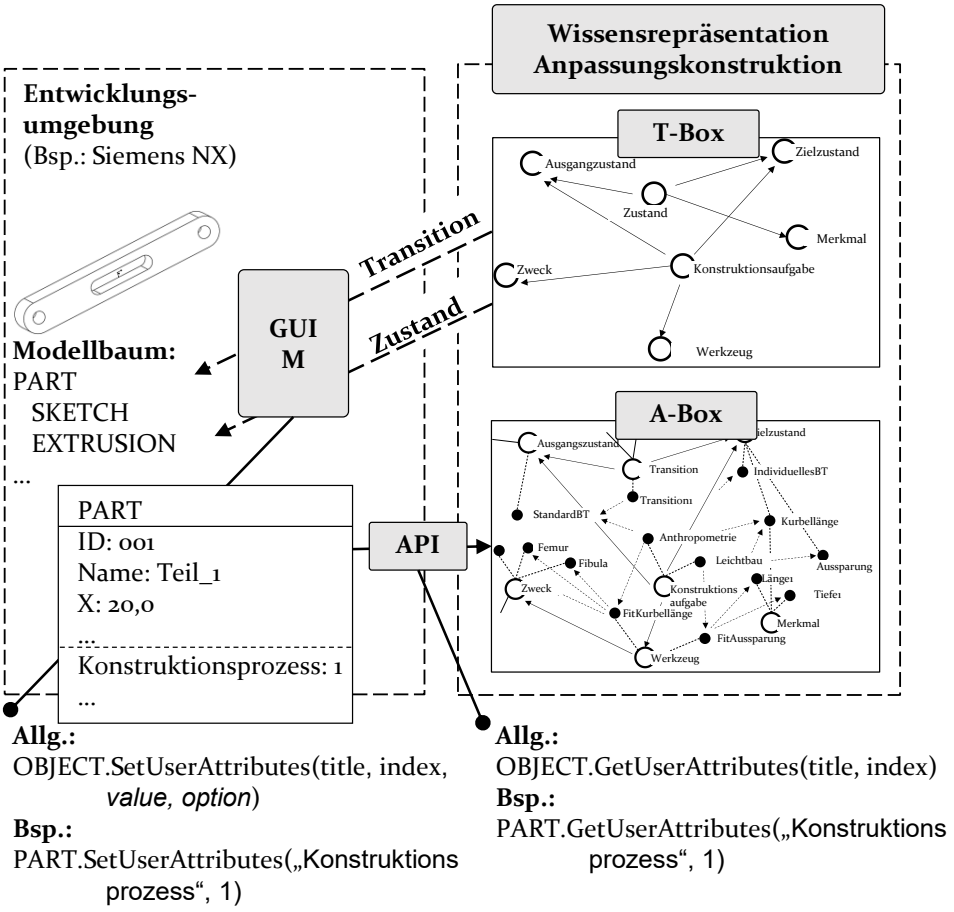


Bild 54: Darstellung der vier Komponenten der Modellierungsapplikation mit dem Klassenmodell der T-Box und einer exemplarisch befüllten A-Box durch die Instanzen des Produktmodells

Die Eingaben der Anwendenden werden durch ein Annotationschema verarbeitet, welches die geometrischen Informationen des Produktes mit den Informationen des Anpassungsprozesses verknüpft. Diese zusätzlichen Informationen ergänzen die Bedeutung eines geometrischen Objekts durch deren Bedeutung für den Anpassungsprozess. Daher wird dieser Prozess als semantische Anreicherung bezeichnet. Das geometrische Objekt wird als Anker bezeichnet, an das zusätzliche Semantik angehängt wird. Hier werden in der Praxis zusätzliche Attribute in die geometrischen Objekte der Entwicklungsumgebung geschrieben. Sie werden in NX UserAttributes genannt und sind in anderen CAD System ebenso üblich. Neben der Darstellung der vier Komponenten wird in Bild 54 exemplarisch auf dem Bauteil (PART) als geometrisches Objekt annotiert. In dessen UserAttributes wird ein zusätzliches Attribut „Konstruktionsprozess 1“ über den SetUserAttributes(titel, index,

value, *option*) Befehl integriert. Hier werden die UserAttributes so angelegt, dass der Titel ein Term aus der Ontologie ist und damit von dieser erkannt wird und eine Bedeutung erhält. Der Index wird zur Identifizierung der Instanzen des gleichen Terms verwendet. *Value* und *option* sind optional. Dieses kann durch GetUserAttributes(titel, index) ausgelesen werden. Beide Befehle sind für alle NX Objekte verfügbar.

Die Konzepte der T-Box der Wissensrepräsentation unterteilen sich in Terme für Zustände und Transitionen. Diese werden an unterschiedlichen geometrischen Objekten annotiert. Transitionen (Konstruktionsaufgabe, Konstruktionsprozess) beschreiben Zustandsänderungen sowohl chronologisch als auch hierarchisch. Die Chronologie dient der Modellierung der Reihenfolge. Die Hierarchie dient der Zuordnungen von Anforderungen zu Merkmalen. Da sich Transitions auf das gesamte Produkt beziehen, werden sie an das oberste Element im Modellbaum annotiert (s. PART in Bild 54).

Die Zustände beschreiben das Produktmodell zu einem Zeitpunkt im Anpassungsprozess. Eine Instanz des Konzepts Zustand muss daher alle Merkmale sowie deren Ausprägungen zu einem definierten Zeitpunkt repräsentieren. Die Merkmale des Produkts werden über die API automatisch in der A-Box als Merkmalsinstanz instanziiert. Hierfür können in NX alle Ausdrücke, welche für ein Objekt definiert wurden, abgefragt werden (OBJECT.Expressions). Die Ausdrücke untergeordneter Objekte werden dabei mitberücksichtigt. Ein Ausdruck besteht aus einem eindeutigen Namen, einem Typ, einem Wert, einer Einheit und optionalen Parameter, wie beispielsweise einer Formel. Ein Beispiel für eine Linie mit Länge 15 mm ist „Linie_01, Linie, 15, mm“ (s. Bild 55). Der Wert kann sich aus einer Referenz ergeben. Eine zweite Linie

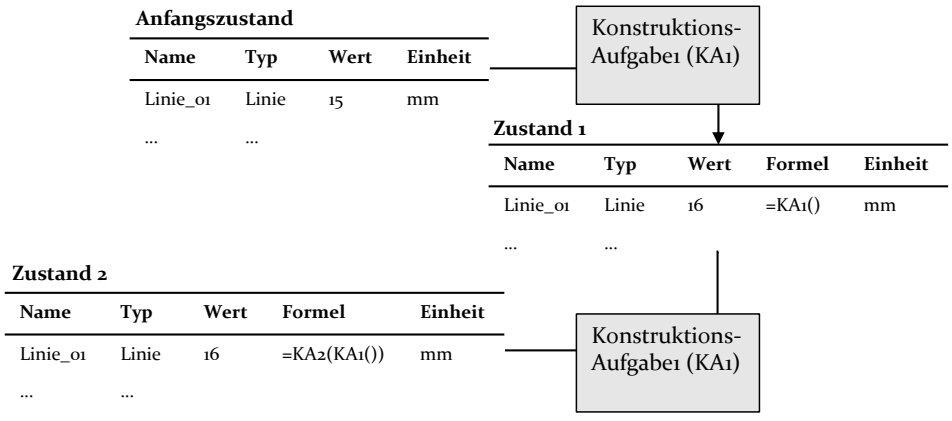


Bild 55: Darstellung dreier exemplarischer Zustände

Über den Anpassungsprozess ändert sich die Ausprägung des Merkmals, d.h. der Wert ist variabel. Er wird darüber bestimmt, welche Konstruktionsaufgaben vor dem Zeitpunkt des Zustandes bereits bearbeitet wurden. Folglich referenziert der Wert die Konstruktionsaufgaben, die bis zu diesem Zeitpunkt bereits erfolgt sind (s. Bild 55). Der Ausgangszustand ist automatisch der initiale Wert, bspw. „15". Hier ist keine Referenz einer Konstruktionsaufgabe im Wert enthalten. Dieses Merkmal wird nach Konstruktionsaufgabe 1 auf die Formel „=KA1()" ausgedrückt. Bei der Automatisierung des Anpassungsprozesses ist hier das entsprechende Automatisierungswerkzeug, welche für vorgegebene Anforderungen die Ausprägung des Merkmals zurück gibt. Analog ist der Wert nach Konstruktionsaufgabe 2 „=KA2(KA1())". Damit keine Inkonsistenzen entstehen können, werden die Annotationen nach der chronologischen Reihenfolge verschachtelt. Dies ist exemplarisch in Bild 55 dargestellt.

Bisher wurde dargelegt, wo die zusätzlichen Informationen über den Anpassungsprozess abgelegt werden. Nachfolgend wird gezeigt, wie die Anwendenden über die GUI M bei diesem Prozess unterstützt werden. Zuvor soll kurz gezeigt werden, wie die geometrischen Elemente in die A-Box der Ontologie geladen werden, damit sie ihre Bedeutung für den Anpassungsprozess erhalten. Zur Synchronisation zwischen annotiertem Produktmodell und Wissensrepräsentation wird der Befehl GetUserAttributes(titel, index) in einer Schleife über alle geometrischen Objekte des Modellbaums ausgeführt. Die abgefragten UserAttribute werden in Tabellen zwischengespeichert. Damit sie von dort über eine vorgeschriebene Regel automatisch in Protégé instanziiert werden können [P6]. Das Zwischenspeichern ist nicht zwangsläufig notwendig, erlaubt jedoch den Einsatz von vorgefertigten Tools wie beispielsweise Cellfie[1], welches automatisiert Daten aus einer tabellarischen Repräsentation in als Instanzen der A-Box abbilden kann.

6.1.2 Die Anwenderinteraktion bei der Modellerstellung

Das Modellieren des Anpassungsprozesses erfolgt über eine in die Entwicklungsumgebung integrierte GUI M. Dazu kann die Applikation in der Entwicklungsumgebung gestartet werden. Beim Start werden die Anwendenden zunächst gefragt, ob ein neuer Anpassungsprozess modelliert werden soll oder ein bestehender angepasst werden soll. Beim Anlegen eines neuen Prozesses muss ein Arbeitsverzeichnis definiert werden. Dieses Arbeitsverzeichnis dient zur Synchronisation mit Protégé. Daher darf nur ein Modell pro Arbeitsverzeichnis existieren. Beim Erstellen eines neuen Modells wird die „model.xml" im Arbeitsverzeichnis angelegt. Diese koordiniert das Modell sowie dessen

[1] https://github.com/protegeproject/cellfie-plugin

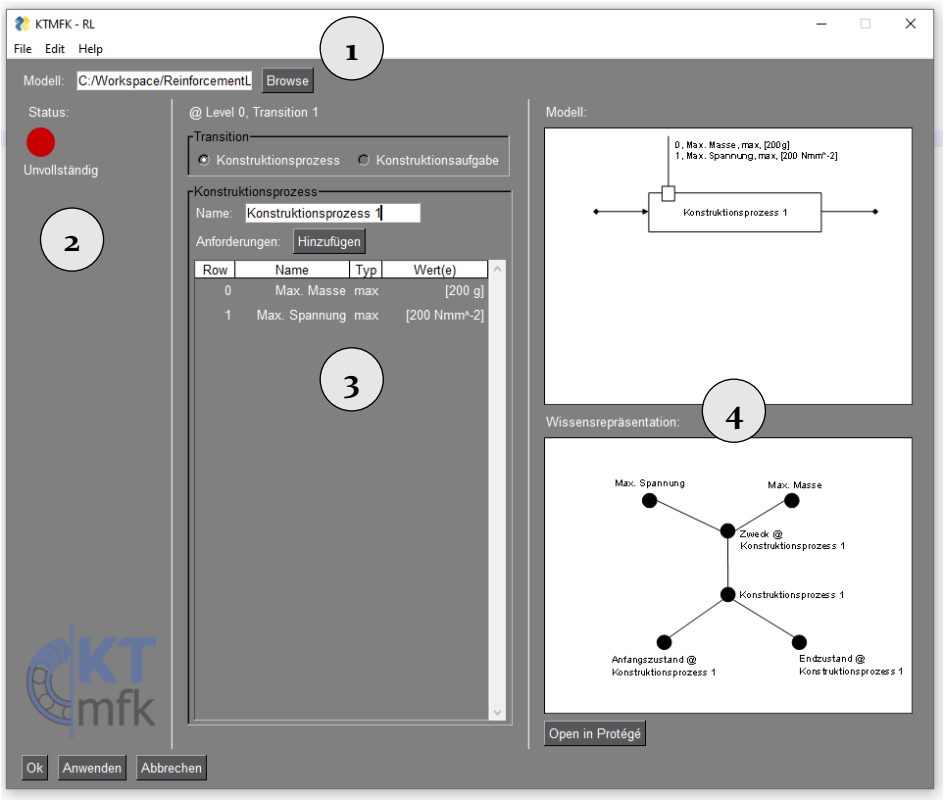


Bild 56: Darstellung der GUI für Konstruktionsprozess

Die graphische Anwenderschnittstelle ist dreigeteilt. Links befindet der Status Tab (② in Bild 56). Dort wird mit einer Ampel signalisiert, welchem Zustand das Modell hat. Wenn alle Anforderungen über Konstruktionsaufgaben mit Merkmalen verknüpft sind, wird das Modell als vollständig interpretiert und der Status „Ok“ mit einer grünen Ampel signalisiert. Es wird nur auf Vollständigkeit der Eingaben geprüft. Die Korrektheit der Angaben kann ohne Prinzipielle Lösung nicht geprüft werden und soll auch nicht die Aufgabe dieser Applikation sein. In der Mitte befindet sich die Interaktionsfläche für die Anwendenden (③ in Bild 56). Zuerst wird die Position im Modell angegeben. In Bild 56 ist „Level 0, Transition 1“ genannt. Jede Transition kann entweder ein Konstruktionsprozess oder eine Konstruktionsaufgabe sein. Daher wird dies zunächst in einer Entweder-Oder-Abfrage abgefragt. Diese bestimmt die darunter liegende Interaktionsfläche für den Konstruktionsprozess (wie in Bild 56 dargestellt) oder die Konstruktionsaufgabe. Auf der

rechten Seite befindet sich der Visualisierungstab (④ in Bild 56). Dort wird das aktuelle Modell sowie die Instanz des aktuell bearbeiteten Objekts in der Wissensrepräsentation dargestellt.

Der Aufbau ist so gestaltet, dass die Anwendenden durch den Modellierungsprozess geleitet werden. Das Vorgehen ist in Bild 57 schematisch dargestellt. Beim Erstellen eines neuen Anpassungsprozesses ① in Bild 57, wird automatisch der Ausgangszustand, der Zielzustand und eine Transition angelegt. Sie werden auf dem höchsten Objekt im Modellbaum annotiert. Der Nutzer wird aufgefordert, alle Anforderungen an den Anpassungsprozess zu definieren. Diese werden ebenfalls dort annotiert. Zusammen ergeben sie das kleinste mögliche Modell aus einer Transition und zwei Zuständen. Nun wird der Nutzer von Transition zu Transition geleitet. Die Zustände werden automatisch generiert. Die Annotationen werden ebenfalls automatisch erzeugt. Eingaben werden über „Anwenden" bestätigt. Dies führt zur Aktualisierung des Modells und der Wissensrepräsentation. Bei der Aktualisierung wird überprüft, ob alle Anforderungen über Konstruktionsaufgaben zu Merkmalen verknüpft wurden. Ist dies der Fall, ist der Status vollständig und wird durch eine grüne

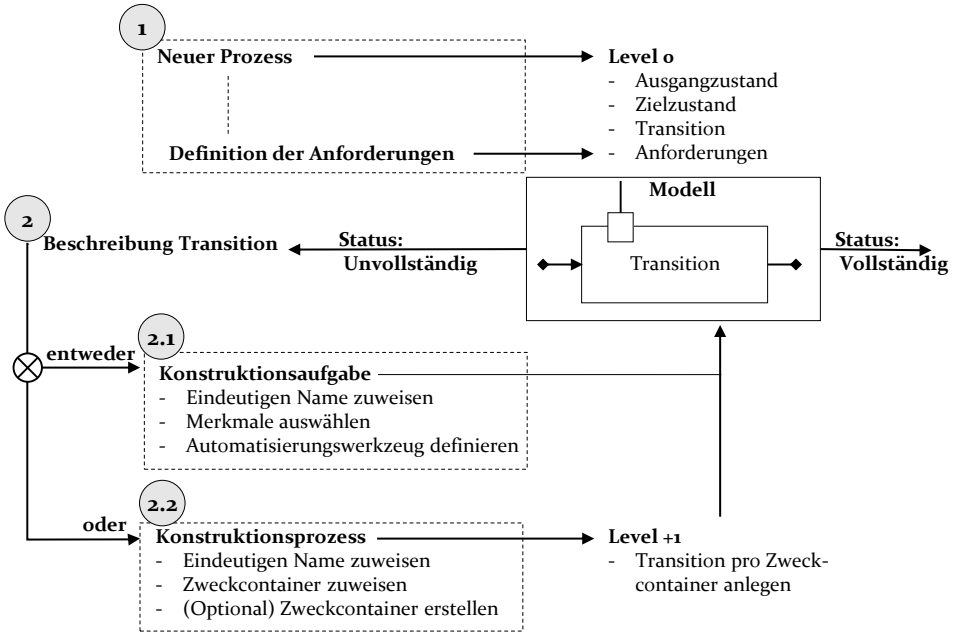


Bild 57: Darstellung des Annotationsprozesses

Zunächst muss festgelegt werden, ob es sich bei der Transition um eine Konstruktionsaufgabe oder um einen Konstruktionsprozess handelt (s. ② in Bild 57). Für eine Konstruktionsaufgabe werden alle Ausdrücke des Produktmodells abgefragt und den Anwendenden tabellarisch aufgeführt (s. Bild 56). Die Anwendenden müssen der Konstruktionsaufgabe einen eindeutigen Name geben und den Pfad zum Automatisierungswerkzeug angeben (s. 2.1 in Bild 57). Außerdem müssen die Merkmale, die auf Basis des Zweckes der Konstruktionsaufgabe und deren Automatisierungswerkzeug angepasst werden, sollen in der Tabelle markiert werden (③ in Bild 56). Die Referenzen der Konstruktionsaufgabe wird auf den Merkmalen des Produktmodells annotiert. Die Konstruktionsaufgabe wird auf dem höchsten Objekt des Modellbaums annotiert. Die Eingaben werden durch „Anwenden“ bestätigt und das Modell aktualisiert. Wenn der Status nun vollständig ist, ist die Modellierung abgeschlossen und das Modell und die Wissensrepräsentation können gespeichert werden. Andernfalls geht die Applikation automatisch zur nächsten Transition über. Für einen Konstruktionsprozess müssen die ausgangsseitigen Zweckcontainer angegeben und befüllt werden (s. 2.2 in Bild 57). Es werden mindestens zwei Container vorgegeben. Über „Hinzufügen“ können beliebig viele Container hinzugefügt werden. Die Anforderungen des Eingangs müssen diesen anschließend zugewiesen werden. Die Eingaben werden durch „Anwenden“ bestätigt, das Modell wird aktualisiert und auf Vollständigkeit geprüft.

6.1.3 Zusammenfassung zur Modellerstellung

Die Modellerstellung dient der computer-verarbeitbaren Darstellung des zu automatisierenden Anpassungsprozesses. Die Anwendenden werden durch die GUI M durch den Modellierungsprozess (vgl. Bild 57 auf Seite 112) unterstützt. Das Modell dient zur Verknüpfung von Anforderungen zu Merkmalen. Da diese häufig nicht eindeutig sind, d. h. eine Anforderung kann sich auf viele Merkmale beziehen und umgekehrt, wird der Anpassungsprozess hierarchisch und chronologisch strukturiert. Die Hierarchie für so lange untergeordnete Konstruktionsprozesse ein, bis die Konstruktionsaufgaben die Verknüpfung von Anforderungen zu Merkmalen realisieren können. Das erstellte Modell wird in der GUI M visualisiert und automatisch in der Wissensrepräsentation instanziiert.

6.2 Die Steuerung des Selbstverstärkenden Lernprozesses

Zur Automatisierung einer Konstruktionsaufgabe des Anpassungsprozesses mit Hilfe des Selbstverstärkenden Lernens muss für diese ein trainierter Agent vorliegen. Dieser wird als Automatisierungswerkzeug in der Konstruktionsaufgabe verknüpft. Der Agent dient dazu, das Verhalten der Produktentwickelnden zu imitieren. Dazu muss er zunächst die Konstruktionsaufgabe, deren Anforderungen und Randbedingungen erhalten. Anschließend müssen Stellgrößen des Agenten von den Anwendenden eingestellt werden, damit nach dem Training das gelernte Verhalten evaluiert werden kann. Obwohl diese Schritte bereits in verwandten Werkzeugen so oder in abgewandelter Form zum Einsatz kommen, wie beispielsweise das Vorgeben von einzelnen Zielfunktionen bei der Mehrzieloptimierung, ist keiner der Schritte als selbstverständlich für die Produktentwickelnden zu betrachten. Damit die Anwendenden beim Einsatz des Selbstverstärkenden Lernens unterstützt werden, wurden drei GUIs entwickelt. Deren Umgang wird in diesem Abschnitt erläutert. Damit dies möglich ist, muss zunächst die Architektur des Werkzeugs vorgestellt werden.

6.2.1 Die Architektur des Automatisierungswerkzeugs

Unter anderem zur besseren Übersicht wurde die Architektur in Python objektorientiert implementiert. Außerdem können so mehrere Objekte einer Klasse instanziiert werden. Dies ist beispielsweise für die Anforderungsobjekte oder die Tiefen Neuronalen Netze erforderlich. Das zugehörige UML Diagramm ist in Bild 58 dargestellt. Die Attribute und Methoden der Klassen wurden ausgeblendet, um die Architektur in einer Darstellung zeigen zu können. Sie werden anschließend in Unterabschnitt 6.2.2 und Unterabschnitt 6.2.3 gezeigt. Dort werden die GUIs ebenfalls vorgestellt.

Da das Konzept auf dem Verständnis des Selbstverstärkenden Lernens nach Sutton und Barto aufbaut, sind Environment und Agent die zentralen Klassen der Architektur [32]. Objekte der Klasse Environment definieren die Lernumgebung des Agenten, sie erhalten von den Anwendenden die maximalen und minimalen Werte für die Grenzen des Lösungsraums. Außerdem beinhaltet jedes Environment ein Objekt der Klasse RewardFunction. Als Teil der Lernumgebung spiegelt die RewardFunction die Eigenschaften des Produktes in Belohnungen wider. Diese berechnen sich aus den Anforderungen der Konstruktionsaufgabe. Die Interaktion mit der Lernumgebung sowie deren Aufgaben werden in Unterabschnitt 6.2.2 beschrieben. Objekte der Klasse Agent definieren einen Agenten als Akteur in der Umgebung. Er benötigt a priori definierte Hyperparameter, die als Stellschrauben von den Anwendenden eingestellt werden müssen. Für den verwendeten Q-Learning

Algorithmus ist es sinnvoll, die Tiefen Neuronalen Netze in eigene Objekten zu instanziieren. So können jederzeit die aktuellen Gewichte der Evaluation und des Target DQN abgefragt werden. Das Einstellen der Hyperparameter und die Aufgaben des Agenten werden in Unterabschnitt 6.2.3 beschrieben. Die Evaluation des Lehrverhalten wird gesondert dargelegt, da sie nach dem Training stattfindet. Die in Abschnitt 5.5 vorgestellten Indikatoren werden in GUI I den Produktentwickelnden bereitgestellt. Der Umgang mit GUI I wird in Unterabschnitt 6.2.4 abschließend beschreiben.

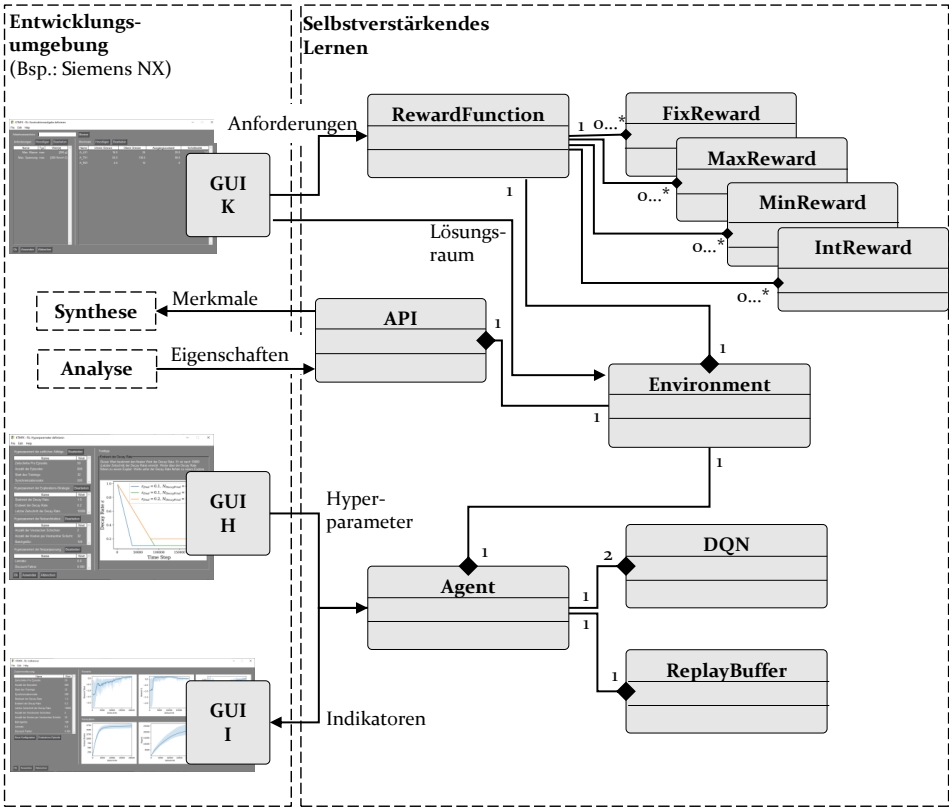


Bild 58: Darstellung der Architektur des Automatisierungswerkzeugs für das Selbstverstärkende Lernen als UML-Diagramm. Die Attribute der Klassen wurden aus Gründen der Übersicht nicht dargestellt und werden in Bild 59 und Bild 60 beschrieben

6.2.2 Die Anwenderschnittstelle zur Übergabe von Konstruktionsaufgaben

In GUI K (K:Konstruktionsaufgabe definieren) werden die Anforderungen sowie der Lösungsraum an die Lernumgebung übergeben (s. Bild 59). Hierzu stellt die GUI K zwei Tabellen dar. In der linken Tabelle entspricht eine Zeile einer Anforderungen, in der rechten Tabelle entspricht eine Zeile einem Merkmale. Beide können entweder mit Hilfe der Angabe eines Arbeitsverzeichnisses automatisch aus einer Konstruktionsaufgabe geladen werden. Dann müssen lediglich die Informationen über „Bearbeiten" eingefügt werden. Alternativ können manuell Zeilen über „Hinzufügen" angelegt werden. Dabei muss beachtete werden, dass der eindeutige Name des Merkmals in den Ausdrücken des Produktmodells zu finden ist. Über „OK" werden die Informationen aus den Tabellen an das Automatisierungswerkzeug übergeben, welches die Objekte wie folgt instanziiert und den Anwendenden GUI H einblendet (s. Unterabschnitt 6.2.3).

Verarbeitung der Anwendereingaben aus GUI K

Die Anwendereingaben werden bei der Instanziierung der Objekte verarbeitet. Es wird jeweils ein Objekt der Klasse Environment, RewardFunction sowie API angelegt. Zusätzlich werden entsprechend der Anforderungen FixReward, MinReward, MaxReward und IntReward Objekte angelegt (s. Bild 59). Eine Anforderung besteht aus dem eindeutigen Namen, dem Typ der Anforderung (Mindest-, Maximal, Intervall-, oder Festforderung; s. Tabelle 7 auf Seite 90), und den entsprechenden Eingabewerten, ab welchem Wert die Anforderung erfüllt ist. Optional kann eine Normalisierung und eine Gewichtung angegeben werden. Andernfalls wird Pareto-Normalisiert und gleich gewichtet. Diese Informationen werden in den Attributen der RewardFunction abgelegt. Die einzelnen Anforderungen werden je nach Typ einem Objekt angelegt, beispielsweise ein Objekt der Klasse MaxReward für eine Maximalforderung. Zusammen ergeben sie eine Liste in dem RewardFunction Objekt, welches über die Methode *calc_reward()* aus übergebenen Eigenschaften die individuellen Belohnungen und deren gewichtete Summe berechnet.

Ein Merkmal besteht aus einem eindeutigen Namen, sowie einer oberen und einer unteren Grenze, welche den Bauraum limitieren und den Lösungsraum beschränken. Damit Fertigungsparameter direkt berücksichtigt werden und diskrete Werte den Lösungsraum bilden, kann optional die Schrittweite angegeben werden. Außerdem kann der Ausgangszustand – die Parameterkombination, mit der Agent beginnen soll – optional definiert werden. Diese Informationen werden in dem Environment Objekt als Attribut *parameter_grid* hinterlegt. Hierzu werden sogenannte Pandas DataFrames verwendet, da

diese Tabellen eine Vielzahl von Tabellenoperationen verarbeiten können. Die enthaltenen Informationen werden genutzt, um die restlichen Attribute des Environment Objekts zu instanziieren. Dabei ist das zentrale Attribut des Environments der *state*. Dieser ist ein Vektor, welcher auf die aktuelle Parameterkombination im Lösungsraum zeigt. Er wird unter anderem von dem Objekt der Klasse API genutzt, um damit das Produktmodell zu aktualisieren und über die Analyse die Eigenschaften zu erhalten. Parameterkombination und resultierende Eigenschaften werden in der *fem_results* Tabelle mitgeschrieben. Die Synthese und Analyse Komponenten sind von der genutzten Entwicklungsumgebung abhängig. Für diese Umsetzung wurde ein NXOpen Skript angelegt, welches die Schritte der Produktentwickelnden in der CAx

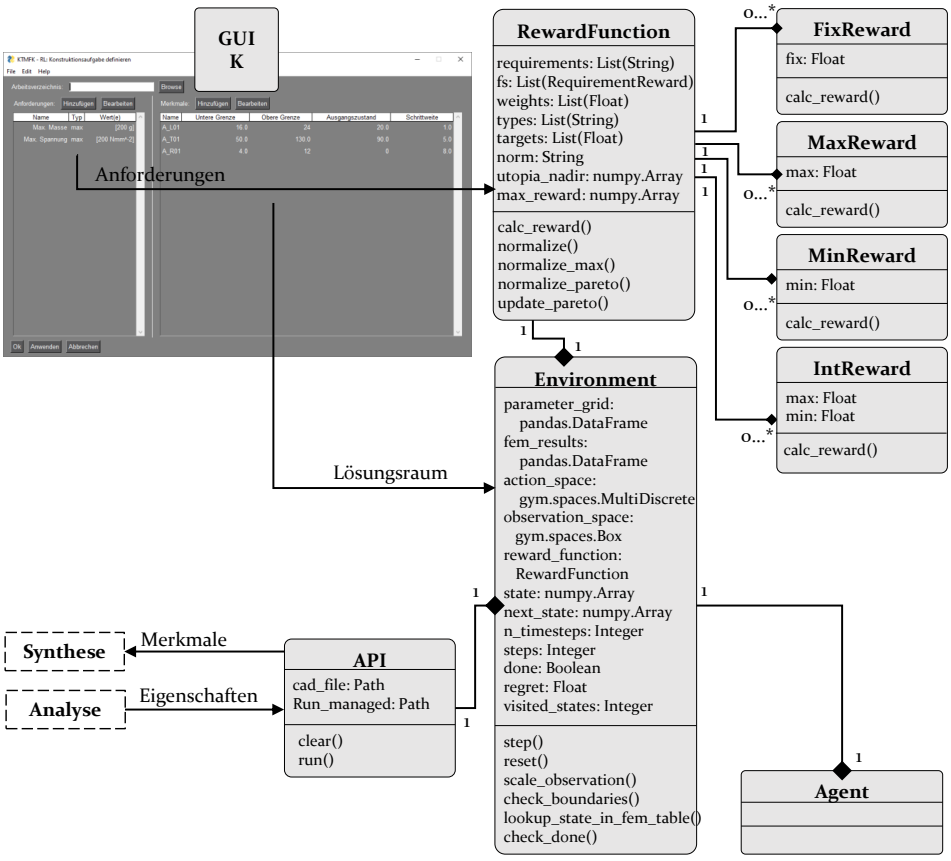


Bild 59: Darstellung der Übergabe einer Konstruktionsaufgabe durch deren Anforderungen und Merkmale an die Lernumgebung sowie deren untergeordnete Klassen, Attribute und Methoden

Die Lernumgebung dient aus Sicht des Selbstverstärkenden Lernens dazu, diskrete Zustände abzubilden, die über Aktionen des Agenten verändert werden können und dieser dabei eine direkte Belohnung erhält. Aus Sicht der Produktentwicklung ergibt sich die Belohnung aus den Eigenschaften. Um die Eigenschaften zu erhalten, müssen die Merkmale über die Analyse berechnet, simuliert oder erprobt werden. Je nach Absicherungsmethode entsteht hier der größte Aufwand für das Training. Daher sollten nur notwendige Parameterkombinationen entsprechend analysiert werden. Bereits bekannte oder unbrauchbare Kombinationen sollten nicht berücksichtigt werden. Bevor die Merkmale der API übergeben werden, überprüft die Methode *lookup_state()*, ob diese Parameterkombination bereits abgesichert wurde. Falls dies der Fall ist, wird das Ergebnis der Analyse aus der *fem_results* übernommen. Anderenfalls überprüft *check_boundaries()*, ob sich die Merkmalskombination noch im Lösungsraum befindet. Falls dies ebenfalls zutrifft, wird *visited_states* um 1 erhöht und die Kombination analysiert.

6.2.3 Die Anwenderschnittstelle zur Einstellung der Hyperparameter

Nachdem die Konstruktionsaufgabe definiert wurde, müssen die Hyperparameter eingestellt werden. Bei dieser kritischen Aufgabe müssen die Anwendenden unterstützt werden, da keiner der Hyperparameter selbsterklärend ist. In GUI H (H: Hyperparameter definieren, s. Bild 60) befindet sich links die Zusammenfassung der Hyperparameter und rechts deren Erklärung und, wenn dies möglich ist, deren Visualisierung. Über „Bearbeiten“ können die Werte eingestellt werden. Den Anwendenden werden Werte für die Hyperparameter in der Grundeinstellung bereitgestellt. Diese orientieren sich an der Größe des Lösungsraums. Sie sind jedoch Richtwerte, die nur ermöglichen sollen, dass bei der Bestätigung des Hyperparametern mit „OK“ der Agent das Training beginnen kann. Durch die Bestätigung wird Agent und alle zugehörigen Objekte mit den eingestellten Hyperparametern instanziiert.

Verarbeitung der Anwendereingaben aus GUI H

Die Eingaben aus GUI H werden verwendet, um den Agenten und dessen untergeordnete Klassen zu instanziieren. Für ein Objekt der Klasse Agent werden zwei Objekte der Klasse DQN instanziiert. Die Evaluation DQN und das Target DQN. Zu Beginn des Trainings sind beide identisch. Der ReplayBuffer ist die Historie des Agenten. Er speichert alle Erfahrungen als Tupel aus. Der Agent führt die Aktionen in der Umgebung aus und lernt sie, indem er die Gewichte des Evaluation DQN anpasst. Er ist eine Umsetzung des Q-Learning Algorithmus nach WATKINS & DAYAN [199].

Die Methode *play_game()* führt das Training aus. Diese durchläuft zwei Schleifen, die äußere über die Episoden und die innere über die Zeitschritte pro Episode. Zu Beginn jeder Episode wird durch die Methode *reset()* der Ausgangszustand eingestellt. Jeder Zeitschritt beschreibt die Interaktion zwischen Umgebung und Agent. Auf Basis des Zustands der Umgebung wählt

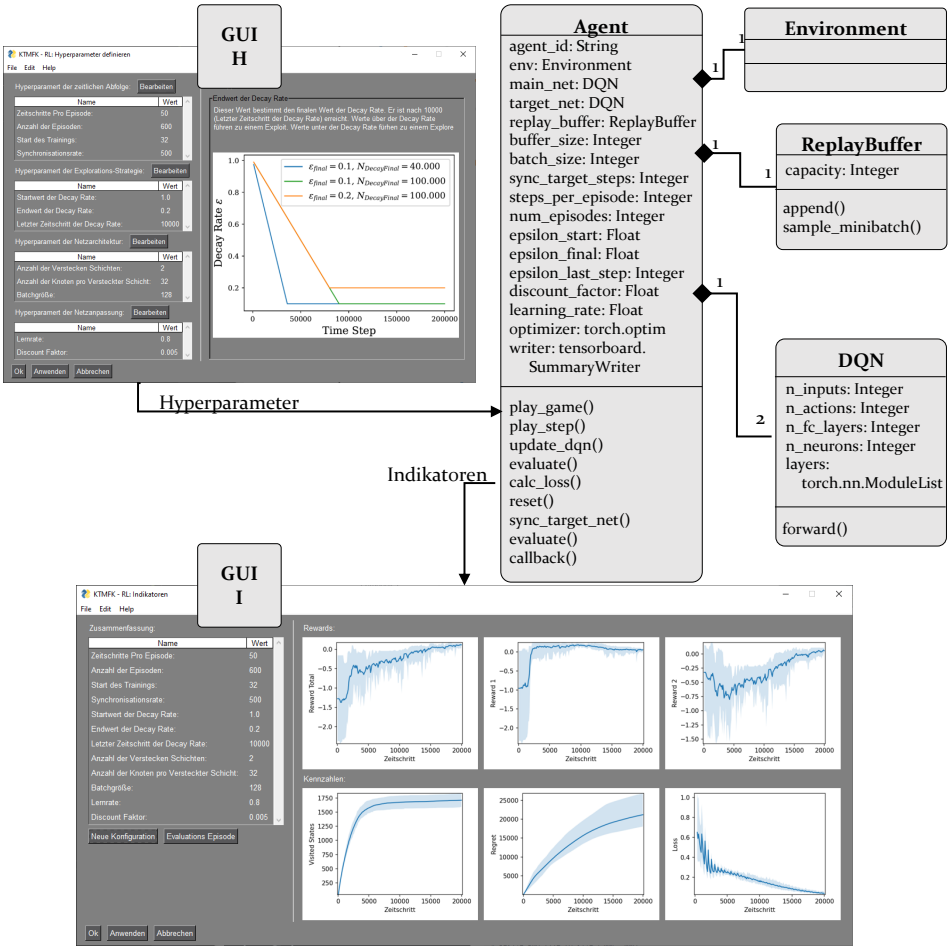


Bild 60: Darstellung der Übergabe der Hyperparameter an den Agent sowie dessen untergeordnete Klassen, Attribute und Methoden und der Zusammenfassung des Trainings

Bei einer Exploration wird eine zufällige Aktion ausgeführt. Bei einem Exploit wird das Evaluation DQN benutzt, um die Aktion mit der größten Aussicht auf hohe, zukünftige Belohnungen vorherzusagen (*forward()* Methode). Die Aktion wird in der Lernumgebung ausgeführt und zusammen mit dem erreichten

Zustand und der erhaltenen Belohnung im ReplayBuffer abgelegt (*append()* Methode). Anschließend wir das Evaluation DQN über *update_dqn()* aktualisiert. Ein Batch wird über die *sample_minibatch()* Methode aus dem ReplayBuffer geniert und damit die *calc_loss()* Methode aufgerufen. Diese berechnet aus dem Batch mit Hilfe der Gleichung 11 (s. Seite 54) die Anpassung des Evaluation DQN. Die Methoden *callback()* und *evaluate()* dienen der Evaluation des Trainings und werden nachfolgend in GUI I erörtert.

6.2.4 Die Anwenderschnittstelle zur Evaluation der Indikatoren

Nach dem Training werden die Ergebnisse den Anwendenden zur Evaluation in der GUI I (I: Indikatoren) bereitgestellt (s. Bild 60). Dort befindet sich in der linken Spalte die Zusammenfassung der Hyperparameter und rechts die Visualisierung der *callback()* Methode. Diese zeigt die in Abschnitt 5.5 vorgestellten Indikatoren für das Lernverhalten und den Lernaufwand. Über die Schaltflächen "Neue Konfiguration" und „Evaluations Episode" kann eine neuer Trainingsdurchlauf gestartet oder eine Evaluations Episode durchgeführt werden. Ersteres führt die Anwendenden zurück zu GUI H. Zweiteres ruft ein weiteres Fenster auf und fordert die Anwendenden auf einen Zustand vorzugeben. Mit Hilfe der *evaluate()* Methode wird von diesem Zustand aus die gelernte Strategie verfolgt, bis die maximale bekannte Belohnung erreicht wurde oder die Episode beendet wird. Dabei werden die Gewichte des Evaluation DQN eingefroren und nur Exploits ausgeführt. Die Strategie kann als Folge von Zuständen und Aktionen hinsichtlich jeder individuellen Zielfunktion (Anforderung) oder der übergeordneten Zielfunktion visualisiert werden.

6.2.5 Zusammenfassung zur Steuerung des Selbstverstärkenden Lernprozesses

Bei der Anwendung von Selbstverstärkendem Lernen ist die gegenseitige Abhängigkeit von Erfahrungen mit der Lernumgebung und dem antrainierten Verhalten des Agent zu berücksichtigen. Erfahrungen beeinflussen die Anpassung der tiefen Neuronalen Netze, welches zukünftige Erfahrungen bestimmt. Falsch eingestellte Hyperparameter können dazuführen, dass Erfahrungen mit der Lernumgebung fehlen und nicht berücksichtigt werden können. Die Einstellung und Überwachung obliegt den Produktentwickelnden, da sie abschätzen können, ob der Agent ausreichend Erfahrungen mit der Lernumgebung gemacht hat. Ein mögliches Szenario ist ein Agent, welcher nicht den gewünschten Total Reward von 0,0 erreicht, also nicht alle Anforderungen gleichzeitig erfüllt. Hier müssen Produktentwickelnde entscheiden, welche Stellschrauben (Hyperparameter) angepasst werden, um

die Strategie des Agenten zu verbessern. Mit Hilfe der vorgestellten Indikatoren können Produktentwickelnde einsehen, ob der Explorationsphase zu kurz ist, der Agent zu kurzsichtig ist oder die Netzanpassung zu langsam für den Anwendungsfall ist. Diese Möglichkeiten müssen nacheinander in Betracht gezogen werden.

Die erst Möglichkeit ist vor allem dann wahrscheinlich, wenn der Agent keine Merkmalskombination gefunden hat, die alle Anforderungen gleichzeitig erfüllt. Folglich kann der Agent keine Strategie zu dieser erlernen. Hier müssen die Produktentwickelnden abschätzen, ob bessere Merkmalskombinationen im Lösungsraum vorliegen sollten. Falls dies der Fall ist, sollte der Exploration des Agenten mehr Raum eingestanden werden und die Anzahl der Zeitschritte bis zur minimalen Decay Rate erhöht werden. Andernfalls muss die Konstruktionsaufgabe angepasst werden, indem beispielsweise der Lösungsraum vergrößert werden muss. Die zweite Möglichkeit ist wahrscheinlich, wenn passende Merkmalskombinationen gefunden werden, aber sich der Agent nicht zielstrebig auf diese zu bewegt. Dies können Produktentwickelnde an einem hohen Regret und einem nicht konvergierende Loss erkennen. In diesem Fall kann die Discount Rate erhöht werden, um weiter vom Ausgangspunkt – der ursprünglichen Merkmalskombination – entfernte Bereiche für den Agenten interessanter zu machen. Außerdem ist dieser Effekt geringer je mehr verschiedene Startpunkt genutzt werden. Die dritte Möglichkeit ist für die Produktentwickelden mit Hilfe des Losses erkennbar. Dieser konvergiert im Gegensatz zur zweiten Möglichkeit. Dies ist ein Indiz dafür, dass der Agent, eine passende Merkmalskombination gefunden hat und auch deren Wert erkannt hat, jedoch noch nicht ausreichend Zeit in Form von Zeitschritten hatte, um das DQN entsprechend anzupassen. In diesem Fall kann die Anzahl der Zeitschritte erhöht werden oder die Lernrate erhöht. Bei der Erhöhung der Lernrate muss äußerst vorsichtig vorgegangen werden, da bei einer zu hohen Lernrate negative Effekte auf das Training entstehen können. Es können mehrere Möglichkeiten gleichzeitig auftreten. Entscheidend dabei ist, das für die Abschätzung der Merkmalskombinationen der ersten Möglichkeit das Expertenwissen der Produktentwickelnden benötigt wird.

7 Anwendungsbeispiele für automatisierte Anpassungskonstruktionen

Nachdem das Konzept und dessen Umsetzung in den vorangegangenen Kapiteln vorgestellt wurde, wird in diesem Kapitel die Modellierung der Anpassungskonstruktion (Abschnitt 7.1), die Machbarkeit des Selbstverstärkenden Lernens für die Automatisierung einer Konstruktionsaufgabe (Abschnitt 7.2), der Einfluss der Hyperparameter auf das Lernverhalten und den Trainingsaufwand (Abschnitt 7.3), sowie die Übertragbarkeit eines trainierten Lerners auf eine nachfolgende Produktgeneration an drei Demonstratorbauteilen evaluiert (Abschnitt 7.4). Diese stammen aus dem Kontext der sogenannten *Mass Customization*.

Der Kontext der Mass Customization

Bei der *Mass Customization* werden individuelle Anforderungen der Auftragstellenden direkt bei der Anpassung des Produktes berücksichtigt. Zeitgleich wird angestrebt, diese Anpassung mit dem gleichen Durchsatz einer Massenproduktion zu realisieren [224]. Aus wirtschaftlichen Gründen wird die Eindringtiefe, mit der die Anforderungen im Produktentwicklungsprozess berücksichtigt werden, möglichst gering gehalten, so dass der Neuheitsgrad dieser Produktentwicklung eine Anpassungskonstruktion nicht übersteigt. Demnach sind die Einstellungsmöglichkeiten der Anforderungen limitiert und die einzelnen Schritte zur Anpassung des Produktes gut dokumentiert. Außerdem werden hoch-automatisierte Produktions- aber auch Entwicklungsprozesse benötigt.

Exemplarisch werden drei individualisierte Komponenten für den leistungsorientierten Radsport für die Evaluation herangezogen. Sie sollen einerseits an die individuellen körperlichen Gegebenheiten (anthropometrische Anforderungen) der Auftragstellenden angepasst. Beispielsweise sollte die Länge des Kurbelarms einer Fahrradkurbel an die Länge des Oberschenkels, der Wade und des Fußes sowie an die individuelle Beweglichkeit (maximale Gelenkwinkel im Knie- und Fußgelenk) angepasst werden [225]. Andererseits sollen die Merkmale so optimiert sein, dass die Produkte sicher im Radsport eingesetzt werden können und dennoch so leicht wie möglich sind. Hieraus ergeben sich Anforderungen an die zulässigen Spannungen und Verformungen sowie an das Gewicht des Produktes. Der Detailgrad der Demonstratorbauteile und der Anforderungen wurde so reduziert, dass die Auswirkungen der Aktionen des Selbstverstärkenden Agenten auf die Merkmale nachvollzogen werden können und das Vorgehen gleichzeitig der Individualisierung realer Bau-

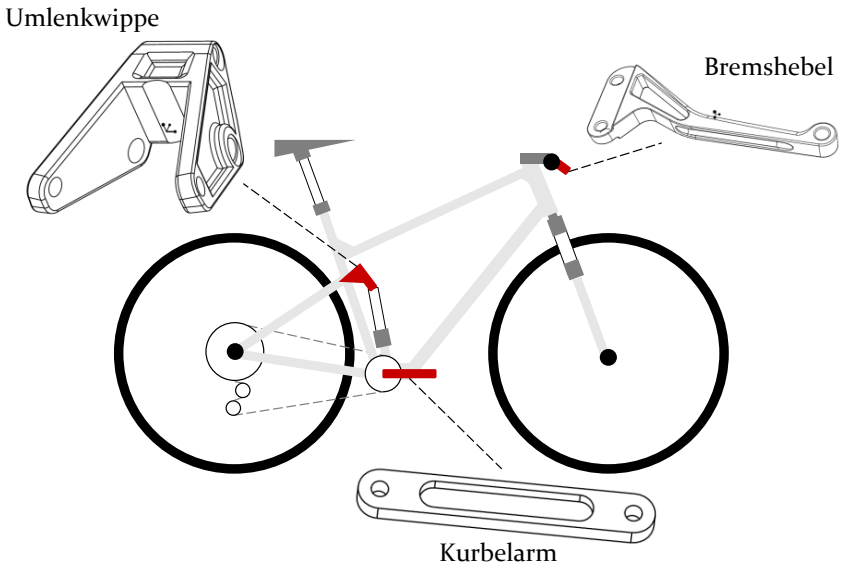


Bild 61: Darstellung eines MTB und Kennzeichnung der zu individualisierenden Komponenten

7.1 Das Modell des Anpassungsprozesses für individualisierte Produkte

Zunächst wird der Anpassungsprozess so modelliert, dass er durch explizit beschriebene Konstruktionsaufgaben dargestellt werden kann. Eine so beschriebene Konstruktionsaufgabe kann die Fragen beantworten: Welche Merkmale werden angepasst? Warum werden diese Merkmale angepasst? Wann werden die Merkmale angepasst? Der Anpassungsprozess besteht für alle drei Demonstratoren zunächst aus der Anpassung an die anthropometrischen Anforderungen (bspw. die Länge des Kurbelarms oder die Länge des Bremshebels). Anschließend werden Merkmale, die die anthropometrischen Anforderungen nicht beeinflussen, so optimiert, dass die Masse minimal bei gleichzeitiger Erfüllung der sicherheitsrelevanten Normen ist (bspw. DIN EN ISO 4210-2 [226] oder DIN EN ISO 4210-4 [227]). Die Prüfverfahren der Normen geben die Anforderungen durch Prüfverfahren vor. So wird beispielsweise der Kurbelarm mit 1500 N statisch belastet, wobei die auftretenden

Spannungen kleiner als die durch das Material vorgegeben zulässigen Spannungen sein müssen. Der Anpassungsprozess ist auf die Merkmale, deren

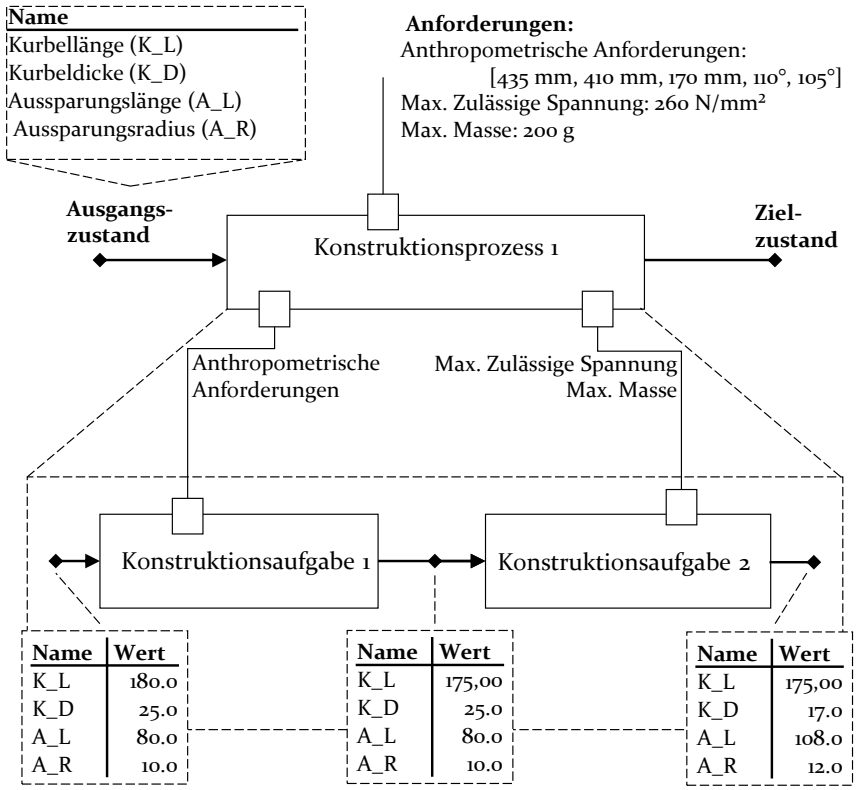


Bild 62: Modell der Anpassungskonstruktion für individualisierte Fahrradkomponenten

Bild 62 stellt das Modell des Anpassungsprozesses zur Individualisierung eines Kurbelarms dar. Das höchste Level beginnt am unerwünschten Ausgangszustand einer Standard Kurbel und endet am gewünschten Zielzustand einer individualisierten Kurbel. Jeder Zustand repräsentiert alle Merkmale und deren Ausprägungen. Aus Übersichtsgründen werden die vier Merkmale (Kurbellänge, Kurbeldicke, Aussparungslänge und Aussparungsradius), welche einen relevanten Beitrag zu individuellen Eigenschaften leisten, dargestellt. Diese resultierenden Eigenschaften sollen die beschriebenen Anforderungen (anthropometrische Anforderungen, maximale zulässige Spannung und maximale Masse) erfüllen. Zur Modellierung werden die Anforderungen über die Zweckcontainer des Konstruktionsprozesses auf die Konstruktions-

aufgaben aufgeteilt, da diese nicht durch eine einzige Konstruktionsaufgabe mit den Merkmalen verknüpft werden können. Es wird ein untergeordnetes Level modelliert. Dort wird in der chronologischen Abfolge zunächst die Kurbelarmlänge angepasst, um den individuellen Anforderungen zu entsprechen (s. Konstruktionsaufgabe 1 in Bild 62). Anschließend wird die Aussparung so angepasst, dass die zulässigen Spannungen und die Vorgaben an das Gewicht erfüllt werden (s. Konstruktionsaufgabe 2 in Bild 62). Zum besseren Verständnis sind die Änderungen der Merkmalsausprägungen über die Zustände ebenfalls dargestellt. Da auf dem untergeordneten Level alle Anforderungen über Konstruktionsaufgaben verknüpft wurden ist das Modell vollständig. Hier sei nochmals angemerkt, dass ein vollständiges Modell nicht garantiert, dass ein gewünschter Zustand gefunden werden kann. Es ist lediglich die Aussage, dass mindestens ein Automatisierungswerkzeug ein Merkmal anpassen kann, dessen Änderungen über die resultierenden Eigenschaften im Bezug zu den gewünschten Anforderungen steht. Restriktionen die den Lösungsraum einschränken werden hier nicht berücksichtigt.

Mit Hilfe der GUI M das Modell im CAD-System annotiert werden. Über die API wird es automatisch in der Wissensrepräsentation instanziiert. Die Darstellung der gesamten Wissensrepräsentation ist zu unübersichtlich. Daher wird in Bild 63 exemplarisch die zweite Konstruktionsaufgabe zur Anpassung der Aussparungsgeometrie exemplarisch dargestellt. Die Konzepte sind bewusst in Englisch gehalten und die Namenskonvention in der Camel Case-Notation, da die Wissensrepräsentation über ihre URI international einsetzbar sein soll.

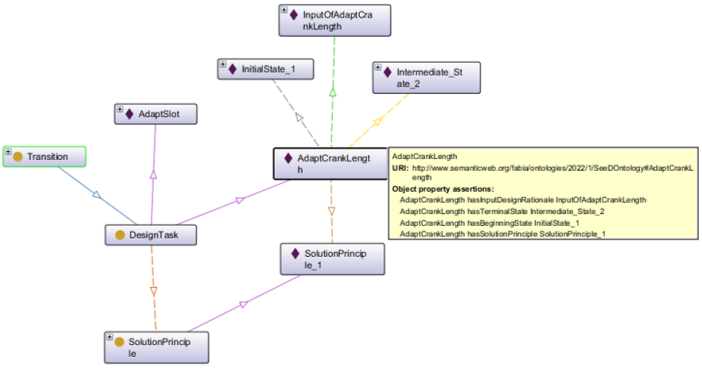


Bild 63: Darstellung eines Ausschnittes der Wissensrepräsentation für individualisierte Fahrradkomponenten

Bild 63 zeigt, dass die Konstruktionsaufgabe mit den Zuständen *InitialState_1* und *IntermediateState_2* über die Eigenschaft *hasBeginningState* und *hasTerminalState* verknüpft ist. Diese Verknüpfung beantwortet einerseits die Frage: „Wann werden die Merkmale angepasst?“. Andererseits werden über die Zustande die Merkmale repräsentiert, sodass die Frage: „Welche Merkmale werden angepasst?“ ebenfalls über die Abfrage dieser Eigenschaft beantwortet werden kann. Über die Eigenschaft *hasInputRationale* wird der Zweckcontainer verknüpft, der wiederum über *hasRequirement* über die Anforderungen *MaxWeight* und *MaxStress* definiert wird und die Frage: „Warum werden diese Merkmale angepasst?“ beantwortet. Im Folgenden soll diese Konstruktionsaufgabe durch selbstverstärkendes Lernen automatisiert werden.

7.2 Machbarkeitsstudie

Anhand der zweiten Konstruktionsaufgabe des zuvor beschriebenen Individualisierungsprozesses soll das Selbstverstärkende Lernen für deren Automatisierung evaluiert werden. Da bisher keine Ansätze für das Selbstverstärkende Lernen zur Optimierung von parametrischen Produktmodellen existieren, muss zunächst gezeigt werden, dass das vorgestellte Konzept in der Lage ist, die gewünschten Zielzustände zu finden und welcher Aufwand hierfür notwendig ist. Hierzu wird folgender Versuch durchgeführt. Die vorliegenden Konstruktionsaufgaben werden als Lernumgebung an die Selbstverstärkenden Lerner übergeben. Die notwendigen Informationen liegen bereits in den jeweiligen Modellen vor und werden zu Beginn jedes Anwendungsfalls noch einmal kurz vorgestellt. Für jedes Demonstratorbauteil werden die gleichen Hyperparameter über die GUI H eingestellt. Diese sind in Tabelle 9 zusammengefasst. Wie Tabelle 9 zu entnehmen ist, werden pro Agent 30.000 Zeitschritte für das Training durchgeführt. Diese Zahl ist absichtlich sehr hoch gewählt, um genügend Erfahrungen mit dem Lösungsraum zu generieren. Außerdem wird durch die verhältnismäßig großen Wert für $N_{DecayLastStep}$, an dem ϵ den kleinstmöglichen Wert erreicht ($\epsilon_{Final} = 0{,}01$) die Agenten zu viel Exploration gezwungen. Mit diesen Hyperparametern werden für jeden Anwendungsfall 5 Ausgangszustände definiert (I0 - I4). Die unterschiedlichen Ausgangszustände sollen verhindern, dass der Agent bereits nahe am gewünschten Zielzustand startet und die Ergebnisse durch diese kurzen Wegen verzerrt werden. Außerdem werden für jeden dieser Ausgangszustände 10 Agenten trainiert. Diese Agenten werden über 10 *Random Seeds* definiert. Der *Random Seed* legt die initialen Gewichte der DQNs sowie die zufälligen Aktionen der Exploration fest. Durch das Determinieren des Random Seeds sind die Experimente vollständig reproduzierbar. Um zu überprüfen, ob die Agenten tatsächlich die besten Zustände finden, wird anschließend der komplette Lösungsraum berechnet. Dies ist möglich, da es

sich um diskrete Zustände handelt. Dieser Schritt dient der Evaluation der Ergebnisse und ist nicht Teil der industriellen Praxis.

Tabelle 9: Übersicht der Hyperparameter des Benchmarks:

Name	Abkürzung	Grundeinstellung
Zeitschritte Pro Episode	N_T	50
Anzahl der Episoden	N_E	600
Start des Trainings	N_{Start}	500
Synchronisationsrate	N_{Start}	500
Startwert der Decay Rate	ϵ_{Start}	1,0
Endwert der Decay Rate	ϵ_{Final}	0,01
Letzter Zeitschritt der Decay Rate	$N_{decayfinal}$	16000
Anzahl der Versteckten Schichten	N_S	2
Anzahl der Knoten pro Versteckter Schicht	N_K	32
Batchgröße	N_{Batch}	32
Lernrate	α	0,0005
Discount Faktor	γ	0,8

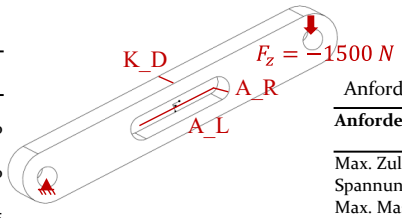
7.2.1 Die Ergebnisse des Kurbelarms

Die Anpassung des Kurbelarms berücksichtigt drei Merkmale. Die Dicke des Kurbelarms, die Länge und den Radius der Aussparung. Die Tiefe der Aussparung ergibt sich aus der Dicke der Kurbel. Die Länge der Kurbel selbst wurde durch die vorherige Konstruktionsaufgabe, die die anthropometrische Anforderungen berücksichtigt bereits festlegt. Die Anforderungen und Merkmale sowie der aus der DIN EN ISO 4210-2 vorgeschriebene Lastfall zur Absicherung der Kurbel sind in Bild 64 dargestellt [226].

Die Ergebnisse repräsentieren die Indikatoren, welche in Bild 65 - Bild 69 dargestellt sind. In Bild 65 sind für die fünf Ausgangszustände die individuellen Belohnungen (*Mass Reward* und *Stress Reward*) sowie die übergeordnete Zielgröße (*Total Reward*) über die Zeitschritte dargestellt. Dabei drückt der Reward Wert 0,0 aus, dass eine individuelle Anforderung, beispielsweise die maximale zulässige Spannung, eingehalten wird. Reward Werte, die über 0,0 liegen, erfüllen individuelle Anforderungen. Liegen sowohl die individuellen

Parameter Grid:

Merkmal	Untere Grenze	Obere Grenze	Schrittweite
Kurbeldicke (K_D) [mm]	16,0	24,0	1,0
Aussparungslänge (A_L) [mm]	50,0	130,0	5,0
Aussparungsradius (A_R) [mm]	4,0	12,5	0,5



Anforderungen:

Anforderung	Typ	Werte(e)
Max. Zul. Spannung [MPa]	Max	180
Max. Masse [mm]	Mac	200

Bild 64: Darstellung der Übergabe einer Konstruktionsaufgabe durch deren Anforderungen und Merkmale an die Lernumgebung

Reward Werte als auch deren gewichtete Summe - der *Total Reward* - über 0,0, dann erfüllt der zugrundeliegende Zustand alle Anforderungen zeitgleich.

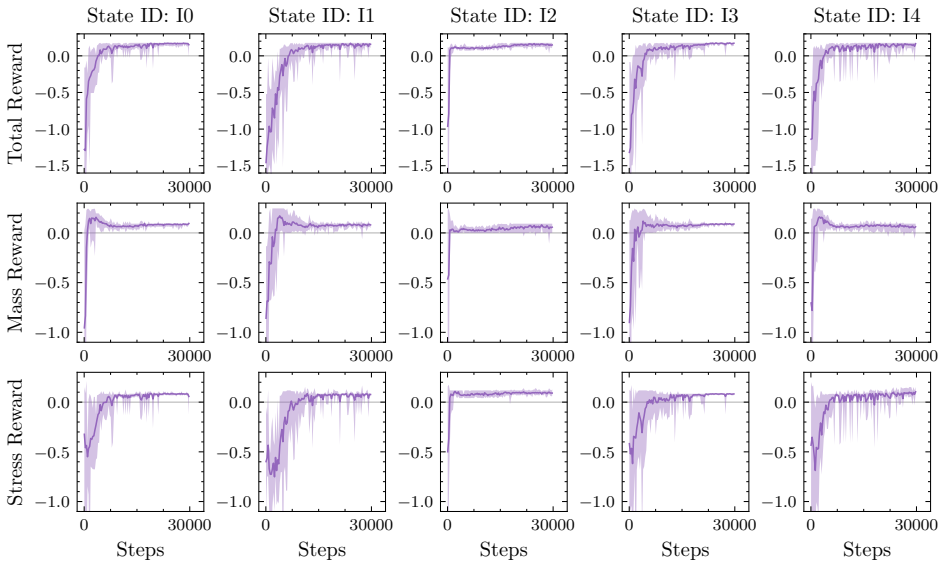


Bild 65: Darstellung der erreichten Belohnungen für die einzelnen Anforderungen und deren gleich gewichtete Summe als übergeordnete Zielfunktion für fünf Ausgangszustände (I0 – I4) gemittelt über 10 Random Seeds für den Kurbelarm

In Bild 65 ist zu erkennen, dass die Agenten unabhängig vom Ausgangszustand die gewünschten Zustände erreicht. Das Konvergenzverhalten ist für alle bis auf den Ausgangszustand mit der ID I2 sehr ähnlich. Bei I2 ist das Konvergenz-Plateau deutlich schneller erreicht. Um die erhaltenen Belohnungen einschätzen zu können, wurde der komplette Lösungsraum berechnet. Da es sich um zwei Zielfunktionen handelt, lassen sich die Zustände als Pareto-Font darstellen (s. Bild 66). Um die gewünschten zu quantifizieren sind die fünf Merkmalskombinationen mit der höchsten Belohnung sind

in Tabelle 10 zusammengefasst. Weitere Merkmalskombinationen können in Tabelle 14 des Anhangs nachvollzogen werden. Aus dem Abgleich zwischen Bild 65, Bild 66 und Tabelle 10 kann gefolgert werden, dass der Agent unabhängig vom Ausgangszustand die gewünschten Zustände findet.

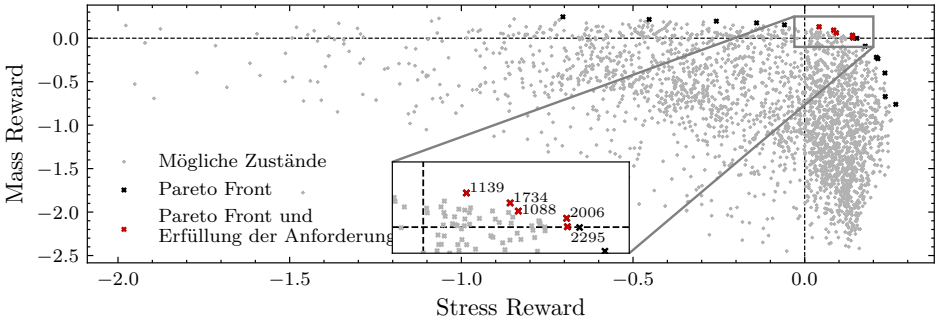


Bild 66: Darstellung aller Möglichen Zustände (Merkmalskombinationen) unter Kennzeichnung der Pareto Front und der Anforderungen für die individuellen Zielfunktionen

Tabelle 10: Die fünf Merkmalskombinationen mit den höchsten erreichten Belohnungen für den Kurbelarm:

ID	Kurbelarm-dicke	Aussparungs-länge	Aussparungs-radius	Masse [kg]	Spannung [kN mm^{-2}]	Reward Mass	Reward Stress	Total Reward
1734	21.0	130.0	12.0	0.181140	0.162231	0.092753	0.084361	0.177114
2006	22.0	125.0	12.0	0.193096	0.150676	0.033955	0.139221	0.173176
1139	19.0	125.0	12.0	0.173358	0.171174	0.131027	0.041903	0.172930
1088	19.0	110.0	12.0	0.187470	0.160556	0.061625	0.092313	0.153938
1360	20.0	105.0	12.0	0.200092	0.148028	0.001355	0.151792	0.150437

Um das Anpassungsverhalten der tiefen Neuronalen Netze zu verstehen, ist in Bild 67 der Loss für die fünf Ausgangszustände logarithmisch dargestellt. Dort ist zu erkennen, dass dieser zunächst stark fällt und anschließend stagniert. Folglich wird im Bereich des starken Abfalls das Netz stark angepasst. Dies spricht dafür, dass der Agent Erfahrungen sammelt, die starke Veränderungen für die Strategie nach sich ziehen. Im Bereich der Stagnation wird das Netz nur noch marginal angepasst. Analog werden hier keine Änderungen an der Strategie vorgenommen, der Agent ist sich dieser sicher.

Ein weitere Indikator ist der Regret. Dieser ist für die 5 Ausgangszustände in Bild 68 dargestellt. Dabei ist anzumerken, dass der Regret immer steigt, solange der Ausgangszustand nicht der Zielzustand ist. Je flacher der Regret ist, desto näher ist der Ausgangszustand am Zielzustand. In Bild 68 ist zu erkennen, dass Ausgangszustand I2 einen sehr flachen Verlauf hat. Die Verluste, die auf dem Weg zum Zielzustand gemacht werden sind demnach kleiner als bei den anderen Ausgangszuständen. Da in jedem Fall ein gewünschter

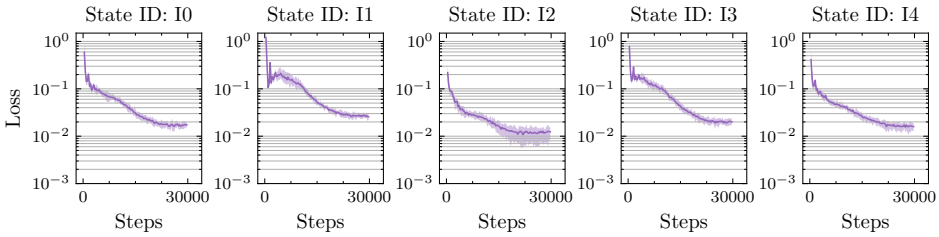


Bild 67: Darstellung des Losses für fünf Ausgangszustände (I0 – I4) gemittelt über 10 Random Seeds für den Kurbelarm

Zielzustand erreicht wird, lässt sich folger, dass I2 deutlich näher an den gewünschten Zustanden im Lösungsraum liegen muss.

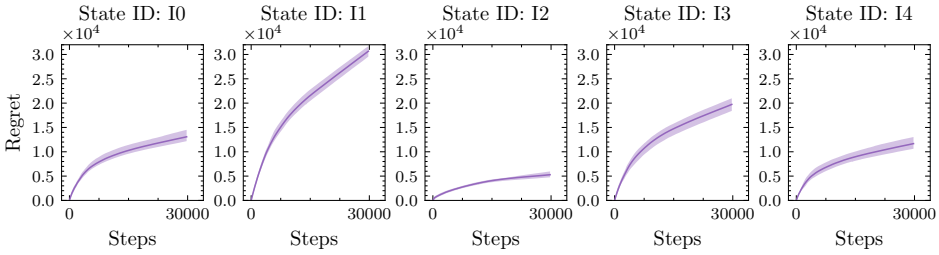


Bild 68: Darstellung des Regrets für fünf Ausgangszustände (I0 – I4) gemittelt über 10 Random Seeds für den Kurbelarm

Bild 69 zeigt für alle 5 Ausgangszustände die Anzahl der notwendigen Zustände (*Visited States*) an. Die Zahlen sind absolut, so dass ein Agent von Zustand I0 circa 1500 Zustände benötigt, um die passende Strategie zu finden. Wiederum zeigt sich, dass I2 ein Sonderfall ist, der deutlich weniger Zustände benötigt, um einen passenden Zielzustand zu erreichen. Aus den Darstellungen Bild 65 - Bild 69 kann gefolgert werden, dass ein Selbstverstärkender Lernprozess in der Lage ist, unabhängig vom Ausgangszustand eine Merkmalskombination zu finden, die die Anforderungen erfüllt.

7.2.2 Die Ergebnisse der Umlenkwippe

Der in Unterabschnitt 7.2.1 vorgestellte Versuch wird für die Anpassung einer Umlenkwippe wiederholt. Diese berücksichtigt 5 Merkmale. Die Anforderungen und Merkmale sowie der aus der DIN EN ISO 4210-2 vorgeschriebene Lastfall zur Absicherung sind in Bild 70 dargestellt [226]. Die Umlenkwippe wurde ausgewählt, weil sie im Kontext der Individualisierung von Radkomponenten im Gegensatz zum Kurbelarm ein symmetrisches, voluminöses Bauteil darstellt. Der Lösungsraum ist durch die 5 einstellbaren Merkmale

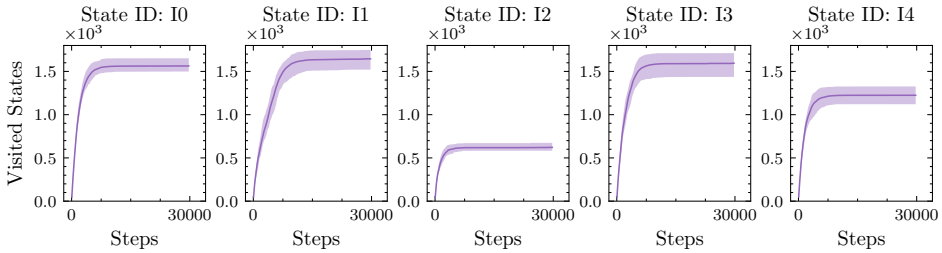
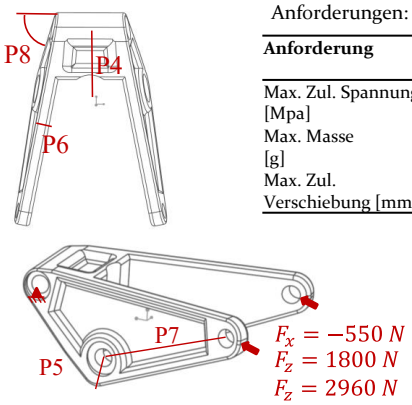


Bild 69: Darstellung der benötigten Zustände für fünf Ausgangszustände (I0 – I4) gemittelt über 10 *Random Seeds* für den Kurbelarm

Parameter Grid:

Merkmal	Untere Grenze	Obere Grenze	Schrittweite
P4 [mm]	3.0	6.0	1.0
P5 [mm]	10.0	13.0	1.0
P6 [mm]	6.0	10.0	1.0
P7 [mm]	55.0	71.0	1.0
P8 [°]	79.0	85.0	1.0



Anforderungen:

Anforderung	Typ	Werte(e)
Max. Zul. Spannung [Mpa]	Max	165
Max. Masse [g]	Max	110
Max. Zul. Verschiebung [mm]	Max	0,5

Bild 70: Darstellung der Übergabe einer Konstruktionsaufgabe durch deren Anforderungen und Merkmale an die Lernumgebung

In Bild 71 sind für die fünf Ausgangszustände die individuellen Belohnungen (*Mass Reward*, *Stress Reward* und *Stress Deformation*) sowie die übergeordnete Zielgröße (*Total Reward*) über die Zeitschritte dargestellt Analog zum Kurbelarm ist ein gewünschter Zustand erreicht, wenn sowohl die individuellen Reward Werte als auch deren gewichtete Summe - der *Total Reward* - über 0,0 liegen. Anschließend wurde zur Evaluation der gesamte Lösungsraum berechnet. Die fünf Merkmalskombinationen mit den höchsten Belohnungen sind in Tabelle 11. Weitere Merkmalskombinationen können Tabelle 15 des Anhangs entnommen werden. Bild 71 ist zu entnehmen, dass die gewünschten Zustände unabhängig vom Ausgangszustand gefunden werden, da wiederum alle Agenten eine Merkmalskombination finden, deren Belohnungen für alle Rewards größer als 0,0 ist. Beim Vergleich mit den bestmöglichen Merkmalskombinationen aus Tabelle 11 ist zu erkennen, dass im Fall der Umlenkwippe

die höchsten Belohnungen des gesamten Lösungsraums knapp über 0,0 liegen. Dementsprechend ist durch das Konvergieren der Agenten auf einem Bereich knapp über 0,0 gewährleistet, dass diese Merkmalskombinationen durch die Agenten gefunden werden.

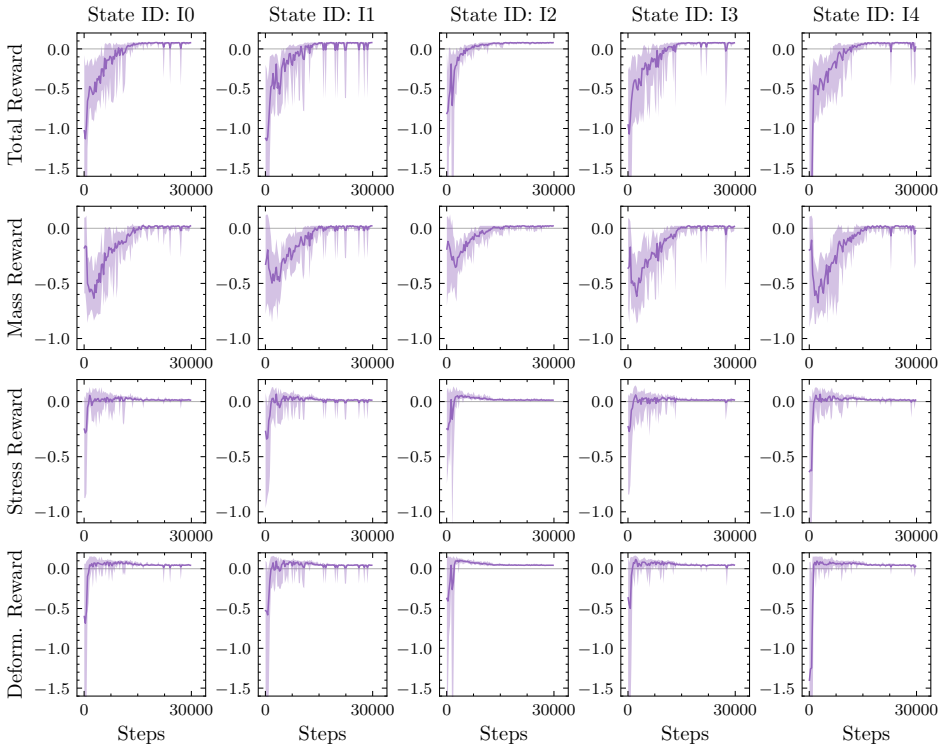


Bild 71: Darstellung der erreichten Belohnungen für die einzelnen Anforderungen und deren gleich gewichtete Summe als übergeordnete Zielfunktion für fünf Ausgangszustände (I0 – I4) gemittelt über 10 Random Seeds für die Umlenkwippe

In Bild 72 sind die Verläufe für den Loss, den Regret sowie die benötigten Zustände für die 5 Ausgangszustände der Umlenkwippe zusammengefasst. Qualitativ gleichen sich die Verläufe mit denen des Kurbelarms (s. Bild 67 - Bild 69). Die dort getroffenen Aussagen gelten folglich auch für die Umlenkwippe. Wie auch in Bild 67 fällt stark der Loss und stagniert gegen Ende des Trainings auf einem sehr niedrigen Niveau. Beachtenswert ist, dass hier alle fünf Verläufe im ersten Drittel ein Plateau haben, welches darauf hindeuten kann, dass der Agent dort bereits eine Strategie erlernt hat, welche er im späteren Verlauf zu Gunsten einer anderen, mutmaßlich bessern getauscht hat. Quantitativ werden in diesem deutlich größeren Lösungsraum auch deutlich mehr Zustände zum Erlernen der Strategie benötigt.

Tabelle 11: Die fünf Merkmalskombinationen mit den höchsten erreichten Belohnungen für die Umlenkwippe:

ID	P4	P5	P6	P7	P8	stress [MPa]	deformation [mm]	mass [kg]	reward mass	reward stress	reward deformation	overall reward
2029	6	13	8	55	85	158.397	0.425875	0.106350	0.022226	0.013404	0.044353	0.079984
8845	6	12	8	55	85	159.743	0.431103	0.106329	0.022356	0.010672	0.041225	0.074254
8380	6	11	8	55	85	160.667	0.436433	0.106307	0.022487	0.008795	0.038036	0.069317
7922	6	10	8	55	85	162.840	0.441766	0.106286	0.022617	0.004384	0.034845	0.061845
9282	6	13	8	56	85	160.754	0.447804	0.106718	0.019985	0.008619	0.031232	0.059836

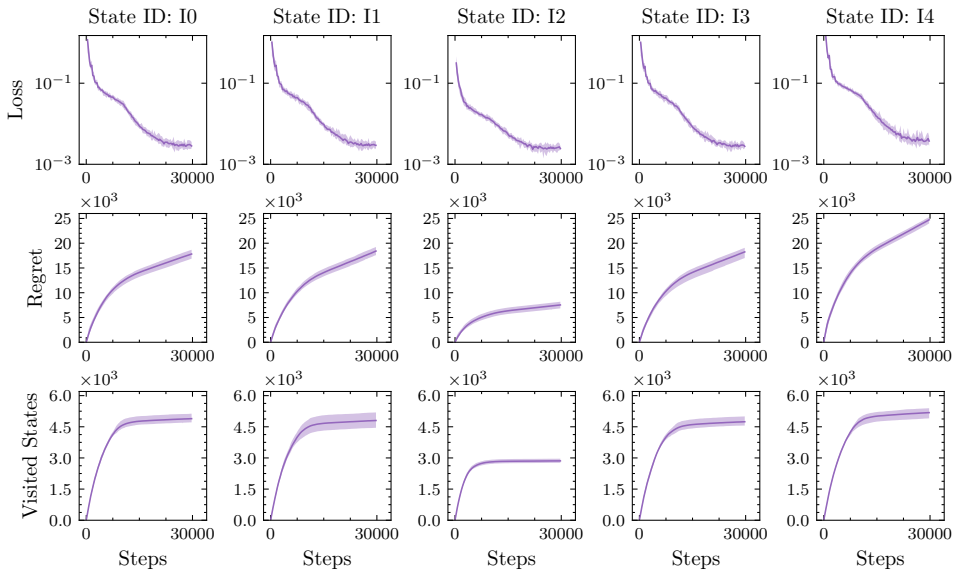


Bild 72: Darstellung des Losses, des Regrets und der benötigten Zustände für fünf Ausgangszustände (I0 – I4) gemittelt über 10 *Random Seeds* für die Umlenkwippe

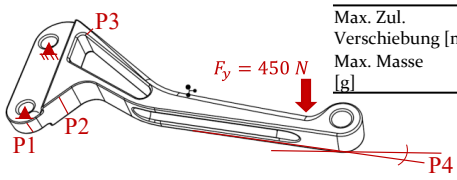
7.2.3 Die Ergebnisse des Bremshebels

Die Anpassung der Bremshebel berücksichtigt 3 Merkmale. Die Anforderungen und Merkmale sowie der aus der DIN EN ISO 4210-4 vorgeschriebene Lastfall zur Absicherung sind in Bild 73 dargestellt [227]. Der Bremshebel wurde ausgewählt, weil bei der Minimierung von P2 und P3, die augenscheinlich zur Reduzierung der Masse notwendig sind, sehr hohe Spannungen unter- und oberhalb der Aussparung auftreten können, welche zu sehr starken Verzerrungen des FEM-Netzes führen können. Der Spannungsverlauf wird den Agenten jedoch vorenthalten und nicht als Reward zurückgeführt. Ziel ist es, zu erproben, ob die DQNs der Agenten ausreichen, um die Unstetigkeit

im Verlauf der Netzverschiebung zu lernen. Dazu werden die individuel-

Parameter Grid:

Merkmal	Untere Grenze	Obere Grenze	Schrittweite
P1	10	18	1,0
P2	2,0	5,0	1,0
P3	3,0	9,0	1,0
P4	7,0	11,0	1,0



Anforderungen:

Anforderung	Typ	Werte(e)
Max. Zul. Verschiebung [mm]	Max	0,8
Max. Masse [g]	Max	300

Bild 73: Darstellung der Übergabe einer Konstruktionsaufgabe durch deren Anforderungen und Merkmale an die Lernumgebung

In Bild 74 sind für die fünf Ausgangszustände die individuellen Belohnungen (*Mass Reward* und *Stress Deformation*) sowie die übergeordnete Zielgröße (*Total Reward*) über die Zeitschritte dargestellt. Analog zum Kurbelarm ist ein gewünschter Zustand erreicht, wenn sowohl die individuellen Reward Werte als auch deren gewichtete Summe - der *Total Reward* - über 0,0. Zur Evaluation wurde der gesamt Lösungsraum berechnet. Die fünf Merkmalskombinationen mit den höchsten Belohnungen sind in Tabelle 11 dargestellt. Weitere Merkmalskombinationen können Tabelle 16 des Anhangs entnommen werden. Bild 74 ist zu entnehmen, dass die gewünschten Zustände unabhängig vom Ausgangszustand gefunden werden.

Tabelle 12: Die fünf Merkmalskombinationen mit den höchsten erreichten Belohnungen für den Bremshebel:

ID	P2	P3	P4	P5	Verschiebung [mm]	Masse [g]	Reward Mass	Reward Stress	Total Reward
1642	12	3.0	8.0	10	0.852471	0.307233	0.006074	0.007129	0.013203
1659	12	3.0	8.5	10	0.859892	0.305897	0.000087	0.010572	0.010658
1770	12	3.5	8.0	10	0.863655	0.303169	-0.008847	0.017601	0.008754
1521	12	2.5	8.0	10	0.843818	0.311122	0.013054	-0.008671	0.004383
1531	12	2.5	8.5	10	0.856143	0.310023	0.003111	-0.000174	0.002937

In Bild 75 sind die Verläufe für den Loss, den Regret sowie die benötigten Zustände für die 5 Ausgangszustände den Bremshebel zusammengefasst. Qualitativ gleichen sich die Verläufe mit denen des Kurbelarms (s. Bild 67 - Bild 69). Die dort getroffenen Aussagen gelten folglich auch für den Bremshebel.

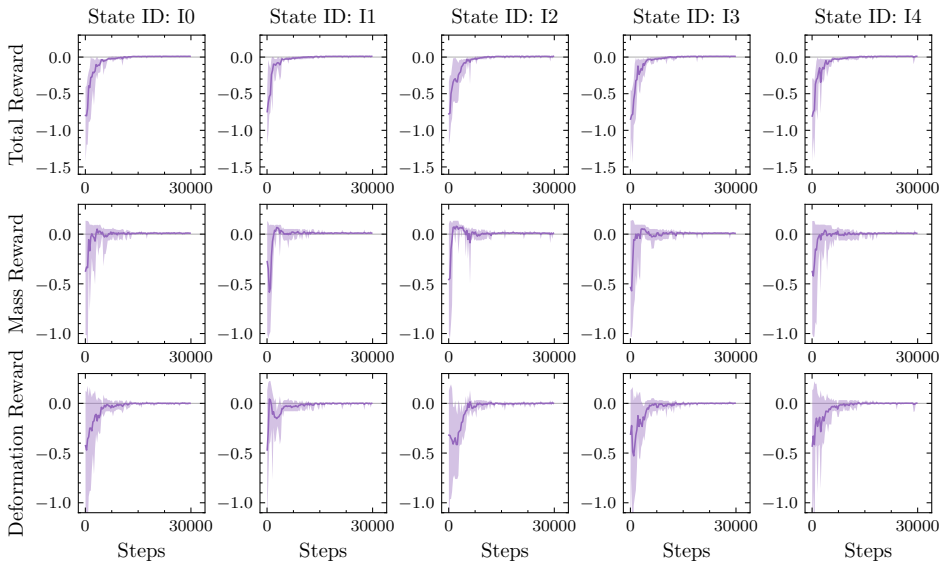


Bild 74: Darstellung der erreichten Belohnungen für die einzelnen Anforderungen und deren gleich gewichtete Summe als übergeordnete Zielfunktion für fünf Ausgangszustände (I0 – I4) gemittelt über 10 *Random Seeds* für den Bremshebel

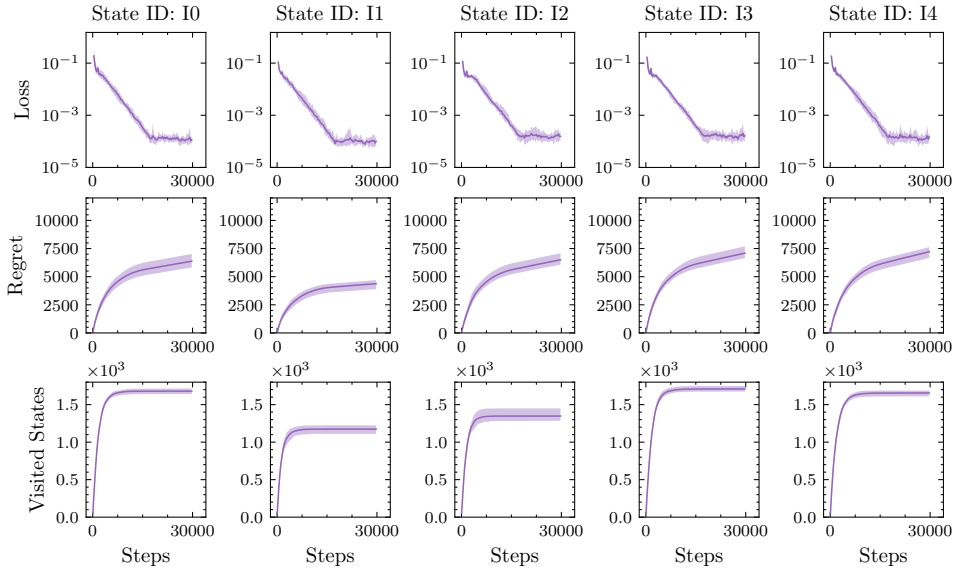


Bild 75: Darstellung des Losses, des Regrets und der benötigten Zustände für fünf Ausgangszustände (I0 – I4) gemittelt über 10 Random Seeds für den Bremshebel

7.2.4 Zwischenfazit zur Machbarkeitsstudie

In den vorangegangen Abschnitten wurde anhand von drei unterschiedlichen Fahrradkomponenten gezeigt, dass Selbstverstärkendes Lernen eingesetzt werden kann, um in großen Lösungsräumen Merkmalskombinationen so anzupassen, dass diese die vorher definierten Anforderungen erfüllen. In den gezeigten Versuchen wurden jeweils die bestmöglichen Merkmalskombinationen gefunden. Dies konnte durch eine vollständige Berechnung des jeweiligen Lösungsraums evaluiert werden. Der Aufwand zum finden einer solchen Merkmalskombination und das Erlernen der Strategie zu dieser wurde in Visited States angegeben. Diese stellen die benötigten Simulationen dar und erlauben eine Hardware-unabhängige Abschätzung des Aufwandes der zum Vergleich mit anderen Werkzeugen zur Merkmalsanpassung (beispielsweise der Mehrzieloptimierung [P8]) verwendet werden kann. Bei der Betrachtung der Visited States hat sich gezeigt, dass diese signifikant niedriger sind, wenn die Ausgangskombination der Merkmal nahe an einer passenden Merkmalskombination liegt. Für das Erlernen einer allgemeinen Strategie zur Anpassung des Bauteils sollte dies berücksichtigt werden, beispielsweise durch mehrere Ausgangspositionen.

7.3 Hyperparameterstudie

Im vorangegangen Abschnitt wurde die Automatisierung durch Selbstverstärkendes Lernen gezeigt. Die dabei verwendeten Hyperparameter wurden durch Probieren und Erfahrung mit Selbstverstärkendem Lernen gewählt. Im Lauf dieser Arbeit wurde mehrfach betont, dass diese Hyperparameter entscheidend für den Erfolg und die Effizienz des Selbstverstärkenden Lernens sind. Dies wurde in [P9] untersucht und soll im Folgenden dargestellt werden. Anhand von zwei Demonstratorbauteilen (Kurbelarm und Umlenkwippe) werden die Effekte der Hyperparamtervariation untersucht. Die Gruppen wurden in Unterabschnitt 6.2.3 bereits vorgestellt. Sie definieren sich aus den Aufgaben der Hyperparameter für das Selbstverstärkende Lernen: 1.) Explorations-Strategie, 2.) Netzanpassung und 3.) Netzarchitektur. Die Hyperparameter der zeitlichen Abfolge wurden in [P9] untersucht. Dabei wurden keine Effekte festgestellt. Für alle Hyperparameterkombinationen wird der Regret und die Visited States relativ zu den in Abschnitt 7.2 verwendeten Hyperparmetern, welche als Benchmark dienen, ermittelt. Die zwei Demonstratoren werden direkt gegenübergestellt, um gemeinsame Tendenzen zu visualisieren.

Für die Gegenüberstellung wurden Heatmaps erarbeitet. Exemplarisch ist in Bild 76 die Heatmap für die notwendigen Zustände des Kurbelarms dargestellt. Jede Kachel repräsentiert den Mittelwert und die Grenzen des dazugehörigen

95% Konfidenzintervalls einer Hyperparameterkombination. Der Mittelwert ergibt sich aus den 10 Random Seeds die bei der Initiierung der Agenten verwendet wurden und den fünf Ausgangszuständen, um auszuschließen, dass ein günstiger Ausgangszustand vorliegt. Ein Mittelwert von 1,00 bedeutet, dass genauso viele Zustände zur Berechnung notwendig sind, wie bei der Verwendung der Hyperparameter des Benchmark. Analog bedeutet ein Mittelwert von 0,50, dass halb so viele Zustände benötigt wurden. Dies wird über die Farbskala dargestellt. Es gilt: je kleiner desto besser.

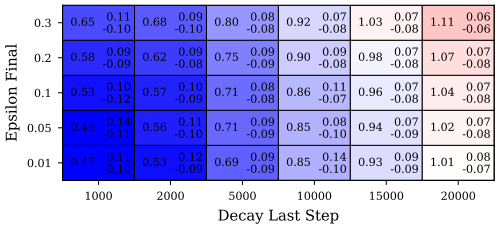


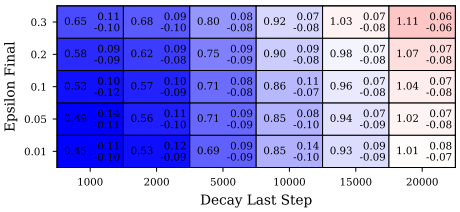
Bild 76: Darstellung einer exemplarischen Heatmap für die notwendigen Zustände für die Hyperparameter der Explorations-Strategie $\epsilon_{Final} \in \{0,3; 0,2; 0,1; 0,05; 0,01\}$ und $N_{DecayLastStep} \in \{1000; 2000; 5000; 10000; 15000; 20000\}$

Neben den notwendigen Zuständen wird der Regret auf die gleiche Weise untersucht. Der Regret gibt den Unterschied zwischen der ausgeführten Strategie und der bestmöglichen Strategie an. Daher gibt ein Mittelwert von 1,00 an, dass die untersuchte Hyperparameterkombination die gleichen Verluste wie die Strategie der Benchmarkparameter enthält. Verlust entsteht immer dann, wenn der Agent eine Entscheidung trifft, die nicht optimal ist. Ein Mittelwert von 2,00 bedeutet, dass die Strategie der untersuchten Hyperparameter doppelt so große Verluste hatte, wie die Strategie der betrachteten Hyperparameter. Es gilt: daher je kleiner desto besser.

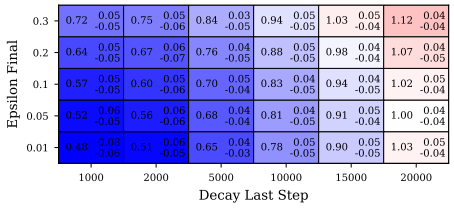
7.3.1 Effekte der Hyperparameter der Explorations-Strategie

Die Explorations-Strategie entscheidet darüber, wie das tiefe Neuronale Netz zur Vorhersage der nächsten Aktion genutzt wird. Wie in Bild 45 auf Seite 92 gezeigt, wird über die Hyperparameter ϵ_{Final} und $N_{DecayLastStep}$ der Gradient eines Schwellenwertes über die Zeitschritte bestimmt. Bild 77 und Bild 78 zeigen den Effekt dieser Hyperparameter auf die notwendigen Zustände und den entsprechenden Regret.

Bei der Betrachtung der vier Darstellung aus Bild 77 und Bild 77 fällt auf, dass der Einfluss der untersuchten Hyperparameter in diesen Fällen nicht vom untersuchten Bauteil abhängt, da die Verläufe annähernd identisch sind, obwohl Geometrie, als auch Größe des Lösungsraums und Anzahl

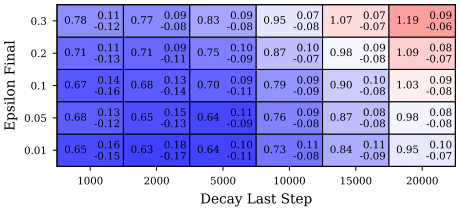


(a) Kurbelarm

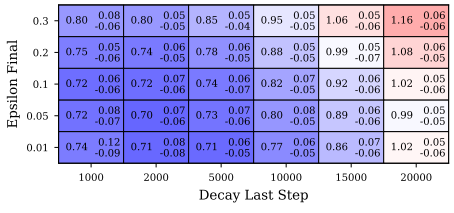


(b) Umlenkwippe

Bild 77: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter ϵ_{Final} und $N_{DecayLastStep}$ der Explorations-Strategie anhand der notwendigen Zustände



(a) Kurbelarm



(b) Umlenkwippe

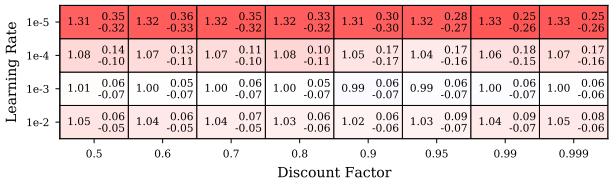
Bild 78: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter ϵ_{Final} und $N_{DecayLastStep}$ der Explorations-Strategie anhand des Regrets

der Zielfunktionen unterschiedlich sind. Darüber hinaus kann aus Bild 77 geschlossen werden, dass durch einen steileren Gradienten die Anzahl der notwendigen Zustände halbiert werden kann. Demnach explorieren die Agent der Benchmarkhyperparameter den Lösungsraum unnötig stark.

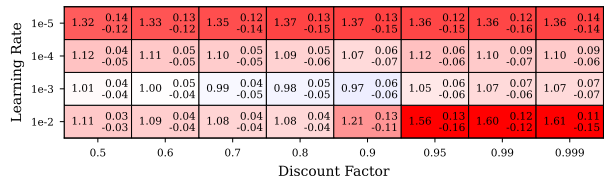
7.3.2 Effekte der Hyperparameter der Netzanpassung

Die Netzanpassung über die Bellman-Gleichung (Gleichung 9 auf Seite 53) wird über die Hyperparameter Discount Faktor γ und Lernrate α gesteuert. Diese sind in Bild 79 und Bild 80 gegenübergestellt. Der Discount Faktor bestimmt die Weitsichtigkeit der Agenten. Je höher γ ist, desto stärker werden weiter entfernt liegende Zustände bei der Berechnung des Losses für die Netzanpassung gewichtet. Die Lernrate bestimmt die Auswirkung des Losses auf die Anpassung des Netzes.

Bild 79 und Bild 80 zeigen, dass die Effekte für beide Demonstratorbauteile in Tendenzen übereinstimmen, jedoch nicht identisch sind. Beispielsweise haben große Discount Faktoren bei großen Lernraten einen viel negativeren Effekt für die Umlenkwippe, als für den Kurbelarm. Die zwei Darstellungen bezüglich der notwendigen Zustände aus Bild 79 zeigen, dass sowohl für zu große, als auch zu kleine Lernraten mehr Zustände benötigt werden. Dies



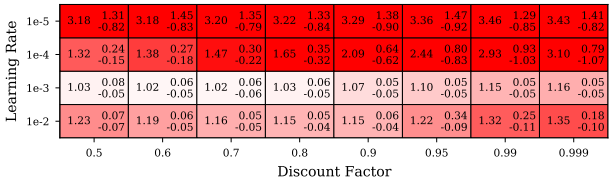
(a) Kurbelarm



(b) Umlenkwippe

Bild 79: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter Discount Faktor γ und Lernrate α der Netzanpassung anhand der notwendigen Zustände

zeigt sich ebenso für zu große oder zu kleine Discount Faktoren. Bezüglich des Regrets sind diese Effekte sogar ausgeprägter (s. Bild 80). Da kaum Werte unter 1,00 in beiden Tabellen vorliegen, lässt sich die Anzahl der notwendigen Zustände und der Regret im Vergleich zu den Benchmark Hyperparametern durch eine Anpassung der Lernrate und des Discount Faktors kaum reduzieren.



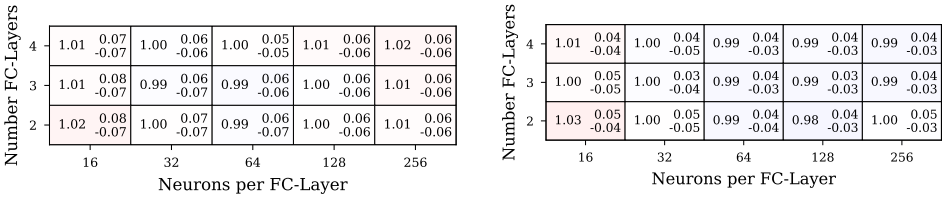
(a) Kurbelarm

(b) Umlenkwippe

Bild 80: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter Discount Faktor γ und Lernrate α der Netzanpassung anhand des Regrets

7.3.3 Effekte der Hyperparameter der Netzarchitektur

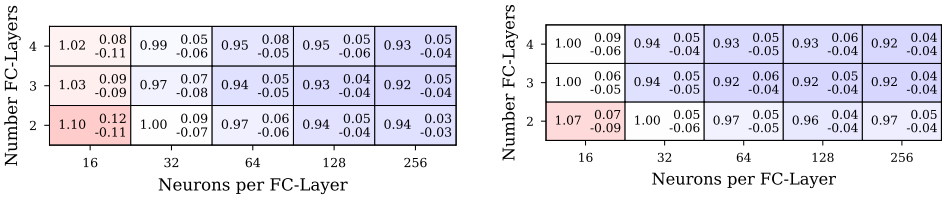
Die Netzarchitektur entscheidet über die Möglichkeiten, den Lösungsraum abzubilden. Hier werden die tiefen, neuronalen Netze verwendet, die ausschließlich vorwärts, d. h. in Richtung der Ausgänge, durchlaufen werden. Eine Rückkopplung ist nicht vorhanden. Somit reduziert sich die Netzarchitektur auf die Anzahl der versteckten Schichten N_S und die Anzahl der Knoten pro Schicht N_K. Deren Effekte auf die notwendigen Zustände und den Regret werden in Bild 81 und Bild 82 gegenübergestellt.



(a) Kurbelarm (b) Umlenkwippe

Bild 81: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter versteckte Schichten N_S und Anzahl der Knoten pro Schicht N_K der Netzarchitektur anhand der notwendigen Zustände

Wie im Falle der Hyperparameter für die Explorations-Strategie, sind die Effekte der Hyperparametervariation der Netzarchitektur für die beiden Demonstratorbauteile annähernd identisch. Die Anpassung des Netzes hat kaum einen Effekt auf die notwendigen Zustände in Bild 81. Anhand des Regrets (s. Bild 82) zeigt sich jedoch, dass Agenten von einem größeren Netz leicht profitieren würden.



(a) Kurbelarm (b) Umlenkwippe

Bild 82: Gegenüberstellung verschiedener Kombinationen für die Hyperparameter versteckte Schichten N_S und Anzahl der Knoten pro Schicht N_K der Netzarchitektur anhand des Regrets

7.3.4 Zwischenfazit zur Hyperparameterstudie

In den vorangegangen Abschnitten wurde anhand von zwei Fahrradkomponenten der Einfluss der Hyperparameter auf die verschiedenen Indikatoren

untersucht. In Summe wurden knapp 80 Versuche für jeder der zwei unterschiedlichen Komponenten durchgeführt. Dabei hat sich gezeigt, dass bei diesen die Hyperparameter der Explorations-Strategie und der Netzanpassung sehr großen Einfluss auf die Strategie und den Aufwand für das Erlernen der Strategie haben. Besonders auffällig ist, dass die Effekte in beiden Komponenten ähnlich auftreten. Die qualitativen Verläufe der Heatmaps sind annähernd identisch und die quantitativen Auswirkungen vergleichbar. Dies deutet darauf hin, dass eine passende Hyperparameterkombination für weitere Anwendungsfälle als Startpunkt dienen könnte.

7.4 Transferstudie

Ein wesentlicher Vorteil gegenüber anderer Optimierungsverfahren ist, dass bei maschinellen Lernverfahren das trainierte Modell persistent gespeichert und wiederverwendet werden kann. In Abschnitt 5.6 wurde das Konzept hierfür vorgestellt. Für die drei Demonstratorbauteile ist das Verfahren analog. Durch die Berücksichtigung der anthropometrischen Anforderungen in der ersten Konstruktionsaufgabe, findet die zweite Konstruktionsaufgabe unter ähnlichen aber nicht gleichen Randbedingungen statt. Im Folgenden soll gezeigt werden, ob diese ähnliche Konstruktionsaufgabe effizienter gelöst werden kann, wenn anstelle eines zufällig initiieren DQNs ein vortrainiertes Netz verwendet wird. Dazu wird das Netz des in Abschnitt 7.2 vorgestellten Kurbelarms mit der Länge 175,0 mm auf Kurbelarme der Länge 172,5 mm und 170,0 mm übertragen. In Bild 83 sind die individuellen Belohnungen (Stress Reward und Mass Reward) sowie deren gleich gewichtete Summe als übergeordnete Zielfunktion dargestellt, um zu überprüfen, ob die vortrainierten Netze ebenfalls die gewünschten Zielzustände erreichen. Hierzu wurden wiederum drei Selbstverstärkende Lerner mit je 10 Agenten trainiert. Die grüne Kurve zeigt die Belohnungen unter Verwendung der Benchmark Hyperparameter.

Die violette Kurve zeigt die Belohnung bei optimierten Hyperparametern nach den Erkenntnissen aus Abschnitt 7.3. Dementsprechend wurde ϵ_{Final} von 0,01 und $N_{DecayLastStep}$ von 1000 verwendet. Die orange Kurve zeigt die Belohnungen der Agenten mit vortrainiertem DQN und optimierten Hyperparametern. In Bild 83 ist zu erkennen, dass alle Agenten auf die gewünschten Zustände konvergieren. Augenscheinlich erreicht die orange Kurve das Plateau zuerst. Dies deutet darauf hin, dass diese Agenten die gewünschten Zustände schneller erreichen. Dies soll durch die Indikatoren weiter untersucht werden. Daher sind in Bild 84 und Bild 85 die benötigten Zustände zum Erreichen der Zielzustände und des Regrets zur Abschätzung der Verluste dargestellt.

In Bild 84 kann abgelesen werden, dass sich die notwendigen Zustände und damit die benötigten Simulationen im Fall des Kurbelarms um etwa 40% durch Optimierung der Hyperparameter übertragen lassen. Außerdem wird die Anzahl der notwendigen Zustände durch Übertragung des Netzes, welches an einem 175,0 mm langen Kurbelarms trainiert wurde, auf eine Kurbel mit der Länge 172,5 mm bzw. 170,0 mm noch einmal deutlich reduziert. Die in Bild 85 dargestellten Verläufe für den Regret zeigen für den vortrainierten Fall (orange) annähernd lineare Verläufe mit minimalen Konvergenzintervallen. Dieser lineare Anstieg entsteht durch einen Agenten, der sich vom Ausgangszustand zum Zielzustand bewegt. Solange der momentane Zustand nicht der Zielzustand ist, entsteht Verlust, der im Regret aufsummiert wird.

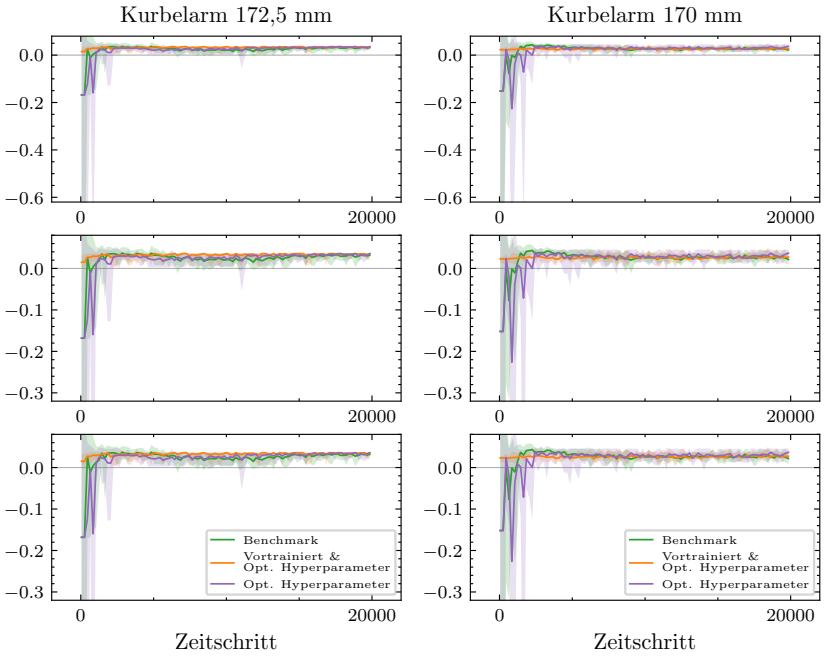


Bild 83: Darstellung der individuellen Belohnungen und deren gewichtete Summe für die Agenten mit Benchmark Hyperparametern (grün), optimierten Hyperparametern (violett) und optimierten Hyperparametern und vortrainierten DQN (orange)

Zwischenfazit zur Transferstudie

Im vorangegangen Abschnitt wurde der Kurbelarm genutzt, um die Übertragung einer gelernten Strategie auf nachfolgende Produktgenerationen zu untersuchen. Auf Basis einer Kurbel der Länge 175,0 mm wurden die Gewichte des DQNs, welches die Strategie des Agenten abbildet, trainiert und gespeichert. Diese wurden anschließend für die identische Konstruktionsaufgabe

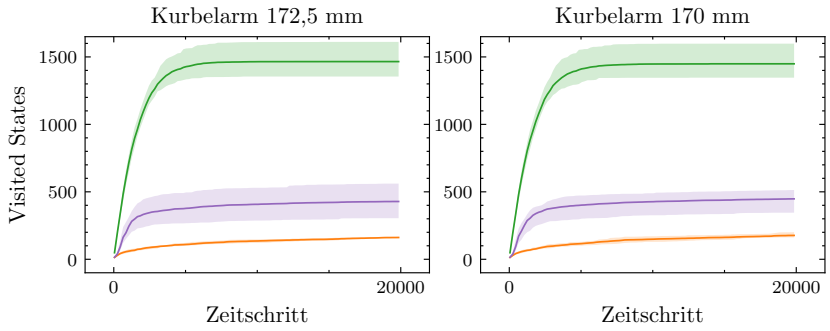


Bild 84: Darstellung der notwendigen Zustände für die Agenten mit Benchmark Hyperparametern (grün), optimierten Hyperparametern (violett) und optimierten Hyperparametern und vortrainierten DQN (orange)

unter ähnlichen Randbedingungen (Kurbelarme der Länge 172,5 mm und 170,0 mm), welche nachfolgende Produktgenerationen simulieren sollen, verwendet. Da sowohl die individuellen Rewards für Spannung und Masse als auch deren gewichtete Summe gegen einen Wert über 0,0 über die Zeitschritte konvergieren, konnte gezeigt werden, dass die vortrainierten Netze ebenfalls Merkmalskombinationen finden, welche allen Anforderungen gleichzeitig gerecht werden. Werden die Visited States betrachtet, hat sich gezeigt, dass der Aufwand des Anpassens der Strategie auf diese nachfolgenden Konstruktionsaufgaben ist deutlich niedriger als das Lernen ohne vortrainiertes DQN.

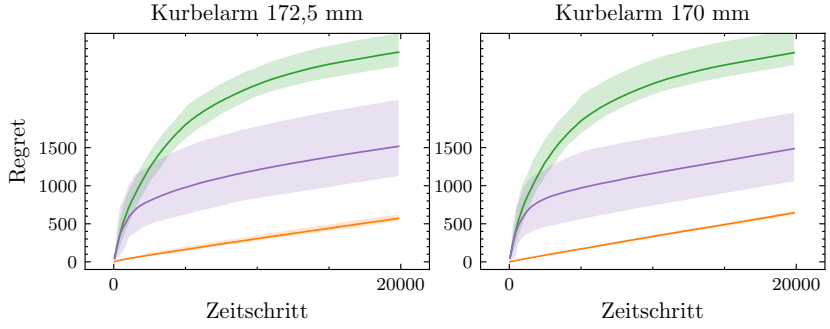


Bild 85: Darstellung des Regrets für die Agenten mit Benchmark Hyperparametern (grün), optimierten Hyperparametern (violett) und optimierten Hyperparametern und vortrainierten DQN (orange)

8 Diskussion der Erkenntnisse hinsichtlich der Forschungsfragen

Dieses Kapitel dient zur Diskussion der Erkenntnisse bezüglich der in Kapitel 3 aufgestellten Forschungsfragen und zu deren Résumé hinsichtlich der Automatisierung der Anpassungskonstruktion. Zunächst sei als erste und vermeintlich triviale Erkenntnis dieser Arbeit vorangestellt, dass jede Form der Automatisierung die Expertise bezüglich der zu automatisierenden Tätigkeit voraussetzt. Vor allem das Erfassen und Abbilden der Zusammenhänge in computer-verarbeitbaren Modellen gibt den Rahmen der Möglichkeiten für die automatisierende Technologie vor. In dieser Arbeit wurde daher der Begriff des computer-verarbeitbaren Modells der Anpassungskonstruktion hervorgehoben, welches sich in der ersten Forschungsfrage widerspiegelt: Wie muss die Anpassungskonstruktion und deren Konstruktionsaufgaben modelliert werden, um als rechner-verarbeitbares Modell die Automatisierung zu ermöglichen? Folgende Erkenntnisse konnten zur Beantwortung dieser Frage beigetragen werden:

- Das Modell beschreibt den Prozess der Anpassungskonstruktion. Es ist folglich ein Prozessmodell und muss als Computermodell formuliert werden, um computer-verarbeitbar zu sein [85]. Ziel, Inhalt, Methode, Hilfsmittel und optionale Randbedingungen müssen für die Modellierung definiert werden [92]. Das Ziel ergibt sich aus der Aufgabenstellung dieser Arbeit. Der Inhalt des Modells wurde über die Analyse der wissenschaftlichen Grundlagen für die Anpassungskonstruktion erarbeitet. Dort hat sich gezeigt, dass eine Vielzahl von Definitionen für die Anpassungskonstruktion existieren, welche sich in einander überführen lassen. In deren Konsens basiert die Anpassungskonstruktion auf einem Vorgänger und erbt von diesem die Prinzipielle Lösung. Diese Arbeit schlägt zur Abstraktion der Anpassungskonstruktion vor, die Anpassung des Vorgängers chronologisch und hierarchisch zu betrachten. Ersteres stellt die zeitliche Abfolge der Änderungen als Konstruktionsaufgaben dar. So ergeben sich aus der ersten Konstruktionsaufgabe Randbedingungen für die Zweite und alle Nachfolgenden. Zweiteres drückt die Informationen der Prinzipiellen Lösung aus und stellt die Relationen zwischen Anforderungen, Merkmalen und Eigenschaften dar.
- Als Modellierungsmethode wurde in dieser Arbeit ein semantisches Netz gewählt. Es beschreibt die formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung der Anpassungskonstruktion [162]. Es wird durch Konzepte sowie deren Axiome computer-verarbeitbar darstellt. Gemeinsam

bedeutet, dass sich das Modell auf möglichst viele, unterschiedliche Anpassungskonstruktionen übertragen lassen soll. Hierfür wurde die Anzahl iterativ bis auf die in Tabelle 4 und Tabelle 5 vorgestellten Konzepte und deren Axiome kondensiert. Außerdem ist das Modell flexibel erweiterbar, um gegebenenfalls spezifische Produktentwicklungsprozesse abbilden zu können. Die benutzen Hilfsmittel können Tabelle 13 des Anhangs entnommen werden.

- Die zentrale Randbedingung ist, dass das Modellieren die Aufgabe der Produktentwickelnden als Experten der betrachteten Domäne ist. Damit wird die Sichtweise weiterverfolgt, dass die Produktentwickelnden weiterhin die Rolle der Protagonisten erfüllen, weil die finale Entscheidung über die Qualität der Ergebnisse des Produktentwicklungsprozesses in ihrer Hand liegt. Da jedoch nicht davon auszugehen ist, dass Produktentwickelnde ohne spezifische Weiterbildung computer-verarbeitbare Modelle erstellen können, müssen sie hierbei unterstützt werden. In dieser Arbeit wurde der Ansatz verfolgt, die Modellierung des Anpassungsprozesses auf die Produktentwickelnden auszurichten und deren Überführung in ein computer-verarbeitbares Modell davon zu entkoppeln (s. Bild 86) Der Prozess der Modellierung ist folglich zweigeteilt. Einerseits in die Beschreibung der Anpassungskonstruktion mit Hilfe einer GUI innerhalb der Arbeitsumgebung der Produktentwickelnden (s. ② in Bild 86). Hier wurde dies exemplarisch in der CAx-Umgebung Siemens NX demonstriert. Andererseits in deren Überführung in ein computer-verarbeitbares Modell – eine Wissensrepräsentation der Anpassungskonstruktion (s. ③ in Bild 86). Diese ist automatisiert und erfolgt im Hintergrund ohne Arbeitsaufwand für die Produktentwickelnden.

Aus dieser Herangehensweise für die Modellierung ergeben sich folgende Limitationen. Die prinzipielle Lösung des Vorgängers wird vorausgesetzt. Zusätzlich wird vorausgesetzt, dass sich aus dieser die Zuordnung von Merkmalen zu Eigenschaften bzw. Anforderungen ableiten lässt. Dies ist nicht zwangsläufig gegeben und muss andernfalls vor der Modellierung zusätzlich erarbeitet werden. Des weiteren darf hier die Anpassungskonstruktion nicht die Neukonstruktion einer Komponenten enthalten. Die Neukonstruktion beinhaltet meist die Lösung von Problemen im Konstruktionsprozess. Hier sind Lösung und/oder der Weg zur Lösung zumindest teilweise unbekannt. Für deren Bearbeitung werden kreative Tätigkeiten benötigt, die durch die hier erarbeiteten Technologien nicht automatisierbar sind.

Neben der Modellierung der Anpassungskonstruktion als zu automatisierende Domäne wurden die wissenschaftlichen Grundlagen für deren Automatisierung selbst beleuchtet. Da sich dort wenige Publikationen explizit auf die

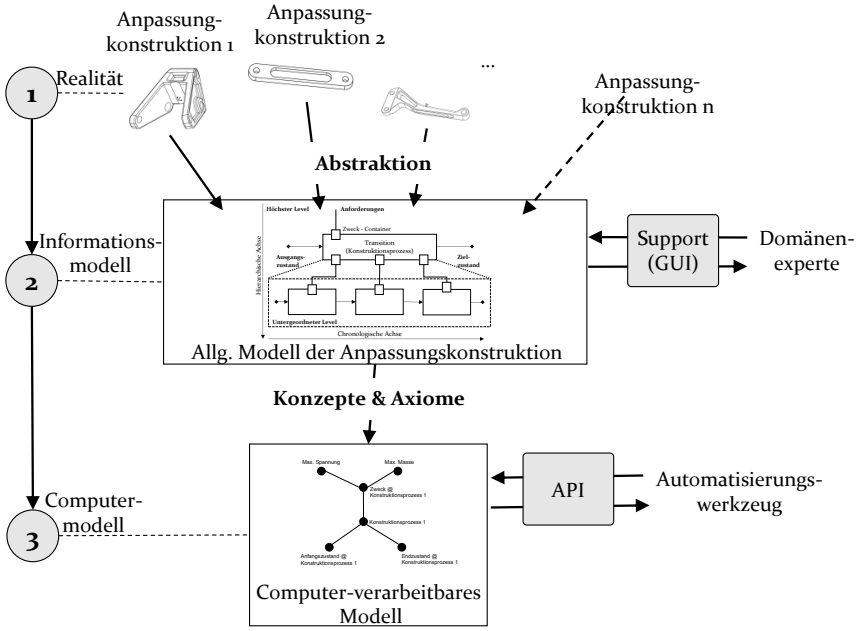


Bild 86: Kernelemente der Modellierung einer computer-verarbeitbaren Repräsentation der Anpassungskonstruktion zur deren Automatisierung

Anpassungskonstruktion beziehen, wurden Methoden und Werkzeuge analysiert, die die qualitative Gestalt – Merkmale und deren Ausprägungen – als Ergebnis generieren. Dabei wurde festgestellt, dass die Vorhersage von neuen Merkmalsausprägungen für neue Anforderungen für viele Technologien herausfordernd ist, da die Anzahl der gleichzeitig abzuschätzenden Zielgrößen (Merkmale) sehr hoch ist. Vor allem datengetriebene Ansätze benötigen hierfür eine ausreichend große Datengrundlage, d. h. das Verhältnis zwischen gleichzeitig abzuschätzenden Merkmalen und vergleichbaren Produkten muss Zugunsten der Anzahl der Produkte sein. Um dies zu überwinden, stellt diese Arbeit das Selbstverstärkende Lernen als datenunabhängige Technologie für die Automatisierung der Anpassungskonstruktion vor.

Hier ist erneut zu betonen, dass Produktentwickelnde als Experten der Domäne, die Ergebnisse – die gefundenen Merkmalskombinationen und deren Eigenschaften – bewerten. Es kann jedoch nicht vorausgesetzt werden, dass sich Produktentwickelnde mit den Algorithmen des Selbstverstärkenden Lernens detailliert auseinander setzten müssen, um diese zur Automatisierung einsetzten zu können. Vielmehr müssen die Eingriffs- und Beobachtungsmöglichkeiten für die Produktentwickelnden aufbereitet werden. Daher wurde die zweite Forschungsfrage: wie können Produktentwickelnde die Automa-

tisierung der Anpassungskonstruktion durch Selbstverstärkendes Lernen einsetzen und evaluieren, in diese zwei Aspekte unterteilt. Hinsichtlich des ersten Aspekts, welche Stellgrößen müssen die Produktentwickelnden beim Einsatz des Selbstverstärkenden Lernens kennen und beachten, konnten folgende Erkenntnisse erarbeitet werden:

- Die Produktentwickelnden müssen einerseits die Konstruktionsaufgabe als Lernumgebung formulieren können und anderseits die sogenannten Hyperparameter – Parameter, die das Verhalten des Agenten beeinflussen aber vor dessen Training definiert werden müssen – definieren. Deren Auswirkungen auf das Verhalten des Selbstverstärkenden Lernprozesses konnte in allen betrachteten Anwendungsfällen in Abschnitt 7.3 gezeigt werden.
- Das Formulieren der Konstruktionsaufgabe als Lernumgebung beinhaltet das Festlegen der änderbaren Merkmale, deren maximale und minimale Werte, sowie die minimale Anpassungsrate und das Übergeben der Anforderungen. Die hier vorgestellte Methode überführt die Merkmale automatisch in den Lösungsraum und modelliert aus Anforderungen die Belohnungsfunktionen. Für die untersuchten Demonstrator Bauteile war es möglich diesen Prozess linear und für alle Bauteile gleich ablaufen zu lassen. Lediglich konkrete Werte, beispielsweise die minimalen und maximalen Werte der einzelnen Merkmale unterscheiden sich. Dies limitiert einerseits die Modellierung des Lösungsraums, da auf individuelle Gestaltungsmöglichkeiten, wie beispielsweise individuelles Rewardshaping, verzichtet wird, vereinfacht die Arbeit für die Produktentwickelnden, da sie hier keine zusätzliche Aufmerksamkeit aufbringen müssen. In Abschnitt 6.2 konnte gezeigt werden, wie sich dieses Konzept mit Hilfe einer GUI einfach umsetzten und in die Entwicklungsumgebung integrieren lässt.
- Das Definieren und Anpassen der Hyperparameter ist meist ein iterativer und aufwendiger Prozess. Dies gilt für Selbstverstärkendes Lernen mehr als für etablierte Maschinelle Lernprozesse, da momentan noch keine Opensource Werkzeuge zur Optimierung zur Verfügung stehen. Außerdem müssen im Vergleich zu anderen maschinellen Lernverfahren mehr Hyperparameter gleichzeitig betrachtet werden. Diese Arbeit hat zuerst alle Hyperparameter identifiziert und zusammengetragen (s. Tabelle 8). Anschließend wurden sie in Gruppen unterteilt, beispielsweise eine Gruppe für die Hyperparameter, die sich auf die Anpassungsrate der Gewichte beziehen. Mit Hilfe dieser Untergliederung werden weniger Hyperparameter gleichzeitig betrachtet. Folglich können die Effekte der Änderung eines Hyperparameters einfacher nachvollzogen werden. Allerdings besteht das Risiko, dass Wechselwirkungen zwischen den Gruppen nicht berücksichtigt werden. Da die Hyperparameter nach deren Einsatz im Selbstverstärken-

den Lernen sinnvoll untergliedert wurden, wurde davon ausgegangen, dass diese Wechselwirkungen vergleichsweise selten auftreten. Dies hat sich bei den untersuchten Anwendungsfällen bestätigt. Darüber hinaus hat sich bei der Gegenüberstellungen von unterschiedlichen Demonstratorenbauteilen gezeigt, dass Tendenzen für sinnvolle Bereiche der Hyperparameter gleich auftreten (vgl. Heatmaps Bild 77 – 82) . Diese müssen jedoch erst durch weitere Bauteile untersucht werden, um allgemeinere Aussagen treffen zu können.

Für das Einstellen dieser Stellgrößen benötigen Produktentwickelnde zusätzliche Informationen über den Selbstverstärkenden Lernprozess. Andernfalls müssten sie nur auf Basis der vom Selbstverstärkenden Lernprozess vorgeschlagenen Merkmalskombinationen Anpassungen der Stellgrößen vornehmen, falls deren Eigenschaften die Anforderungen nicht erfüllen. Diese Arbeit hat hierfür sogenannte Indikatoren erarbeitet (s. der Reward in Unterabschnitt 5.5.2, der Loss in Unterabschnitt 5.5.1, sowie der Regret in Unterabschnitt 5.5.3). Sie geben Aufschlüsse über das Verhalten des Agenten. Dies spiegelt den zweiten Teil der Forschungsfrage wider: Welche Indikatoren können herangezogen werden, um das Lernverhalten und den Aufwand des Selbstverstärkenden Lernens zu untersuchen? Folgende Erkenntnisse konnten hierbei erarbeitet werden:

- Die Indikatoren zeigen zunächst an, ob und wie gut jede einzelne Anforderung für sich durch eine Merkmalskombination erfüllt wird. Dies sind die individuellen Rewards. Für jede Anforderung existiert ein Reward. Zusätzlich wird durch die übergeordnete Zielgröße , der Total Reward gekennzeichnet, ob alle Anforderungen durch die Merkmalskombination gleichzeitig erfüllt werden und die Konstruktionsaufgabe somit erfolgreich gelöst wurde (vgl. bspw. Bild 65, 71 oder 74). Des weiteren gibt es für die Aufwandsabschätzung den hardwareunabhängigen Indikator der Visited States. Dieser zählt die benötigten Analyseschritte, da sie den Hauptaufwand beim Training darstellen. Zuletzt wird der Regret für den Vergleich von verschiedenen Selbstverstärkenden Lernprozessen angegebenen. (vgl. bspw. Bild 67, 72 oder 75). Dieser wird vor allem beim Vergleich verschiedener Hyperparameterkombinationen herangezogen, da er die momentane Strategie mit der bestmöglichen, bekannten Strategie ins Verhältnis setzt.
- Bei der Betrachtung der Indikatoren müssen zwei grundsätzliche Fälle unterschieden werden. Erstens: der Selbstverstärkende Lernprozess findet Merkmalskombinationen, die den Anforderungen an die Konstruktionsaufgabe entsprechen. In diesem Fall sind alle individuellen Rewards und deren gewichtete Summen, der Total Reward, größer als 0,0. Hier kann und sollte, wenn die Strategie in nachfolgenden Produktgenerationen wieder-

verwendet werden soll, diese im Hinblick auf den benötigten Aufwand zum Erreichen dieser Merkmalskombinationen optimiert werden. Im Wesentlichen sollte hierbei die Explorationsphase des Selbstverstärkenden Lernprozesses verkürzt werden, so dass sich der Agent schneller auf eine Strategie festlegen muss. Für den zweiten Fall – der Selbstverstärkende Lernprozess findet keine passende Merkmalskombination – dem Selbstverstärkenden Lernprozess mehr Raum für Exploration gegeben werden.

Sowohl bei den Stellgrößen als auch bei den Indikatoren wurde entschieden, ob diese für die Produktentwickelnden notwendig und sinnvoll sind. Die Zusammenfassung der Eingriffsmöglichkeiten auf die vorgestellten Stellgrößen limitiert die Gestaltungsmöglichkeiten des Selbstverstärkenden Lernprozesses zu Gunsten eines handhabbaren Werkzeugs. Zusätzliche Optionen wie das genannte Rewardshaping oder andere Erweiterungsmöglichkeiten des Selbstverstärkenden Lernprozesses reduzieren unter Umständen den Aufwand für das Training weiter. Sie wurden bewusst nicht den Produktentwickelnden in den GUIs angeboten. Zukünftig ist vorstellbar diese in einem Expertenmodus optional anzubieten, wie dies beispielsweise bei der Topologieoptimierung üblich ist.

Als Résumé wird auf Basis dieser Erkenntnisse die dritte Forschungsfrage beantwortet: Welche Kriterien sollten Konstruktionsaufgaben erfüllen, damit eine Automatisierung durch Selbstverstärkendes Lernen gerechtfertigt ist? Grundsätzlich sollte Selbstverstärkendes Lernen zum Automatisieren von Konstruktionsaufgaben als Alternative zu datengetrieben Werkzeugen und Werkzeugen der Mehrzieloptimierung betrachtete werden, da alle drei die Relationen von Anforderungen zu Merkmalen approximieren können. Selbstverstärkendes Lernen ist aktuell innovativ im Vergleich zu den diesen weitaus etablierten Werkzeugen. Daher ist es weniger ausgereift und aufwendiger einzusetzen. Momentan existieren keine kommerziellen Werkzeuge und die Anzahl der Publikationen ist vergleichsweise gering. Selbstverstärkendes Lernen kann jedoch das Portfolio der Automatisierungstechnologien erweitern, wenn die Datengrundlage der Konstruktionsaufgabe nicht für datengetriebene Werkzeuge ausreicht, beispielsweise durch eine zu geringe Anzahl vergleichbarer Produkte oder eine zu hohe Anzahl gleichzeitig abzuschätzender Merkmale, oder die Randbedingungen der Konstruktionsaufgabe zu häufig für die effiziente Verwendung eines Algorithmus der Mehrzieloptimierung angepasst werden, beispielsweise für die individuelle Produkte. Dort ist die Frequenz der Anpassungskonstruktionen sehr hoch und die Änderungen der Merkmalsausprägung vergleichsweise gering. Diese Kombination bietet sich für Selbstverstärkendes Lernen an, da durch die Übertragungsmöglich-

keiten der gelernten Strategie der Aufwand zum adaptieren einer neuen Anpassungskonstruktion potentiell sehr gering ist.

Außerdem bietet Selbstverstärkendes Lernen die Möglichkeit Einblicke in die Zusammenhänge des Anpassungsprozesses zu erhalten. Das DQN, welches auf eine Anpassungskonstruktion trainiert wurde, regiert auf einen übergebene Zustand mit einer Aktion. Bild 87 zeigt die Reaktionen eines DQNs zur Anpassung einer Zahngeometrie aus [P8]. Aus der Folge von Zuständen und Aktionen über die Zeitschritte lässt sich ableiten, welches Merkmal wie

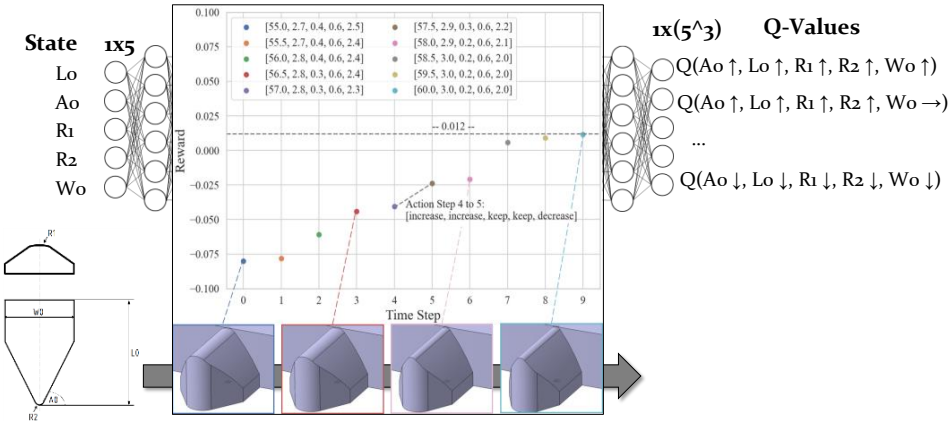


Bild 87: Kernelemente der Modellierung einer computer-verarbeitbaren Repräsentation der Anpassungskonstruktion zur deren Automatisierung

Darüber hinaus hat Selbstverstärkendes Lernen das Potential Wissen über die Anpassung in Form des DQNs zu speichern. Diese Arbeit zeigt die hierfür die Übertragung einer gelernten Strategie von einem Produkt auf dessen Nachfolger. Wenn nicht Strategien zur Anpassung einzelner Produkte, sondern zur Anpassung genereller Features, beispielsweise die Anpassung von Kerben oder abrupten Querschnittsänderungen, welche zur Spannungsüberhöhungen führen, gelernt werden, können diese auf ein breiteres Produktspektrum wieder angewandt werden. Diese Idee wird im folgendem Ausblick aufgegriffen.

9 Zusammenfassung und Ausblick

Die Anpassungskonstruktion ist eine Konstruktionsart mit limitierten konstruktiven Freiheitsgraden. Wie der Name erkennen lässt, wird ein Vorgängerprodukt auf neue Anforderungen angepasst. Dabei bleibt die prinzipielle Lösung des Vorgängers bestehen, so dass die Anpassungen im Bereich der Merkmale verortet sind. Trotz dieser Limitationen kann die Automatisierung von Anpassungskonstruktionen herausfordernd sein, wenn beispielsweise die Produkte selbst komplizierte Zusammenhänge zwischen Anforderungen und Merkmalen beinhalten. Aus der in dieser Arbeit gezeigten Analyse der wissenschaftlichen Grundlagen lässt sich ableiten, dass vor allem komplizierte Produkte mit vielen Merkmalen für datengetriebene Lösungen wie beispielsweise überwachte, Maschinelle Lernverfahren problematisch sind, weil häufig nicht ausreichend viele Produkte analysiert werden können, um Zusammenhänge zwischen den Anforderungen und Merkmalen ausreichend genau zu approximieren. Die vorhandenen datengetriebenen Automatisierungswerkzeuge skalieren daher in viele Fällen schlecht mit der Anzahl der Merkmale. Alternativ können Optimierungsalgorithmen eingesetzt werden, da sie keine Datengrundlage für die Berechnung benötigten. Allerdings müssen Optimierungsalgorithmen bei einer Änderung der Randbedingungen wieder neu zu Konvergenz gebracht werden. Folglich gehen die bisher erarbeiteten Erkenntnisse ohne zusätzliche Hilfsmittel verloren. Die Übertragbarkeit auf neue Randbedingungen oder nachfolgende Anpassungskonstruktionen ist hier ohne zusätzlichen Aufwand nicht möglich.

Um diese Herausforderungen zu überwinden, stellt diese Arbeit das Selbstverstärkende Lernen als Automatisierungswerkzeug vor. Die Vorgehensweise ähnelt der Mehrzieloptimierung, bei der sich die Anforderungen an das Produkt in einzelnen Zielfunktionen widerspiegeln. Der Optimierungsalgorithmus sucht für diese meist konträren Zielfunktionen die Merkmalskombinationen im Lösungsraum, welche alle Zielfunktionen gleichzeitig optimal erfüllen. Beim Selbstverstärkenden Lernen werden die Zusammenhänge nicht durch einen Optimierungsalgorithmus im klassischen Sinne erfasst, sondern ein tiefes, Neuronales Netz für das Auffinden der passenden Merkmalskombinationen trainiert. Dieses sogenannte Deep Q-Network bildet die Strategie des Selbstverstärkenden Lernprozesses ab. Es reagiert auf Merkmalskombinationen mit Anpassungsvorschlägen, die die Merkmale so ändern, dass die Anforderungen letztendlich erfüllt werden. Für das Training wird kein Datensatz benötigt, da der Selbstverstärkende Lernprozess Erfahrungen mit der Konstruktionsaufgabe eigenständig generiert und auf deren Basis lernt. Des weiteren ist es möglich, die erlernten Zusammenhänge

abzuspeichern und wiederzuverwenden, beispielsweise für die nachfolgende Anpassungskonstruktion. Selbstverstärkendes Lernen ist damit keine konkurrierende Technologie, die bisherige Ansätze ablösen soll, sondern eine Alternative falls Skalierbarkeit oder Übertragbarkeit primär gefordert werden.

Die Forschungslandschaft hat gezeigt, dass Selbstverstärkendes Lernen nach momentanem Stand in der Produktentwicklung noch kaum genutzt wird. Diese Arbeit ermöglicht Produktentwickelnden den Einsatz dieser neuen Technologie. Dabei wurde untersucht und abgewägt, welche Eingriffsmöglichkeiten Produktentwickelnde benötigen und wie sie deren Auswirkungen und das gelernte Verhalten beurteilen können. Diese Eingriffsmöglichkeiten wurden als Stellgrößen und Indikatoren exemplarisch in einer Applikation innerhalb der Entwicklungsumgebung Siemens NX implementiert, sodass Produktentwickelnde nicht ihre gewohnte Arbeitsumgebung verlassen müssen. Anhand dieser Umsetzung wurde mit Hilfe verschiedener Anwendungsbeispiele die Machbarkeit, die Auswirkungen der Stellgrößen und die Übertragung auf nachfolgende Anpassungskonstruktionen untersucht. Die gezeigten Anwendungsbeispiele bestätigen die Übertragbarkeit der Strategie im Rahmen ähnlicher Produkte. Dort konnte der Trainingsaufwand des Selbstverstärkend Lerners um bis zu 90% durch die Übertragung der Strategie reduziert werden, ohne dass die gefundenen Merkmalskombinationen die Anforderungen schlechter erfüllen. Die Übertragung im Rahmen ähnlicher Produkte ist für die gezeigten Fälle plausibel, da die Auswirkungen der Randbindungsänderungen auf die Zielgrößen klein und stetig sind. Beispielsweise wurde eine zulässige, maximale Spannung als Anforderung vorgegeben. Durch eine Änderung der Randbedingungen wie beispielsweise die Verlängerung des Hebelarms der angreifenden Kraft, wird die Amplitude des Spannungsverlaufs geändert, nicht der Verlauf selbst. Dieser wird vom übertragenen DQN nach wie vor erfasst. Erst wenn die Änderung der Randbedingungen unstetige Verläufe hervor rufen würde, ist es wahrscheinlich, dass das DQN nicht mehr übertragen werden kann.

Dies sollte bei den zukünftigen Überlegungen für den Einsatzes des Selbstverstärkenden Lernens berücksichtigt werden. Zukünftig sollten nicht Bauteile, sondern in verschiedenen Bauteilen wiederkehrende Aspekte gelernt werden. Da diese auf viele Bauteile übertragen werden können und damit diesen Vorteil des Selbstverstärkenden Lernens besser ausnutzt. Ein Beispiel hierfür ist der Umgang mit einer Kerbe im Spannungsverlauf. Der Effekt der Spannungsüberhöhung durch eine Kerbe im Spannungsverlauf tritt in verschiedenen Bauteilen gleichermaßen auf und kann beispielsweise durch Umleitung des Spannungsverlaufes oder durch Abrundung der Querschnittsänderung reduziert werden. Um eine kritische Kerbwirkung eindeutig zu identifizieren, reichen jedoch geometrische Merkmale alleine nicht aus. Erst durch eine

Kombination mit dem Spannungsverlauf kann eine Aussage getroffen werden, ob an der entsprechenden Stelle kritische Spannungsüberhöhungen auftreten. Diese Informationen sieht der Selbstverstärkende Lernprozess bisher jedoch nicht, da der Input des DQNs aus der Merkmalskombination besteht. Alle weiteren Informationen werden dem Selbstverstärkenden Lernprozess vorenthalten. Folglich muss der Input des DQNs erweitert werden. Möglich ist die Merkmalskombination, welche bisher durch einen eindimensionalen Vektor beschrieben wird, in einer zweiten Dimension mit zusätzlichen Informationen anzureichern. Für den Fall der Kerbwirkung könnte beispielsweise für jedes Merkmal der Abstand zum Spannungsmaximum in globalen Koordinaten angegeben werden. Merkmale deren Geometrie eine Querschnitteinschnürung bedeuten und gleichzeitig nahe am Spannungsmaximum sind, können so als kritisch für die Spannungsüberhöhung in Folge einer Kerbwirkung eingestuft und gelernt werden. Das Spektrum der Informationen, die die Merkmalskombination erweitern sollten ist mannigfaltig und bietet Potential für zukünftige Forschungsarbeit im Bereich des Selbstverstärkenden Lernens für die Produktentwicklung. Die Identifikation und Interpretation der benötigten Informationen ist wiederum Aufgabe der Produktentwickelnden, die entscheiden müssen, welche Informationen relevant sind und in der Lernumgebung abgebildet werden sollten.

10 Summary and Outlook

Automation describes the transfer of tasks from humans to machines. In the context of adaptation design, these tasks involve defining the product´s characteristics to meet new requirements while maintaining the same principal solution. They are particularly challenging if characteristics simultaneously influence multiple, competing properties. Here, the best possible compromise of the characteristics must be found. Depending on the context of the adaptation task, they can be automated by optimisation algorithms or by data-driven approaches. However, data-driven approaches require that the underlying database contains all relevant information for recognising the relationships between characteristics and resulting properties. For their part, optimisation algorithms must be brought to convergence whenever the boundary conditions change. For the intersection of adaptation tasks, for which the boundary conditions change regularly due to changes in requirements and at the same time no sufficient database is available, this work presents reinforcement learning as an extension of the current automation possibilities. Reinforcement learning is described as the learning of an agent based on experience with an environment. For this work, the environment represents the working environment of the product developers in the context of adaptation design. The agent is trained to take over tasks by adapting characteristics and receiving the product properties as feedback in return. This work does not to develop a new algorithm for reinforcement learning, but takes reinforcement learning into product development, defines the role of the product developers in its use and to research the transferability of the experience gained from one adaptation task to a subsequent one. The following questions were developed for this purpose:

1. How must the adaptation design process and its tasks be modelled to enable automation?
2. How can product developers use and evaluate reinforcement learning to automate adaptation design tasks?
    a) Which configurable settings do product developers need to consider when using reinforcement learning?
    b) What indication mechanisms can be used to evaluate the learning behaviour and effort of reinforcement learning?
3. What criteria should adaptation tasks fulfil in order to justify automation using reinforcement learning?

The first research question is answered in chapter 5.1 of the thesis, which summarises the contents of [P6]. The adaptation design is modelled as a

process of states and their transitions. States describe characteristic values at a point in time. Transitions between states are construction tasks. Requirements that define the purpose of the task are assigned to them. The concept for reinforcement learning for design tasks was introduced in [P9] and tested for further application scenarios in [P5] and [P8]. With regard to research question 2, this dissertation explicitly places the concept in the context of product development. For this purpose, the design tasks are transferred into learning environments in chapter 5.3. For example, guidelines are defined on how product developers can integrate their expert knowledge into the learning environment there. Furthermore, the findings from [P9] are used to research configurable settings (see chapter 5.4). Their effects can be traced through the indicators from chapter 5.5, so that the fine-tuning of the learning behaviour by product developers can be carried out in the design environment. The third research question clarifies which criteria must be present for reinforcement learning to be used in product development. For this purpose, the approach competes with those of data-driven machine learning and established optimisation algorithms, such as Generative Design. If a sufficient data set is already available to map the design task with a supervised learning process, learning by experience (reinforcement learning) means considerable additional effort. If this is not the case, optimisation algorithms or reinforcement learning can be used. Here, reinforcement learning is to be considered if the experiences of one design task can be used for further, subsequent ones. The concept of transfer learning describes how an agent can be used in a subsequent design task without having to learn it from scratch by using the experience already gained in the previous design task as a basis. The concept extends the previously published considerations and will be briefly illustrated in the following using an example from bulk sheet metal forming. Figure 87 on page 151 shows the result (the strategy after completed training) of a reinforcement learning agent for adapting a tooth geometry with 5 characteristics (L_0, A_0, R_1, R_2 and W_0). They have to be adapted in such a way that two competing requirements (forming force and degree of mould filling) are optimally fulfilled. The forming force should be as small as possible and an upper limit should not be exceeded (maximum requirement). The upper limit is represented by the individual *reward* of 0.0 as feedback to the agent. The degree of form filling is a minimum requirement. The equally weighted sum of the individual *rewards* is used to calculate the *reward* shown in Figure 87 on page 151. A feature combination that receives a *Reward* ≥0,0 thus fulfils all requirements. The strategy used by the agent to achieve these characteristics is mapped by a deep neural network. The characteristics are passed to its input neurons. On the output neurons, a possible combination to change the characteristics (actions) is returned. The displayed progression (*Reward* over *Time Steps*) shows how the agent adapts the characteristics. The states

(combinations of characteristics) passed through are given in the legend. The agent moves in a discrete solution space so that no post-processing needs to take place to obtain manufacturability geometries. For evaluation purpose, the solution space was completely simulated and the dashed line marks the optimal solution. Figure 87 on page 151 shows that after training is complete, the agent moves from any starting point in the solution space towards the optimal solution by gradually adjusting the features. This strategy is stored in the weights of the deep neural network and, unlike an optimisation algorithm, can be applied to the adaptation of the next tooth geometry, enabling transfer learning from one adaptation task to the subsequent one. Details of this are presented in chapter 5.6.

Anhang

Appendix: Exemplarische Umsetzung in Python

Paketdefinitionen und Umgebung

Tabelle 13: Relevante, verwendete Programme, Pakete und deren Versionen:

Name	Version	Anmerkung
Siemens NX	1899	CAx
Protégé	5.5.0	Modellierung
Python	3.8	
Annaconda	2021.11	
PyTorch	1.8.1	
Pandas	1.1.3	
Numpy	1.19.2	

Klassen und deren Funktionen

Die nachfolgend vorgestellte Implementierung orientiert sich an [P9] und dient ausschließlich der Nachvollziehbarkeit der Ergebnisse.

Listing 1: Agent Modul

```
# import external libaries
import os
import time
import copy
import itertools
import numpy as np
import pandas as pd
from pathlib import Path
import collections
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.tensorboard import SummaryWriter

# import local modules
from DQN import DQN
from Environment import Environment
from ReplayBuffer import ExperienceBuffer
from ResultFunction import RewardFunction

# tuple preparation for replay buffer
Experience = collections.namedtuple(
```

```
'Experience', field_names=['state', 'action', 'reward', 'done', '
  new_state'])

class Agent():
'''
Description:
This agent plays with CAD part designer environment.
His tasks are:
1) selects and executes Action (-> revceives Observation & Reward)
2) stores the Experience (or transitions) in buffer
3) computes Temporal Difference Error, i.e. the (RL) loss, for DQN
  update
4) update DQN to improve Q-Value Approximation
Parameters:
agent_id                unique agent_ID for dataframe
env                     environment for the agent to play in
discount_factor         discount of future rewards
epsilon_start           epsilon for exploitation-exploration strategy (
  epsilon-greedy policy)
epsilon_final           minimal epsilon to converge
epsilon_last_step       steps until epsilon should converge to final
  value
num_episodes            episodes to play
max_steps_per_episode   finite number of steps to play in a episode
optimizer               optimizer for DQN_updates
learning_rate           learning rate for weight update of the policy
  network (i.e learning rate for TD-learning)
policy_net              main policy network
target_net              target network
replay_buffer_size      size of replay buffer object to sample
  experience from
replay_start_size       count of time steps to wait before starting
  training to populate the replay buffer
sync_target_steps       n-Steps for periodic synchronization of target
  network
device                  "cpu" or "cuda"
tensorboard_log         directory to store tensorboard loggings
agent_log               directory to store agents' policy
dataframe_dir           directory to store dataframe (plotting)
Class functions:
sync_target_net         synchronize target network with policy network
  periodically
reset                   reset state and total reward
decay_epsilon           decay epsilon over time
play_game               play a whole game, i.e. learn from environment
  interactions
play_step               play one time step
update_DQN              update our policy network -> needs to compute
  loss
calc_loss               computation of the loss
callback                monitors agent's learning process and saves best
   policy -> needs to evaluate
evaluate                evaluate the agent's policy to get metrics (no
  policy/replaybuffer updates)
External class functions:
step                    take step in the environment and receive
  feedback
```

```
append                  add transition (state, action, reward, done,
  next_state) to replay buffer object
sample_minibatch        sample random mini-batch of transitions from
  replay buffer object
'''

def __init__(self, agent_id, env, discount_factor, learning_rate,
  num_episodes, max_steps_per_episode,\
epsilon_start, epsilon_final, epsilon_last_step, replay_buffer_size,
  sync_target_steps, \
batch_size, replay_start_size, device, policy_net, target_net, \
tensorboard_log=None, agent_log=None, dataframe_dir=None):

self.env = env                                              # Environment
self.state_size = self.env.observation_space.shape[0]   # DQN-INPUT
self.action_size = np.prod(self.env.action_space.nvec)  # DQN-OUTPUT

# store requirements from RewardFunction
self.requirement_keys = copy.deepcopy(self.env.reward_function.
  requirements)
self.requirement_keys.append('total')

# list of all possible actions
action_variations = []
for item in itertools.product([0,1,2], repeat=self.state_size):
action_variations.append(np.array(item))
self.action_variations = np.asarray(action_variations)

# Agent Reinforcement Learning set up
self.num_episodes = num_episodes
self.max_steps_per_episode = max_steps_per_episode
self.epsilon_start = epsilon_start
self.epsilon_final = epsilon_final
self.epsilon_last_step = epsilon_last_step
self.discount_factor = discount_factor

# Agent Deep Learning set up
self.replay_buffer = ExperienceBuffer(replay_buffer_size)
self.replay_start_size = replay_start_size
self.batch_size = batch_size
self.sync_target_steps = sync_target_steps

# DQN set up
self.policy_net = policy_net
self.policy_net = self.policy_net.float()
self.target_net = target_net
self._sync_target_net()
self.target_net = self.target_net.float()
self.optimizer = optim.Adam(self.policy_net.parameters(), lr=
  learning_rate)

# optional device for computations (CPU or GPU)
self.device = device

# Loggings
self.tensorboard_log = tensorboard_log
self.agent_log = agent_log
self.dataframe_dir = dataframe_dir
```

```
self.agent_id = agent_id

pass

def _reset(self):
'''
Description:
Function reset the agent and the environment.
Moreover, it resets the episodic reward dict.
'''
self.state = self.env.reset()
empty_list = [[] for _ in self.requirement_keys]
self.episodic_reward = dict(zip(self.requirement_keys, empty_list))

def _save_checkpoint(self, episode):
'''
Description:
Function saves model's weights in the defined `agent_log`.
'''
dir_path = self.agent_log
Path(dir_path).mkdir(parents=True, exist_ok=True)
model_path = os.path.join(dir_path, 'checkpoint_{:03d}.ckp'.format(
  episode))
torch.save({'state_dict': self.policy_net.state_dict()}, model_path)

def _restore_checkpoint(self, episode_n):
'''
Description:
Function loads checkpoints to initialize policy network and target
  network.
'''
model_path = os.path.join(self.agent_log, 'checkpoint_{:03d}.ckp'.format
  (episode_n))
ckp = torch.load(model_path, self.device)
self.policy_net.load_state_dict(ckp['state_dict'])
self._sync_target_net()

def _sync_target_net(self):
'''
Description:
Function synchronizes weights of target network with weights of policy
  network.
'''
self.target_net.load_state_dict(self.policy_net.state_dict())

def _decay_epsilon(self, timestep, mode='linear'):
'''
Description:
Function implements different epsilon decay schedules.
'''
if timestep==1: self.epsilon = self.epsilon_start

if mode=='linear':
self.epsilon = max(self.epsilon_final, \
self.epsilon_start + timestep * (self.epsilon_final-self.epsilon_start)
  / (self.epsilon_last_step+np.finfo(float).eps))
elif mode=='exponential':
epsilon_decay_rate = 0.01
```

```
self.epsilon = self.epsilon_final + (self.epsilon_start - self.
  epsilon_final) * \
np.exp(-epsilon_decay_rate * timestep)
elif mode=='base':
if self.epsilon > self.epsilon_final:
self.epsilon *= 0.9995
return self.epsilon

def play_step(self, policy_net, epsilon=0.0):
'''
Description:
Function is resposible for:
1) performings an epsilon-greedy exploration.
2) storing each reward (overall & sub-objectives) at every step in the `
  episodic_reward` dict
3) storing experience tuple in the replay buffer
4) check for terminate state (if episode is over, we compute the `mean
  evaluation rewards`)
'''
done_reward = None

# epsilon-greedy action selection
if np.random.random() < epsilon:
action = self.env.action_space.sample()
else:
state = torch.FloatTensor(self.state).to(self.device)
q_vals = policy_net(state)
_, action_idx = q_vals.max(dim=0)
action = np.asarray(self.action_variations[action_idx])

# take step in the environment
new_state, reward, is_done, info = self.env.step(action)

# update rewards
for key, value in self.episodic_reward.items():
if key == 'total':
value.append(reward)
self.episodic_reward.update({key: value})
else:
value.append(info[key])
self.episodic_reward.update({key: value})

# add experience to replay buffer and update state
exp = Experience(self.state, action, reward, is_done, new_state)
self.replay_buffer.append(exp)
self.state = new_state

# check for terminal state
if is_done:
last_n_steps = 10
done_reward = {}
# average all rewards over last steps of an episode
for key, value in self.episodic_reward.items():
done_reward.update({key: np.mean(value[-last_n_steps:])})

return done_reward

def calc_loss(self, batch, policy_net, target_net, device='cpu'):
```

```
    '''
    Description:
    Function calculates the loss for a given mini-batch.
    '''
    states, actions, rewards, dones, next_states = batch

    # turn action vectors into indeces to fit DQN-Output dimension
    action_indices = np.where((self.action_variations==actions[:,None]).all
      (-1))[1]

    states_v = torch.FloatTensor(np.array(states, copy=False)).to(device)
    next_states_v = torch.FloatTensor(np.array(next_states, copy=False)).to(
      device)
    actions_v = torch.LongTensor(action_indices).to(device)
    rewards_v = torch.tensor(rewards).to(device)
    done_mask = torch.BoolTensor(dones).to(device)

    # get (old) predicted Q(s,a) from policy network
    q_vals = policy_net(states_v).gather(1, actions_v.unsqueeze(-1)).squeeze
      (-1)

    # get estimate of optimal future Q(s',a) from target network
    # mask last steps of episode
    # detach this tensor from computation graph to block gradients in
      backpropagation
    with torch.no_grad():
    next_q_vals = target_net(next_states_v).max(1)[0]
    next_q_vals[done_mask] = 0.0
    next_q_vals = next_q_vals.detach()
    # temporal difference target: expected Q(s,a)
    expected_q_vals = next_q_vals * self.discount_factor + rewards_v
    # Temporal difference error => MSE loss
    return nn.MSELoss()(q_vals.float(), expected_q_vals.float())

    def update_DQN(self):
    '''
    Description:
    Function starts sampling a batch from replay buffer when current
      timestep is larger
    than 'replay_start_size'. Next, the batch is forwarded to the calc_loss
      function
    to compute loss for weight update. Finally, gradients are computed and
      the optimizer
    will update the weights for the given loss.
    '''
    self.optimizer.zero_grad()
    # sample minibatch from replay buffer
    batch = self.replay_buffer.sample_minibatch(batch_size=self.batch_size)
    # calculate loss
    loss_tensor = self.calc_loss(batch, self.policy_net, self.target_net,
      self.device)
    # compute gradient
    loss_tensor.backward()
    loss_tensor = loss_tensor.cpu()
    # store TD errors for monitoring
    self.losses.append(loss_tensor.cpu().detach().numpy())
    # Adapt policy network
    self.optimizer.step()
```

```
return

def _callback(self, writer: SummaryWriter, df: pd.DataFrame, step_idx:
  int, rewards_dict: dict):
'''
Description:
This function is responsible for:
1)  Evaluation of current Policy
2)  Save best Policy
3)  Tensorboard Summary with metrics
4)  DataFrame (Plotting)
'''
print("-------------------------------------------------------")
print("Evaluating Model at step: " + str(step_idx))

n_episodes = 3

# update epsilon greedy rewards
for key, value in self.rewards.items():
value.append(rewards_dict[key])
self.rewards.update({key: value})

# evaluation of agent's current policy
eval_rewards_dict = self.evaluate()
for key, value in self.eval_rewards.items():
value.append(eval_rewards_dict[key])
self.eval_rewards.update({key: value})
if key == 'total':
# mean reward over last 3 episodes
m_reward = np.mean(value[-n_episodes:])

# speed measurement
episode = len(self.eval_rewards['total'])
speed = (step_idx - self.ts_step) / (time.time() - self.ts)
self.ts_step = step_idx
self.ts = time.time()
print(f"{step_idx}: done {episode} episodes, reward={m_reward:.3f}, eps
  ={self.epsilon:.2f}, speed={speed:.2f} steps/s")

# update best mean reward and save best policy
if self.best_m_reward is None or self.best_m_reward < m_reward:
self._save_checkpoint(episode)
if self.best_m_reward is not None:
print(f"Best reward updated from {self.best_m_reward:.3f} to {m_reward
  :.3f}")
self.best_m_reward = m_reward

# ======================================== #
#  +++ Tensorboard Summary & DataFrame +++  #
# ======================================== #

# unique IDs
ids = self.agent_id.split('_')
df.at[episode-1, 'Exp_ID'] = ids[0]
df.at[episode-1, 'State_ID'] = ids[1]
df.at[episode-1, 'Seed_ID'] = ids[2]

# steps
```

```
        df.at[episode-1, 'Step'] = step_idx

        # mean TD errors over the episode
        if self.losses:
        writer.add_scalar("loss", np.mean(self.losses), step_idx)
        df.at[episode-1, 'Loss'] = np.mean(self.losses)

        # epsilon at the timestep
        writer.add_scalar("epsilon", self.epsilon, step_idx)
        df.at[episode-1, 'Epsilon'] = self.epsilon

        # averaged speed per episode
        writer.add_scalar("speed per episode", speed, step_idx)
        df.at[episode-1, 'Speed_per_Episode'] = speed

        # additional metrics: Regret & Visited States
        writer.add_scalar("regret", self.env.regret, step_idx)
        writer.add_scalar("visited_states", len(self.env.visited_states),
          step_idx)
        df.at[episode-1, 'Regret'] = self.env.regret
        df.at[episode-1, 'Visited_States'] = len(self.env.visited_states)

        # iterate through rewards dictionary
        for key, value in self.rewards.items():
        # mean of epsilon-greedy reward (and respective sub-rewards) over last 3
            episodes
        mean_rew_str = f'mean_epsilon_greedy_reward/{key}_reward_{n_episodes}'
        mean_rew_value = np.mean(value[-n_episodes:])

        writer.add_scalar(mean_rew_str, mean_rew_value, step_idx)
        df.at[episode-1, f'mean_epsilon_greedy_reward_{key}'] = mean_rew_value

        # actual epsilon-greedy rewards at the time step
        rew_str = f'epsilon_greedy_reward/{key}_reward'
        rew_value = value[-1]

        writer.add_scalar(rew_str, rew_value, step_idx)
        df.at[episode-1, f'epsilon_greedy_reward_{key}'] = rew_value

        # iterate through evaluation rewards dictionary
        for key, value in self.eval_rewards.items():
        # mean of evaluation reward (and respective sub-rewards) over last 3
          episodes
        mean_rew_str = f'mean_evaluation_reward/{key}_reward_{n_episodes}'
        mean_rew_value = np.mean(value[-n_episodes:])

        writer.add_scalar(mean_rew_str, mean_rew_value, step_idx)
        df.at[episode-1, f'mean_evaluation_reward_{key}'] = mean_rew_value

        # actual evaluation rewards at the time step
        rew_str = f'evaluation_reward/{key}_reward'
        rew_value = value[-1]

        writer.add_scalar(rew_str, rew_value, step_idx)
        df.at[episode-1, f'evaluation_reward_{key}'] = rew_value

        return
```

```
def evaluate(self):
'''
Description:
This method evaluates our main policy across the whole episode length.
  Note, that the DQN
is turned into evaluation mode. Furthermore, we turn off gradients
  computation.
'''
eval_reward = {}
self._reset()

# evaluate policy network
self.policy_net.eval()
with torch.no_grad():
for step in range(0, self.max_steps_per_episode):
# greedy policy
state = torch.tensor(self.state).to(self.device)
q_vals = self.policy_net(state.float())
_, action_idx = q_vals.max(dim=0)
action = np.asarray(self.action_variations[action_idx])

# do step in the environment
new_state, reward, is_done, info = self.env.step(action, evaluation=True
  )
self.state = new_state

# update rewards
for key, value in self.episodic_reward.items():
if key == 'total':
value.append(reward)
self.episodic_reward.update({key: value})
else:
value.append(info[key])
self.episodic_reward.update({key: value})

# check for terminal state
if is_done:
last_n_steps = 10
# average all rewards over last steps of an episode
for key, value in self.episodic_reward.items():
eval_reward.update({key: np.mean(value[-last_n_steps:])})

# put policy networks back to train mode after evaluation
self.policy_net.train()
return eval_reward

def play_game(self):
'''
Description:
1) play step
2) save model if new mean reward was beaten
3) sync target network
4) dqn update
'''
# lists for reward monitoring
empty_list = [[] for _ in self.requirement_keys]
self.rewards = dict(zip(self.requirement_keys, empty_list))
self.eval_rewards = copy.deepcopy(self.rewards)
```

```
self.best_m_reward = None
step_idx = 0
self.ts_step = 0
self.ts = time.time()
solved = False

writer = SummaryWriter(log_dir=self.tensorboard_log)

# initialize Dataframe
eps_greedy_cols = [f'mean_epsilon_greedy_reward_{key}' for key in self.
  requirement_keys] + \
[f'epsilon_greedy_reward_{key}' for key in self.requirement_keys]
evaluation_cols = [f'mean_evaluation_reward_{key}' for key in self.
  requirement_keys] + \
[f'evaluation_reward_{key}' for key in self.requirement_keys]
cols = ['Exp_ID', 'State_ID', 'Seed_ID', 'Step', 'Epsilon', 'Loss', '
  Speed_per_Episode' , 'Regret', 'Visited_States'] + \
eps_greedy_cols + evaluation_cols

df = pd.DataFrame(columns=cols)

# walltimers
t_update = 0.0
t_callback = 0.0
t_play_step = 0.0
t_decay = 0.0
t_reset = 0.0
t_sync = 0.0
ts = time.time()

for episode in range(1, self.num_episodes+1):
t_reset_start = time.time()
self._reset()
t_reset += time.time() - t_reset_start

self.losses = []   # list to store TD errors
for step in range(0, self.max_steps_per_episode):
step_idx += 1

t_decay_start = time.time()
epsilon = self._decay_epsilon(step_idx, mode='linear')
t_decay += time.time() - t_decay_start

t_play_start = time.time()
reward = self.play_step(self.policy_net, epsilon)
t_play_step += time.time() - t_play_start

if reward is not None: # logging
t_callback_start = time.time()
self._callback(writer, df, step_idx, reward)
t_callback += time.time() - t_callback_start

if len(self.replay_buffer) < self.replay_start_size:
continue

if step_idx % self.sync_target_steps == 0:
t_sync_start = time.time()
```

```
        self._sync_target_net()
        t_sync += time.time() - t_sync_start

        # update DQN
        t_update_start = time.time()
        self.update_DQN()
        t_update += time.time() - t_update_start

        # close SummaryWriter
        writer.close()

        # save last Policy
        self._save_checkpoint(self.num_episodes+1)

        # save DataFrame
        t_csv = time.time()
        csv_path = self.dataframe_dir + f'{self.agent_id}.csv'
        df.to_csv(csv_path)
        t_csv = time.time() - t_csv

        walltime = time.time() - ts

        print(f'Complete Walltime: {walltime:.2f} seconds')
        print(f'Play-step Method: {t_play_step:.2f} seconds')
        print(f'Callback Method: {t_callback:.2f} seconds')
        print(f'Sync Method: {t_sync:.2f} seconds')
        print(f'Update Method: {t_update:.2f} seconds')
        print(f'Reset Method: {t_reset:.2f} seconds')
        print(f'CSV Method: {t_csv:.2f} seconds')

        return

    def evaluate_state(self, state):
    '''
    Description:
    Evaluates a given state and perfromces action
    '''
    eval_reward = {}

    # evaluate policy network
    self.policy_net.eval()
    with torch.no_grad():
    # greedy policy
    state = torch.tensor(self.state).to(self.device)
    q_vals = self.policy_net(state.float())
    _, action_idx = q_vals.max(dim=0)
    action = np.asarray(self.action_variations[action_idx])

    # do step in the environment
    new_state, reward, is_done, info = self.env.step(action, evaluation=True
      )

    # put policy networks back to train mode after evaluation
    self.policy_net.train()
    return eval_reward, new_state, action
```

Listing 2: DQN Modul

```
import numpy as np
import torch
import torch.nn as nn

class DQN(nn.Module):
'''
Description:
Model (only FC-Layers) receives observations as input and has to predict
the Q-Action-Values for all possible actions.
Inherits DQN class. See further information:
https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
Parameters:
n_inputs        dimension of the state (solution) space
n_actions       dimension of the action space
n_fc_layers     number of hidden layers
n_neurons       number of neuron per layer
Input:
Dimension                           Scaling (Normalize or Standardize)
env.observation_space.n             StandardScaling ( zero mean & unit
  std) or
MinMaxScaling with [-1, 1] or [0, 1] or
no scaling
Output:
Dimension                           Scaling
np.prod(env.action_space.nvec)      no scaling
External class functions:
model_info      prints information about the DQN object
forward         propagetes a foward pass through the DQN
'''
def __init__(self, n_inputs: int, n_actions: int, n_fc_layers: int,
  n_neurons: int):
super(DQN, self).__init__()

self.layers = nn.ModuleList([nn.Linear(n_inputs, n_neurons)])
for i in range(1, n_fc_layers - 1):
self.layers.append(nn.Linear(n_neurons, n_neurons))
self.layers.append(nn.Linear(n_neurons, n_actions))

def model_info(self):
'''
Description:
Function prints information about the DQN object
'''
return print(self)

def forward(self, x):
'''
Description:
Function propagetes a foward pass through the DQN
'''
y = x
# propagate through network including ReLUs
for i in range(len(self.layers[:-1])):
y = self.layers[i](y)
y = nn.functional.relu(y)
# output w/out Relu as we want to have unbounded Q-values
y = self.layers[-1](y)
```

```
return y
```

Listing 3: Enviroment Modul

```
# import external libraries
import numpy as np
import pandas as pd
import asyncio
import gym
from gym import spaces

class Environment(gym.Env):
'''
Description:
An agent changes the 3D part design through his actions on the geometry'
  s parameters.
First, the environment manipulates part geometry via CAD (NX Siemens).
Second, the environment computes FEM results for a given part geometry.
Third, the environment derives the agents' reward.
Observation:
Type: Box(1) for N states
Num     Observation     Min                 Max
0       state #1        lower bound [0]     upper bound [0]
1       state #2        lower bound [1]     upper bound [1]
N-1     state #N        lower bound [2]     upper bound [2]
Actions:
Type: MultiDiscrete([3] * N)  ->  each state dimension has 3 actions
Num     Action          Semantics
0       NOOP            no change
1       UP              increase state by stepsize
2       DOWN            decrease state by stepsize
Reward:
Reward will be calculated based on a RewardFunction and the respective
  FEM results.
Starting State:
The agent start at Array(initial_value[0], initial_value[1], ... ,
  initial_value[N-1])

  ----------------------------------------------------------------------------------------

Parameters:
timesteps               maximum number of timesteps per episode
reward_fn               function to calculate sub-rewards for each
  requirement
parameter_grid          Pandas DataFrame, which defines initial values,
  lower/upper bounds and stepsizes
fem_results             Pandas DataFrame, which stores all FEM results

'''

def __init__(self, timesteps, reward_fn, parameter_grid: pd.DataFrame,
  fem_results: pd.DataFrame, initial_state=None, best_reward=None):

self._parameter_grid = parameter_grid
self._n_vars = len(self._parameter_grid.index)
self._initial_values = self._parameter_grid.loc[:,'initial_value'].
  to_numpy()
```

```
self._lower_bounds = self._parameter_grid.loc[:,'lower_boundary'].
  to_numpy()
self._upper_bounds = self._parameter_grid.loc[:,'upper_boundary'].
  to_numpy()
self._step_sizes = self._parameter_grid.loc[:,'step_size'].to_numpy()
if initial_state:
self._initial_values = np.array(initial_state, dtype=np.float32) #

self.action_space = spaces.MultiDiscrete([3]*self._n_vars)
self.observation_space = spaces.Box(
low=self._lower_bounds,
high=self._upper_bounds,
shape=(self._n_vars, ),
dtype=np.float32
)

# define necessary states
self._s = None                                          # current state
self._ns = None                                         # next state
self._initial_state = self._initial_values.copy()   # initial state

# member for scaling observations
self._n = 0                                             # counter for
  scaling
self._mean = np.zeros(self._initial_state.shape)        # running mean
self._var = np.zeros(self._initial_state.shape)         # running
  standard deviation

self.done = False                                       # indicator for
  terminal state (e.g. requirements are fulfilled)
self.timesteps = timesteps                              # max timesteps
  for single episode
self._steps = 0

# Reward calculation
self._fem_results = fem_results
self.reward_function = reward_fn

# Additional Metrics - Regret & Visited States
self.regret = 0.0
if best_reward:
self.max_total_reward = best_reward
self.visited_states = []

pass

def reset(self):
'''
This function resets the environment at the beginning of episode.
'''
self.done = False
self._steps = 0
self._s = self._initial_state.copy()
print('\n Environment reset')
return self._scale_observation(self._s)

def _set_CAD_state(self, _state):
'''
```

```
This functions stores the CAD perameters as Excel
'''
_state = pd.DataFrame(data=[_state], columns=self._parameter_grid.index)
_state.to_excel(CAD_INPUT)
return

def _check_boundaries(self, parameter=0, value=0.0):
'''
This function performs a sanity check for each boundary.
'''
check = True
if value < self.observation_space.low[parameter]:
check = False
elif value > self.observation_space.high[parameter]:
check = False
return check

def _lookup_state_in_fem_table(self, state):
'''
This function queries the design logging (i.e., FEM results) for the
  proposed design (i.e., the state).
'''
state = pd.DataFrame(data=[state], columns=self._parameter_grid.index)
cut_me = self._fem_results
for parameter, parameter_value in state.iteritems():
cut_me = cut_me.loc[np.isclose(cut_me[parameter], parameter_value[0])]
if cut_me.empty:
print('Next state is unique')
xs = {}
return True, xs
else:
xs = {}
for requirement in self.reward_function.requirements:
xs.update({requirement: cut_me[requirement].values[0]})
# Calculation 'visited states'
self.visited_states.append(cut_me.index.values[0])
self.visited_states = list(set(self.visited_states))
return False, xs

def _check_done(self, time_step, action):
'''
This function checks if either maximum steps per episode is reached.
'''
check_time_steps = time_step>=self.timesteps
check_requirements = False
return check_time_steps or check_requirements

def _scale_observation(self, obs, mode='minmax'):
'''
This function scales observations between 0 and 1, hence it performes
  data normalization (MinMaxScaling).
'''
if mode == 'standardize':
self._n += 1
old_mean = self._mean
self._mean += (obs - self._mean) / self._n
self._var += ((obs - old_mean) * (obs - self._mean) - self._var) / self.
  _n
```

```
return (obs - self._mean) / np.sqrt(self._var + np.finfo(float).eps)
elif mode == 'minmax':
return (obs - self.observation_space.low) / (self.observation_space.high
    - self.observation_space.low)
elif mode is None:
return obs
else:
ValueError("mode must be either: None, 'minmax' or 'standardize'")

def step(self, action, evaluation=False):
'''
This functions takes a step for given action:
1) parameter modification & sanity (boundary) check
2) query design logging for each requirement of the next state (start
  Cax API to compute CAD/FEM if necessary)
3) reward calculation
4) check if done
Returns:
tuple - (observation, reward, is_done, info)
'''
agent_decision = action
self._ns = np.empty(shape=self._s.shape, dtype=np.float64)

self._steps += 1

# Parameter modification including sanity check
for i, action in enumerate(agent_decision):
step_size = self._step_sizes[i]
# action: NOOP
if action == 0:
self._ns[i] = self._s[i]
# action: UP
elif action == 1:
new_state = self._s[i] + step_size
if self._check_boundaries(parameter=i, value=new_state):
self._ns[i] = self._s[i] + step_size
else:
# Boundary breach is handled via zero-action
self._ns[i] = self._s[i]
# action: DOWN
elif action == 2:
new_state =self._s[i] - step_size
if self._check_boundaries(parameter=i, value=new_state):
self._ns[i] = self._s[i] - step_size
else:
# Boundary breach is handled via zero-action
self._ns[i] = self._s[i]
#print(f'Parameter_{i} has been modified from {self._s[i]} to {self._ns[
  i]} by action: {action}')

# Query of design logging for all requirements
if self.observation_space.contains(self._ns):
unique, xs = self._lookup_state_in_fem_table(self._ns)
if unique:
self._set_CAD_state(self._ns)    # Parameter to Excel
self._run_CAD_API()              # Run CAD API
xs = {}                          # get product characteristics from some
    excelsheet
```

```
else:
raise ValueError("Error inside the parameter modification and sanity
  check.")

# Reward Calculation
sub_rewards = self.reward_function.calc_reward(xs=list(xs.values()),
  evaluation=evaluation)
reward = np.sum(sub_rewards)

# Regret Calculation
self.regret += (self.max_total_reward - reward)

# Check if episode terminates
self.done = self._check_done(time_step=self._steps, action=
  agent_decision)

# update step with next state
self._s = self._ns

# add sub-rewards into dictionary for tensorboard summary
requirement_keys =  self.reward_function.requirements
info = dict(zip(requirement_keys, sub_rewards))

return (self._scale_observation(self._s), reward, self.done, info)
```

Listing 4: Reward Modul

```
import numpy as np

class MinRequirementReward():
def __init__(self, minimum):
self.min = minimum

def calc_reward(self, x):
reward = x - self.min
penalty = 0.0
if reward < 0:
penalty = 2.0 * np.abs(reward)
return reward, penalty

class MaxRequirementReward():
def __init__(self, maximum):
self.max = maximum

def calc_reward(self, x):
reward = self.max - x
penalty = 0.0
if reward < 0:
penalty = 2.0 * np.abs(reward)
return reward, penalty

class RewardFunction():
'''
Description:
This function derives the overall reward from a weighted sum of sub-
  objectives.
```

```
Each sub-objective belongs to a single requirement of the technical
  specification.
Parameters:
requirements    list of strings which specifies the order of the sub-
  objective
fs              list of sub-objective functions
weights         list of weighting factors for each sub-objective
types           list which indicates the function type of each sub-
  objective
targets         list of targets for all requirements
norm            how to normalize the sub-objectives
Methods:
calc_reward     calculates overall reward by weighted sum of sub-
  objectives
normalize       normalization of the sub-objective rewards
'''

def __init__(self, requirements, weights, types, targets, norm=None):
self.fs = []

# generate a list of sub-objectives which encapsulates all requirements
for i, requirement_type in enumerate(types):
if requirement_type=="max":
sub_objective = MaxRequirementReward(targets[i])
self.fs.append(sub_objective)
elif requirement_type=="min":
sub_objective = MinRequirementReward(targets[i])
self.fs.append(sub_objective)
self.types = types
self.weights = weights
self.requirements = requirements

self.norm = norm
self.n = [0] * len(self.fs)
self.init_reward = np.empty((len(self.fs),))
self.init_reward[:] = np.NaN
self.mean_reward = np.zeros((len(self.fs),))
self.max_reward = np.zeros((len(self.fs),))
arr_shape = (len(self.fs), len(self.fs))
self.utopia_nadir = np.ones(shape=arr_shape)
pass

def calc_reward(self, xs, evaluation=False):
'''
Description:
This function will calculate each sub-objective function (including
  penalties) and combine them by means
of weighted sum. Additionally this function will normalize the sub-
  rewards by means of the given `norm`.
'''
total_reward = np.zeros(shape=(len(self.fs),))
reward = np.zeros(shape=(len(self.fs),))
penalty = np.zeros(shape=(len(self.fs),))

# calculate reward and penalty for each sub-objective
for i, sub_objective in enumerate(self.fs):
value = xs[i]
reward[i], penalty[i] = sub_objective.calc_reward(value)
```

```
if self.norm=='pareto':
self._update_pareto(reward, evaluation)

# compute normalized, weighted and penalized sub-rewards
for i in range(len(self.fs)):
rew = self._normalize(reward[i], i)
pen = self._normalize(penalty[i], i, is_penalty=True)
total_reward[i] = self.weights[i] * (rew - pen)
return total_reward

def _update_pareto(self, rewards, evaluation=False):
'''
This functions updates Nadir and Utopia
'''
for i in range(len(self.fs)):
# check for first values ever
if self.n[i] < 1:
for j in range(len(self.fs)):
self.utopia_nadir[i, j] = rewards[j]
if not evaluation:
self.n[i]+=1

# check if new pareto optimality for the respective sub-objective,
  subsequently update row of `utopia_nadir`-Array
if self.utopia_nadir[i, i] < rewards[i]:
print(f'Step {self.n[i]} Reward vorher: {self.utopia_nadir[i, i]} zu: {
  rewards[i]}')
for j in range(len(self.fs)):
self.utopia_nadir[i, j] = rewards[j]
return

def _normalize(self, x, i, is_penalty=False):
'''
This functions performes the normalization
'''
# no normalization
if self.norm is None:
return x

# normalization by means of the 1st computed reward
elif self.norm == 'initial':
return self._normalize_initial(x, i, is_penalty)

# normalization by means of mean reward
elif self.norm == 'mean':
return self._normalize_mean(x, i, is_penalty)

# normalization by means of max reward
elif self.norm == 'max':
return self._normalize_max(x, i, is_penalty)

# normalization by means of pareto optimality difference between 'utopia
  ' and 'nadir'
elif self.norm == 'pareto':
return self._normalize_pareto(x, i, is_penalty)

else:
```

```
raise ValueError('`norm`-Parameter should be one of {None, "initial", "
  mean", "max" or "pareto"}.')

def _normalize_initial(self, x , i, is_penalty=False):
if np.isnan(self.init_reward[i]):
self.init_reward[i] = x
return (1 / np.abs(self.init_reward[i])) * x

def _normalize_mean(self, x , i, is_penalty=False):
'''
This functions computes running mean
'''
if not is_penalty:
self.n[i] += 1
self.mean_reward[i] += (x - self.mean_reward[i]) / self.n[i]
return (1 / np.abs(self.mean_reward[i])) * x

def _normalize_max(self, x , i, is_penalty=False):
'''
This functions computes running (absolute) max
'''
if not is_penalty:
self.max_reward[i] = np.abs(np.array([self.max_reward[i], x])).max()
return (1 / self.max_reward[i]) * x

def _normalize_pareto(self, x , i, is_penalty=False):
'''
This functions computes running pareto
'''
if self.n[i] > 1000:
utopia = self.utopia_nadir[i, i]
nadir = self.utopia_nadir[:, i].min()
return (1 / np.abs(utopia - nadir)) * x
else:
return self._normalize_max(x , i, is_penalty)
```

Listing 5: ReplayBuffer Modul in Anlehnung an [228]

```
'''
This is taken from: https://github.com/PacktPublishing/Deep-
  Reinforcement-Learning-Hands-On-Second-Edition
'''

import numpy as np
import collections

class ExperienceBuffer:
def __init__(self, capacity):
self.buffer = collections.deque(maxlen=capacity)

def __len__(self):
return len(self.buffer)

def append(self, experience):
self.buffer.append(experience)

def sample_minibatch(self, batch_size):
indices = np.random.choice(len(self.buffer), batch_size, replace=False)
```

```
states, actions, rewards, dones, next_states = \
zip(*[self.buffer[idx] for idx in indices])

return np.array(states), np.array(actions), np.array(rewards), \
np.array(dones), np.array(next_states)
```

Appendix: Zusätzliche Auswertungen der Anwendungsbeispiele

Die 20 Merkmalskombinationen mit dem höchsten Total Reward

Tabelle 14: Die 20 Merkmalskombinationen mit dem höchsten Total Reward für den Kurbelarm

ID	crank width	slot length	slot radius	mass [kg]	stress [kN mm^{-2}]	reward mass	reward stress	overall reward
1734	21.0	130.0	12.0	0.181140	0.162231	0.092753	0.084361	0.177114
2006	22.0	125.0	12.0	0.193096	0.150676	0.033955	0.139221	0.173176
1139	19.0	125.0	12.0	0.173358	0.171174	0.131027	0.041903	0.172930
1088	19.0	110.0	12.0	0.187470	0.160556	0.061625	0.092313	0.153938
1360	20.0	105.0	12.0	0.200092	0.148028	−0.001355	0.151792	0.150437
2295	23.0	125.0	12.0	0.199678	0.150516	0.001583	0.139979	0.141562
1733	21.0	130.0	11.5	0.188619	0.162544	0.055971	0.082874	0.138845
1377	20.0	110.0	12.0	0.195052	0.156349	0.024335	0.112284	0.136619
1732	21.0	130.0	11.0	0.196027	0.156245	0.019539	0.112780	0.132320
2023	22.0	130.0	12.0	0.187384	0.165843	0.062046	0.067211	0.129257
1683	21.0	115.0	12.0	0.197268	0.156098	0.013436	0.113476	0.126912
1071	19.0	105.0	12.0	0.192174	0.161683	0.038490	0.086960	0.125451
1138	19.0	125.0	11.5	0.179705	0.175684	0.099808	0.020493	0.120301
1411	20.0	120.0	12.0	0.184972	0.170321	0.073908	0.045950	0.119858
2601	24.0	130.0	12.0	0.199867	0.155138	0.000654	0.118034	0.118689
1699	21.0	120.0	11.5	0.198922	0.157831	0.005301	0.105252	0.110553
1427	20.0	125.0	11.5	0.186734	0.171518	0.065244	0.040267	0.105511
1715	21.0	125.0	11.0	0.200954	0.155076	−0.014073	0.118332	0.104259
1137	19.0	125.0	11.0	0.185993	0.172727	0.068886	0.034528	0.103414
1054	19.0	100.0	12.0	0.196878	0.162026	0.015356	0.085336	0.100692

Tabelle 15: Die 20 Merkmalskombinationen mit dem höchsten Total Reward für die Umlenkwippe

ID	P4	P5	P6	P7	P8	stress [MPa]	deformation [mm]	mass [kg]	reward mass	reward stress	reward deformation	overall reward
2029	6	13	8	55	85	158.397	0.425875	0.106350	0.022226	0.013404	0.044353	0.079984
8845	6	12	8	55	85	159.743	0.431103	0.106329	0.022356	0.010672	0.041225	0.074254
8380	6	11	8	55	85	160.667	0.436433	0.106307	0.022487	0.008795	0.038036	0.069317
7922	6	10	8	55	85	162.840	0.441766	0.106286	0.022617	0.004384	0.034845	0.061845
9282	6	13	8	56	85	160.754	0.447804	0.106718	0.019985	0.008619	0.031232	0.059836
9278	6	13	8	55	84	162.113	0.437936	0.107480	0.015343	0.005860	0.037137	0.058340
7471	5	13	8	55	85	147.650	0.403343	0.111997	−0.036491	0.035223	0.057836	0.056568
2036	6	12	8	56	85	162.534	0.453236	0.106697	0.020115	0.005005	0.027981	0.053101
8844	6	12	8	55	84	163.664	0.443376	0.107459	0.015474	0.002710	0.033882	0.052066
7035	5	12	8	55	85	148.925	0.407910	0.111976	−0.036100	0.032634	0.055103	0.051637
6590	5	11	8	55	85	150.113	0.412601	0.111955	−0.035709	0.030222	0.052296	0.046809
8386	6	11	8	56	85	164.209	0.458975	0.106675	0.020245	0.001605	0.024547	0.046398
2028	6	11	8	55	84	165.325	0.449031	0.107438	0.015604	−0.001984	0.030498	0.044118
1434	5	10	8	55	85	151.047	0.417043	0.111933	−0.035318	0.028326	0.049638	0.042646
9357	6	13	9	55	84	135.156	0.346665	0.116050	−0.110522	0.060589	0.091749	0.041816
9288	6	13	8	57	85	163.529	0.470702	0.107076	0.017805	0.002985	0.017530	0.038321
7928	6	10	8	56	85	165.553	0.464678	0.106654	0.020376	−0.003368	0.021135	0.038142
8937	6	12	9	55	84	136.185	0.350495	0.116028	−0.110131	0.058498	0.089457	0.037825
2035	6	13	8	56	84	165.120	0.460503	0.107852	0.013080	−0.000735	0.023633	0.035979
7921	6	10	8	55	84	166.251	0.454543	0.107416	0.015735	−0.007620	0.027200	0.035314

Tabelle 16: Die 20 Merkmalskombinationen mit dem höchsten Total Reward für den Bremshebel:

ID	P2	P3	P4	P5	Verschiebung [mm]	Masse [g]	Reward Mass	Reward Stress	Total Reward
1642	12	3.0	8.0	10	0.852471	0.307233	0.006074	0.007129	0.013203
1659	12	3.0	8.5	10	0.859892	0.305897	0.000087	0.010572	0.010658
1770	12	3.5	8.0	10	0.863655	0.303169	-0.008847	0.017601	0.008754
1521	12	2.5	8.0	10	0.843818	0.311122	0.013054	-0.008671	0.004383
1531	12	2.5	8.5	10	0.856143	0.310023	0.003111	-0.000174	0.002937
1786	12	3.5	8.5	10	0.867490	0.301835	-0.018126	0.021036	0.002910
1915	12	4.0	8.0	10	0.878469	0.295815	-0.044697	0.036547	-0.008150
1631	12	3.0	7.5	10	0.866575	0.308583	-0.015912	0.003651	-0.012261
1924	12	4.0	8.5	10	0.881880	0.294772	-0.052951	0.039237	-0.013715
1514	12	2.5	7.5	10	0.857139	0.312415	0.002308	-0.018667	-0.016359
2024	12	4.5	8.5	10	0.892771	0.289787	-0.079311	0.052080	-0.027230
1760	12	3.5	7.5	10	0.878690	0.304056	-0.045232	0.015314	-0.029918
2695	13	4.5	8.5	10	0.819524	0.318109	0.032652	-0.062682	-0.030030
2687	13	4.5	8.0	10	0.819606	0.318883	0.032587	-0.068661	-0.036074
1504	12	2.5	7.0	10	0.871486	0.312446	-0.027797	-0.018911	-0.046708
1905	12	4.0	7.5	10	0.895004	0.295877	-0.084714	0.036388	-0.048326
1622	12	3.0	7.0	10	0.881708	0.308605	-0.052535	0.003593	-0.048942
2674	13	4.5	7.5	10	0.835355	0.318943	0.019881	-0.069130	-0.049249
2014	12	4.5	8.0	10	0.900992	0.290723	-0.099206	0.049668	-0.049538
2694	13	4.5	8.5	9	0.850700	0.318109	0.007503	-0.062682	-0.055179

Literaturverzeichnis

[1] Dispan, Jürgen und Schwarz-Kocher, Martin. *Digitalisierung im Maschinenbau. Entwicklungstrends, Herausforderungen, Beschäftigungswirkungen, Gestaltungsfelder im Maschinen- und Anlagenbau*. ger. Working Paper Forschungsförderung 94. Düsseldorf, 2018. http://hdl.handle.net/10419/215889.

[2] Dispan, Jürgen. „Digitale Transformation im Maschinen-und Anlagenbau. Digitalisierungsstrategien und Gestaltung von Arbeit 4.0". *Digitalisierung souverän gestalten*. Springer Vieweg, Berlin, Heidelberg, 2021, S. 118–132.

[3] Verhagen, Wim JC; Bermell-Garcia, Pablo; Van Dijk, Reinier EC und Curran, Richard. „A critical review of Knowledge-Based Engineering: An identification of research challenges". *Advanced Engineering Informatics* 26.1 (2012), S. 5–15.

[4] Arnold, Daniel; Arntz, Melanie; Gregory, Terry; Steffes, Susanne und Zierahn, Ulrich. „Herausforderungen der Digitalisierung für die Zukunft der Arbeitswelt". *ZEW policy brief* 2016 (2016).

[5] Arntz, Melanie; Gregory, Terry; Zierahn, Ulrich; Lehmer, Florian und Matthes, Britta. *Digitalisierung und die zukunft der arbeit: Makroökonomische auswirkungen auf beschäftigung, arbeitslosigkeit und löhne von morgen*. Techn. Ber. ZEW-Gutachten und Forschungsberichte, 2018.

[6] McAfee, Andrew und Brynjolfsson, Erik. *Machine, platform, crowd: Harnessing our digital future*. WW New York: Norton & Company, 2017.

[7] Arntz, Melanie; Gregory, Terry und Zierahn, Ulrich. „Digitalisierung und die Zukunft der Arbeit". *Wirtschaftsdienst* 100.Konferenzheft (2020), S. 41–47.

[8] Bellmann, Lutz; Bourgeon, Pauline; Gathmann, Christina; Kagerl, Christian; Marguerit, David; Martin, Ludivine; Pohlan, Laura und Roth, Duncan. „Digitalisierungsschub in Firmen während der Corona-Pandemie". *Wirtschaftsdienst* 101.9 (2021), S. 713–718.

[9] Ehrlenspiel, Klaus und Meerkamm, Harald. *Integrierte produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Carl Hanser Verlag GmbH Co KG, 2013.

[10] Pahl, Gerhard; Beitz, Wolfgang; Schulz, Hans-Joachim und Jarecki, U. *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*. Springer-Verlag, 2013.

[11] Wartzack, Sandro. *Predictive Engineering-Assistenzsystem zur multikriteriellen Analyse alternativer Produktkonzepte*. VDI-Verlag, 2001.

[12] Sauer, Christopher; Breitsprecher, Thilo; Küstner, Christof; Schleich, Benjamin und Wartzack, Sandro. „SLASSY—An Assistance System for Performing Design for Manufacturing in Sheet-Bulk Metal Forming: Architecture and Self-Learning Aspects". *AI* 2 (2021), S. 307–329. https://www.mdpi.com/2673-2688/2/3/19.

[13] Küstner, Christof. „Assistenzsystem zur Unterstützung der datengetriebenen Produktentwicklung". doctoralthesis. FAU University Press, 2020, xii, 219 Seiten.

[14] Breitsprecher, Thilo. *Entwicklung eines selbstlernenden Assistenzsystems zur automatischen Akquisition von konstruktionsrelevantem Fertigungswissen*. VDI Verlag GmbH, 2019.

[15] La Rocca, Gianfranco. „Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design". *Advanced engineering informatics* 26.2 (2012), S. 159–179.

[16] Leary, Stephen J; Bhaskar, Atul und Keane, Andy J. „A knowledge-based approach to response surface modelling in multifidelity optimization". *Journal of Global Optimization* 26.3 (2003), S. 297–319.

[17] Forrester, Alexander IJ; Sóbester, András und Keane, Andy J. „Multi-fidelity optimization via surrogate modelling". *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463.2088 (2007), S. 3251–3269.

[18] Sun, Guangyong; Li, Guangyao; Zhou, Shiwei; Xu, Wei; Yang, Xujing und Li, Qing. „Multi-fidelity optimization for sheet metal forming process". *Structural and Multidisciplinary Optimization* 44.1 (2011), S. 111–124.

[19] Zheng, Jun; Shao, Xinyu; Gao, Liang; Jiang, Ping und Li, Zilong. „A hybrid variable-fidelity global approximation modelling method combining tuned radial basis function base and kriging correction". *Journal of Engineering Design* 24.8 (2013), S. 604–622.

[20] Ulaganathan, Selvakumar; Couckuyt, Ivo; Ferranti, Francesco; Laermans, Eric und Dhaene, Tom. „Performance study of multi-fidelity gradient enhanced kriging". *Structural and Multidisciplinary Optimization* 51.5 (2015), S. 1017–1033.

[21] Cai, Xiwen; Qiu, Haobo; Gao, Liang und Shao, Xinyu. „Metamodeling for high dimensional design problems by multi-fidelity simulations". *Structural and multidisciplinary optimization* 56.1 (2017), S. 151–166.

[22] Bellman, R.E. *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications, 2003.

[23] Theodoridis, Sergios und Koutroumbas, Konstantinos. „Chapter 5 - Feature Selection". *Pattern Recognition (Fourth Edition)*. Hrsg. von Theodoridis, Sergios und Koutroumbas, Konstantinos. Fourth Edition. Boston: Academic Press, 2009, S. 261–322.

[24] Hughes, Gordon F. „On the mean accuracy of statistical pattern recognizers". *IEEE Trans. Inf. Theory* 14 (1968), S. 55–63.

[25] Yu, Yonggyun; Hur, Taeil; Jung, Jaeho und Jang, In Gwun. „Deep learning for determining a near-optimal topological design without any iteration". *Structural and Multidisciplinary Optimization* 59.3 (2019), S. 787–799.

[26] Du, Xianping und Zhu, Feng. „A new data-driven design methodology for mechanical systems with high dimensional design variables". *Advances in Engineering Software* 117 (2018), S. 18–28.

[27] Dekhtiar, Jonathan; Durupt, Alexandre; Bricogne, Matthieu; Eynard, Benoit; Rowson, Harvey und Kiritsis, Dimitris. „Deep learning for big data applications in CAD and PLM–Research review, opportunities and case study". *Computers in Industry* 100 (2018), S. 227–243.

[28] Vajna, Sándor; Weber, Christian; Zeman, Klaus; Hehenberger, Peter; Gerhard, Detlef und Wartzack, Sandro. *CAx für Ingenieure - Eine praxisbezogene Einführung*. 3. Aufl. Berlin, Heidelberg: Springer Vieweg, 2018.

[29] *VDI 2221-1 Entwicklung technischer Produkte und Systeme - Modell der Produktentwicklung*. Norm. Sep. 2003.

[30] Groover, Mikell P. *Fundamentals of modern manufacturing: materials, processes, and systems*. John Wiley & Sons, 2020.

[31] Russell, Stuart und Norvig, Peter. *Künstliche Intelligenz*. Bd. 2. Pearson Studium München, 2012.

[32] Sutton, Richard S und Barto, Andrew G. *Reinforcement learning: An introduction*. Second. MIT press, 2018.

[33] Koller, Rudolf. *Konstruktionslehre für den Maschinenbau: Grundlagen zur Neu-und Weiterentwicklung technischer Produkte mit Beispielen*. Springer-Verlag, 2013.

[34] Weber, Christian. „Produkte und Produktentwicklungsprozesse abbilden mit Hilfe von Merkmalen und Eigenschaften–eine kritische Zwischenbilanz". *DFX 2012: Proceedings of the 23rd Symposium Design For X, Bamberg/Erlangen, Germany 04.-05.10. 2012*. 2012, S. 25–62.

[35] Lutz, Christoph. „Rechnergestütztes Konfigurieren und Auslegen individualisierter Produkte: Rahmenwerk für die Konzeption und Einführung wissensbasierter Assistenzsysteme in die Konstruktion". Diss. 2011.

[36] Wynn, David C und Clarkson, P John. „Process models in design and development". *Research in Engineering Design* 29.2 (2018), S. 161–202.

[37] Chakrabarti, Amaresh und Blessing, Lucienne TM. „Theories and models of design: A summary of findings". *An anthology of theories and models of design*. Springer, 2014, S. 1–45.

[38] Hansen, Friedrich. *Konstruktionssystematik: eine Arbeitsweise für fortschrittliche Konstrukteure: 2 Aufl*. Verlag Technik, 1955.

[39] Koller, Rudolf. *Konstruktionsmethode für den Maschinen-, Geräte-und Apparatebau*. Springer-Verlag, 2013.

[40] Kesselring, F. „Technische Kompositionslehre. Berlin, Göttingen" (1954).

[41] Roth, Karlheinz. *Konstruieren mit Konstruktionskatalogen: Band 1: Konstruktionslehre*. Bd. 1. Springer-Verlag, 2011.

[42] Weber, Christian. „Modelling products and product development based on characteristics and properties". *An Anthology of theories and models of design*. Springer, 2014, S. 327–352.

[43] Wallace, Ken und Blessing, Luciënne. „An english perspective on the German contribution to engineering design". *Professor Dr.-Ing. Eh Dr.-Ing. Wolfgang Beitz zum Gedenken*. Springer, 1998, S. 583–593.

[44] *VDI 2222-1 Konstruktionsmethodik - Methodisches Entwickeln von Lösungsprinzipien*. Norm. Juni 1997.

[45] Wach, Jörg Johannes. „Problemspezifische Hilfsmittel für die integrierte Produktentwicklung". Diss. Technische Universität München, 1993.

[46] Ponn, Josef und Lindemann, Udo. *Konzeptentwicklung und Gestaltung technischer Produkte: systematisch von Anforderungen zu Konzepten und Gestaltlösungen*. Springer-Verlag, 2011.

[47] Dylla, Norbert. „Denk-und Handlungsabläufe beim Konstruieren". Diss. Technische Universität München, 1990.

[48] Steinhilper, Rolf und Rieg, Frank. *Handbuch Konstruktion*. Carl Hanser Verlag GmbH Co KG, 2012.

[49] Löfqvist, Lars Gunnar. „Design processes and novelty in small companies: a multiple case study“. *DS 58-1: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 1, Design Processes, Palo Alto, CA, USA, 24.-27.08. 2009*. 2009.

[50] Wallentowitz, Henning; Freialdenhoven, Arndt und Olschewski, Ingo. „Strategien in der Automobilindustrie: Technologietrends und Marktentwicklungen. 1“. *Wiesbaden: Vieweg & Teubner, ATZ/MTZ-Fachbuch* (2009).

[51] Tidd, Joe und Bessant, John R. *Managing innovation: integrating technological, market and organizational change*. John Wiley & Sons, 2020.

[52] Pich, Michael T; Loch, Christoph H und Meyer, Arnoud De. „On uncertainty, ambiguity, and complexity in project management“. *Management science* 48.8 (2002), S. 1008–1023.

[53] Clarkson, John und Eckert, Claudia. „Design process improvement: a review of current practice“ (2010).

[54] Karniel, Arie und Reich, Yoram. *Managing the dynamics of new product development processes: a new product lifecycle management paradigm*. Springer Science & Business Media, 2011.

[55] Albers, Albert; Braun, Andreas; Heimicke, Jonas und Richter, Thilo. „Der Prozess der Produktentstehung“. *Handbuch Leichtbau: Methoden, Werkstoffe, Fertigung* (2011), S. 1–30.

[56] *VDI 2221-2 Entwicklung technischer Produkte und Systeme - Gestaltung individueller Produktentwicklungsprozesse*. Norm. Nov. 2019.

[57] Rude, S. „Wissensbasiertes Konstruieren, Zugl.: Karlsruhe, Univ., Habil“. *Schr., 1998, Aachen: Shaker Verlag* (1998).

[58] Griffin, Abbie. „The effect of project and process characteristics on product development cycle time“. *Journal of marketing research* 34.1 (1997), S. 24–35.

[59] Hauschildt, Jürgen; Salomo, Sören; Kock, Alexander und Schultz, Carsten. *Innovationsmanagement*. Vahlen, 2016.

[60] Rajapathirana, R.P. Jayani und Hui, Yan. „Relationship between innovation capability, innovation type, and firm performance“. *Journal of Innovation & Knowledge* 3.1 (2018), S. 44–55.

[61] *VDI 2223 Methodisches Entwerfen technischer Produkte*. Norm. Jan. 2004.

[62] Smith, Michael Harrison. *The Natural Advantage of Nations: Business Opportunities, Innovations and Governance in the 21st Century*. Earthscan, 2013.

[63] Spur, Günter und Krause, Frank-Lothar. *Das virtuelle Produkt: Management der CAD-Technik*. Hanser München, 1997.

[64] Matt, Christian; Hess, Thomas und Benlian, Alexander. „Digital transformation strategies". *Business & information systems engineering* 57.5 (2015), S. 339–343.

[65] Rigger, Eugen; Shea, Kristina und Stankovic, Tino. „Task categorisation for identification of design automation opportunities". *Journal of Engineering Design* 29.3 (2018), S. 131–159.

[66] Chakrabarti, Amaresh. *Engineering design synthesis: understanding, approaches and tools*. Springer Science & Business Media, 2002.

[67] Cagan, Jonathan; Campbell, Matthew; Finger, Susan und Tomiyama, Tetsuo. „A Framework for Computational Design Synthesis: Model and Applications". *Journal of Computing and Information Science in Engineering - JCISE* 5 (Sep. 2005).

[68] Vagia, Marialena; Transeth, Aksel A. und Fjerdingen, Sigurd A. „A literature review on the levels of automation during the years. What are the different taxonomies that have been proposed?" *Applied Ergonomics* 53 (2016), S. 190–202.

[69] Sheridan, Thomas B und Verplank, William L. *Human and computer control of undersea teleoperators*. Techn. Ber. Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab, 1978.

[70] Sheridan, Thomas B. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[71] Endsley, Mica R. „The Application of Human Factors to the Development of Expert Systems for Advanced Cockpits". *Proceedings of the Human Factors Society Annual Meeting* 31.12 (1987), S. 1388–1392.

[72] Ntuen, Celestine A und Park, Eui H. „Human factor issues in teleoperated systems". *Proceedings of the First International Conference on Ergonomics of Hybrid Automated Systems I*. 1988, S. 203–210.

[73] Ntuen, Celestine A und Park, Eui H. *Human interaction with complex systems: conceptual principles and design practice*. Bd. 372. Springer Science & Business Media, 2012.

[74] Riley, Victor. „A General Model of Mixed-Initiative Human-Machine Systems". *Proceedings of the Human Factors Society Annual Meeting* 33.2 (1989), S. 124–128.

[75] Milgram, Paul; Rastogi, Anu und Grodski, Julius J. „Telerobotic control using augmented reality". *Proceedings 4th IEEE International Workshop on Robot and Human Communication*. IEEE. 1995, S. 21–29.

[76] Endsley, Mica R. und Kiris, Esin O. „The Out-of-the-Loop Performance Problem and Level of Control in Automation". *Human Factors* 37.2 (1995), S. 381–394.

[77] Draper, John V. „Teleoperators for advanced manufacturing: Applications and human factors challenges". *International Journal of Human Factors in Manufacturing* 5.1 (1995), S. 53–85.

[78] Endsley, Mica R und Kaber, David B. „Level of automation effects on performance, situation awareness and workload in a dynamic control task". *Ergonomics* 42.3 (1999), S. 462–492.

[79] Lorenz, B; Nocera, FD; Röttger, S; Di Nocera, F; Rottger, S und Parasuraman, R. *The Effects of Level of Automation on the Out-of-the-Loop Unfamiliarity in a Complex Dynamic Fault-Management Task during Simulated Spaceflight Operations, 45 (2), 44–48*. 2001.

[80] Clough, Bruce T. *Metrics, schmetrics! How the heck do you determine a UAV's autonomy anyway*. Techn. Ber. Air Force Research Lab Wright-Patterson AFB OH, 2002.

[81] Clough, Bruce. „Relating autonomy to a task: can it be done?" *1st UAV Conference*. 2002, S. 3445.

[82] Proud, Ryan W; Hart, Jeremy J und Mrozinski, Richard B. *Methods for determining the level of autonomy to design into a human spaceflight vehicle: a function specific approach*. Techn. Ber. National Aeronautics und Space Administration Houston TX, 2003.

[83] Fereidunian, Afreza; Lucas, Caro; Lesani, Hamid; Lehtonen, Matti und Nordman, Mikael. „Challenges in implementation of human-automation interaction models". *2007 Mediterranean Conference on Control Automation*. 2007, S. 1–6.

[84] Fereidunian, Alireza; Lehtonen, Matti; Lesani, Hamid; Lucas, Caro und Nordman, Mikael. „Adaptive autonomy: Smart cooperative cybernetic systems for more humane automation solutions". *2007 IEEE International Conference on Systems, Man and Cybernetics*. 2007, S. 202–207.

[85] Lossack, Ralf-Stefan. *Wissenschaftstheoretische Grundlagen für die rechnerunterstützte Konstruktion*. Springer-Verlag, 2006.

[86] Avgoustinov, Nikolay. *Modelling in mechanical engineering and mechatronics: towards autonomous intelligent software models*. Springer Science & Business Media, 2007.

[87] Ropohl, Günter. *Allgemeine technologie: eine systemtheorie der technik*. KIT Scientific Publishing, 2009.

[88] Kornwachs, Klaus. *Philosophie für Ingenieure*. Carl Hanser Verlag GmbH Co KG, 2018.

[89] Lindemann, Udo. *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*. Springer, 2006.

[90] *VDI 2249:2003-09: Informationsverarbeitung in der Produktentwicklung CAD-Benutzungsfunktionen*. Norm. Nov. 2019.

[91] Stachowiak, Herbert. *Allgemeine modelltheorie*. Springer, 1973.

[92] Duffy, AHB und Andreasen, MM. „Enhancing the evolution of design science". *Proceedings of ICED*. Bd. 95. 1995, S. 29–35.

[93] Funke, Joachim. *Problemlösendes Denken*. Kohlhammer Verlag, 2003.

[94] Bellman, Richard. „The theory of dynamic programming". *Bulletin of the American Mathematical Society* 60.6 (1954), S. 503–515.

[95] Bermell-Garcia, Pablo; Fan, Ip-Shing; Murton, Adrian u. a. „Towards the semantic interoperability between KBE and PLM systems". *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*. 2007, S. 563–564.

[96] Curran, Richard; Verhagen, Wim JC; Van Tooren, Michel JL und Laan, Ton H van der. „A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD". *Expert Systems with Applications* 37.11 (2010), S. 7336–7350.

[97] Ng, Kam KH; Chen, Chun-Hsien; Lee, Carman KM; Jiao, Jianxin Roger und Yang, Zhi-Xin. „A systematic literature review on intelligent automation: Aligning concepts from theory, practice, and future perspectives". *Advanced Engineering Informatics* 47 (2021), S. 101246.

[98] Vuletic, Tijana; Duffy, Alex; Hay, Laura; McTeague, Chris; Pidgeon, Laura und Grealy, Madeleine. „The challenges in computer supported conceptual engineering design". *Computers in Industry* 95 (2018), S. 22–37.

[99] Stokes, Melody u. a. *Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications*. Bd. 3. 3. Professional Engineering Publishing London, 2001.

[100] Skarka, Wojciech. „Application of MOKA methodology in generative model creation using CATIA". *Engineering Applications of Artificial Intelligence* 20.5 (2007), S. 677–690.

[101] Stjepandić, Josip; Verhagen, Wim JC; Liese, Harald und Bermell-Garcia, Pablo. „Knowledge-based engineering". *Concurrent Engineering in the 21st Century*. Springer, 2015, S. 255–286.

[102] Pratt, Michael J u. a. „Introduction to ISO 10303—the STEP standard for product data exchange". *Journal of Computing and Information Science in Engineering* 1.1 (2001), S. 102–103.

[103] *3D-Produktmodellierung: Technische und organisatorische Voraussetzungen Verfahren, Werkzeuge und Anwendungen Wirtschaftlicher Einsatz in der Praxis*. Norm. März 2009.

[104] Kim, Kyoung-Yun; Manley, David G und Yang, Hyungjeong. „Ontology-based assembly design and information sharing for collaborative product development". *Computer-Aided Design* 38.12 (2006), S. 1233–1250.

[105] Zhu, Lijuan; Jayaram, Uma und Kim, Okjoon. „Online semantic knowledge management for product design based on product engineering ontologies". *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Bd. 54792. 2011, S. 1175–1188.

[106] Barbau, Raphael; Krima, Sylvere; Rachuri, Sudarsan; Narayanan, Anantha; Fiorentini, Xenia; Foufou, Sebti und Sriram, Ram D. „OntoSTEP: Enriching product model data using ontologies". *Computer-Aided Design* 44.6 (2012), S. 575–590.

[107] Li, Chun Lei; McMahon, Chris; Newnes, Linda u. a. „Supporting multiple engineering viewpoints in computer-aided design using ontology-based annotations." *DS 75-6: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol. 6: Design Information and Knowledge, Seoul, Korea, 19-22.08. 2013*. 2013, S. 249–258.

[108] Bickel, S.; Sauer, C.; Schleich, B. und Wartzack, S. „Comparing CAD part models for geometrical similarity: A concept using machine learning algorithms". *Procedia CIRP* 96 (2021), S. 133–138.

[109] Vasantha, Gokula; Purves, David; Quigley, John; Corney, Jonathan; Sherlock, Andrew und Randika, Geevin. „Common design structures and substitutable feature discovery in CAD databases". *Advanced Engineering Informatics* 48 (2021), S. 101261.

[110] Ma, Lujie; Huang, Zhengdong und Wang, Yanwei. „Automatic discovery of common design structures in CAD models". *Computers & Graphics* 34.5 (2010), S. 545–555.

[111] Hilaga, Masaki; Shinagawa, Yoshihisa; Kohmura, Taku und Kunii, Tosiyasu L. „Topology matching for fully automatic similarity estimation of 3D shapes". *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, S. 203–212.

[112] Biasotti, Silvia; Marini, Simone; Spagnuolo, Michela und Falcidieno, Bianca. „Sub-part correspondence by structural descriptors of 3D shapes". *Computer-Aided Design* 38.9 (2006), S. 1002–1019.

[113] Bespalov, Dmitriy; Regli, William C und Shokoufandeh, Ali. „Reeb graph based shape retrieval for CAD". *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Bd. 36991. 2003, S. 229–238.

[114] Gao, W; Gao, SM; Liu, YS; Bai, Jing und Hu, BK. „Multiresolutional similarity assessment and retrieval of solid models based on DBMS". *Computer-Aided Design* 38.9 (2006), S. 985–1001.

[115] Sundar, Hari; Silver, Deborah; Gagvani, Nikhil und Dickinson, Sven. „Skeleton based shape matching and retrieval". *2003 Shape Modeling International.* IEEE. 2003, S. 130–139.

[116] Mayer, Johannes und Wartzack, Sandro. „A Concept Towards Automated Reconstruction of Topology Optimized Structures Using Medial Axis Skeletons". *Proceedings of the Munich Symposium on Lightweight Design 2020: Tagungsband zum Münchner Leichtbauseminar 2020*. Springer Berlin Heidelberg. 2021, S. 28–35.

[117] Stangl, Thomas; Wartzack, Sandro u. a. „Feature based interpretation and reconstruction of structural topology optimization results". *DS 80-6 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 6: Design Methods and Tools-Part 2 Milan, Italy, 27-30.07. 15*. 2015, S. 235–244.

[118] Bai, Jing; Gao, Shuming; Tang, Weihua; Liu, Yusheng und Guo, Song. „Design reuse oriented partial retrieval of CAD models". *Computer-Aided Design* 42.12 (2010), S. 1069–1084.

[119] Bai, Jing; Luo, Haonan und Qin, Feiwei. „Design pattern modeling and extraction for CAD models". *Advances in Engineering Software* 93 (2016), S. 30–43.

[120] Zhang, Jie; Zuo, Mi; Wang, Pan; Yu, Jian-feng und Li, Yuan. „A method for common design structure discovery in assembly models using information from multiple sources". *Assembly Automation* (2016).

[121] Schmit, Lucien A. „Structural synthesis-its genesis and development". *AIAA Journal* 19.10 (1981), S. 1249–1263.

[122] Upadhyay, Bhavik D.; Sonigra, Sunil S. und Daxini, Sachin D. „Numerical analysis perspective in structural shape optimization: A review post 2000". *Advances in Engineering Software* 155 (2021), S. 102992.

[123] Bendsøe, Martin P und Sigmund, Ole. „Topology optimization". *Optimization of Structural and Mechanical Systems*. World Scientific, 2007, S. 161–194.

[124] Wall, Wolfgang A.; Frenzel, Moritz A. und Cyron, Christian. „Isogeometric structural shape optimization". *Computer Methods in Applied Mechanics and Engineering* 197.33 (2008), S. 2976–2988.

[125] Marler, R Timothy und Arora, Jasbir S. „Survey of multi-objective optimization methods for engineering". *Structural and multidisciplinary optimization* 26.6 (2004), S. 369–395.

[126] Jahn, Johannes u. a. *Vector optimization*. Springer, 2009.

[127] Coello Coello, Carlos A. „Constraint-handling techniques used with evolutionary algorithms". *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. 2016, S. 563–587.

[128] Greco, Salvatore; Figueira, Jose und Ehrgott, Matthias. *Multiple criteria decision analysis*. Bd. 37. Springer, 2016.

[129] Imam, M Hasan. „Three-dimensional shape optimization". *International Journal for Numerical Methods in Engineering* 18.5 (1982), S. 661–673.

[130] Belegundu, A.D. und Rajan, S.D. „A shape optimization approach based on natural design variables and shape functions". *Computer Methods in Applied Mechanics and Engineering* 66.1 (1988), S. 87–106.

[131] Tortorelli, Daniel A. „A Geometric Representation Scheme Suitable for Shape Optimization". *Mechanics of Structures and Machines* 21.1 (1993), S. 95–121.

[132] Meske, R; Sauter, J und Schnack, E. „Nonparametric gradient-less shape optimization for real-world applications". *Structural and Multidisciplinary Optimization* 30.3 (2005), S. 201–218.

[133] Haftka, Raphael T. und Grandhi, Ramana V. „Structural shape optimization—A survey". *Computer Methods in Applied Mechanics and Engineering* 57.1 (1986), S. 91–106.

[134] Hughes, T.J.R.; Cottrell, J.A. und Bazilevs, Y. „Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement". *Computer Methods in Applied Mechanics and Engineering* 194.39 (2005), S. 4135–4195.

[135] Ding, Yunliang. „Shape optimization of structures: a literature survey". *Computers & Structures* 24.6 (1986), S. 985–1004.

[136] Nguyen, Son H. und Kim, Hyun-Gyu. „Stress-constrained shape and topology optimization with the level set method using trimmed hexahedral meshes". *Computer Methods in Applied Mechanics and Engineering* 366 (2020), S. 113061.

[137] Santhosh, R; Ismail, Shaik; Jain, PC u. a. „Shape optimization of shallow domes subjected to external pressure". *Structural and Multidisciplinary Optimization* 57.2 (2018), S. 903–908.

[138] Zhu, Shaojun; Ohsaki, Makoto; Guo, Xiaonong und Zeng, Qiang. „Shape optimization for non-linear buckling load of aluminum alloy reticulated shells with gusset joints". *Thin-Walled Structures* 154 (2020), S. 106830.

[139] Garcia-Andres, X; Gutierrez-Gil, J; Martinez-Casas, J und Denia, FD. „Wheel shape optimization approaches to reduce railway rolling noise". *Structural and Multidisciplinary Optimization* 62.5 (2020), S. 2555–2570.

[140] Sun, Qiang; Sun, Yuehai und Li, Lun. „Strength analysis and tooth shape optimization for involute gear with a few teeth". *Advances in Mechanical Engineering* 10 (Jan. 2018), S. 11.

[141] Brujic, Djordje; Ristic, Mihailo; Mattone, Massimiliano; Maggiore, Paolo und De Poli, Gian Paolo. „CAD based shape optimization for gas turbine component design". *Structural and Multidisciplinary Optimization* 41.4 (2010), S. 647–659.

[142] Dong, Yifeng; Yao, Xuefeng und Xu, Xiaoyao. „Cross section shape optimization design of fabric rubber seal". *Composite Structures* 256 (2021), S. 113047.

[143] Sonmez, Fazil O. „Optimal shape design of shoulder fillets for flat and round bars under various loadings". *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223 (Aug. 2009), S. 1741–1754.

[144] Sonmez, Fazil O. „Shape optimization of 2D structures using simulated annealing". *Computer Methods in Applied Mechanics and Engineering* 196.35 (2007), S. 3279–3299.

[145] Zhenpei Wang, Yingjun Wang adn; Xia, Zhaohui und Poh, Leong Hien. „Structural Design Optimization Using Isogeometric Analysis: A Comprehensive Review". *Computer Modeling in Engineering & Sciences* 117.3 (2018), S. 455–507.

[146] Eppler, Martin J.; Seifried, Patrick M. und Röpnack, Axel. „Improving knowledge intensive processes through an enterprise knowledge medium". *SIGCPR '99*. 1999.

[147] Roth, D; Binz, H und Watty, R. „Generic structure of knowledge within the product development process". *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia*. 2010.

[148] Luft, Thomas; Roth, Daniel; Binz, Hansgeorg; Wartzack, Sandro u. a. „A new"knowledge-based engineering"guideline". *DS 87-6 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 6: Design Information and Knowledge, Vancouver, Canada, 21-25.08. 2017*. 2017, S. 207–216.

[149] Probst, Gilbert; Raub, Steffen und Romhardt, Kai. *Wissen managen*. Springer, 1997.

[150] Nonaka, Ikujirō; Nonaka, Ikujir o; Ikujiro, Nonaka; Takeuchi, Hirotaka u. a. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Bd. 105. OUP USA, 1995.

[151] North, Klaus. *Wissensorientierte Unternehmensführung: Wissensmanagement gestalten*. Springer-Verlag, 2016.

[152] *Wissensmanagement im Ingenieurwesen: Grundlagen, Konzepte, Vorgehen.* Norm. März 2009.

[153] *Wissensmanagement im Ingenieurwesen: Wissensbasierte Konstruktion (KBE)*. Norm. Mai 2017.

[154] Liese, Harald. *Wissensbasierte 3D-CAD-Repräsentation*. Shaker, 2004.

[155] Hartmann, Dietrich und Lehner, Karlheinz. *Technische Expertensysteme: Grundlagen, Programmiersprachen, Anwendungen*. Springer-Verlag, 2013.

[156] Puppe, Frank. *Einführung in Expertensysteme*. Springer-Verlag, 2013.

[157] Tüchsen, Johann. „Konzeption, Entwicklung und Einführung des Assistenzsystems D-DAS für die Produktentwicklung elektrischer Motoren". Diss. FAU University Press, 2021.

[158] Styczynski, Zbigniew A; Rudion, Krzysztof und Naumann, André. *Einführung in Expertensysteme: Grundlagen, Anwendungen und Beispiele aus der elektrischen Energieversorgung*. Springer-Verlag, 2017.

[159] Rumpe, Bernhard. „Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring". *Springer* 2 (2011), S. 978–3540209058.

[160] Weilkiens, Tim. *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur. Mit einem Geleitwort von Richard Mark Soley*. dpunkt. verlag, 2014.

[161] Dengel, Andreas. *Semantische Technologien: Grundlagen. Konzepte. Anwendungen.* Springer-Verlag, 2011.

[162] Gruber, Thomas R. „Toward principles for the design of ontologies used for knowledge sharing?“ *International journal of human-computer studies* 43.5-6 (1995), S. 907–928.

[163] Hitzler, Pascal; Krötzsch, Markus; Parsia, Bijan; Patel-Schneider, Peter F; Rudolph, Sebastian u. a. „OWL 2 web ontology language primer“. *W3C recommendation* 27.1 (2009), S. 123.

[164] Spruegel, M Sc Tobias C und Wartzack, Ing Sandro. „Das FEA-Assistenzsystem–Analyseteil FEdelM“ (2016).

[165] Sainter, Phillip; Oldham, Keith und Larkin, Andrew. „Achieving benefits from knowledge-based engineering systems in the longer term as well as in the short term“. *Proceedings of: 6th International Conference on Concurrent Enterprising*. Citeseer. 2000.

[166] Baxter, David; Gao, James; Case, Keith; Harding, Jenny; Young, Bob; Cochrane, Sean und Dani, Shilpa. „An engineering design knowledge reuse methodology using process modelling“. *Research in engineering design* 18.1 (2007), S. 37–48.

[167] Bermell-Garcia, Pablo. „A metamodel to annotate knowledge based engineering codes as enterprise knowledge resources“ (2007).

[168] Chandrasegaran, Senthil K; Ramani, Karthik; Sriram, Ram D; Horváth, Imré; Bernard, Alain; Harik, Ramy F und Gao, Wei. „The evolution, challenges, and future of knowledge representation in product design systems“. *Computer-aided design* 45.2 (2013), S. 204–228.

[169] Meerkamm, H und Finkenwirth, K. „«Konstruktionssystem Fertigungsrecht»—ein Expertensystem für den Konstrukteur?“ *VDI-Berichte* 775 (1989), S. 99–114.

[170] Storath, Elmar. „Kontextsensitive Wissensbereitstellung in der Konstruktion“. Diss. Erlangen: Diss. Friedrich-Alexander-Universität Erlangen, 1996.

[171] Weber, Johannes Siegfried. *Konzept eines rechnerunterstützten Assistenzsystems für die Entwicklung umweltgerechter Produkte*. vl. nákl., 1997.

[172] Schön, Achim. „Konzept und Architektur eines Assistenzsystems für die mechatronische Produktentwicklung“. Diss. Technische Fakultät der Universität Erlangen-Nürnberg., 2000.

[173] Hochmuth, Rüdiger. *Methoden und Werkzeuge als Teil eines Assistenzsystems zur rechnergestützten Analyse und Optimierung robuster Produkte*. VDI-Verlag, 2002.

[174] Düchting, Carsten. „Aufbau eines freigabe-und kommunikationsbasierten Assistenzsystems im Produktentstehungsprozess“. Diss. Universität Dortmund, 2005.

[175] Fathi, Madjid; Abramovici, Michael; Holland, Alexander; Lindner, Andreas und Dienst, Susanne. „Nutzungs-Szenarien eines wissensbasierten Assistenzsystems zur Entscheidungsunterstützung in der Produktverbesserung“. *6th Conference on Professional Knowledge Management–From Knowledge to Action*. Gesellschaft für Informatik eV. 2011.

[176] Mayer, Gottfried; Spieckermann, Sven und Wenzel, Sigrid. „Steigerung der Produktivität in Simulationsstudien mit Assistenzwerkzeugen“. *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 107.3 (2012), S. 174–177.

[177] Kestel, P; Schneyer, T; Wartzack, S u. a. „Feature-based approach for the automated setup of accurate, design-accompanying Finite Element Analyses“. *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference*. 2016, S. 697–706.

[178] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[179] Witten, Ian H.; Frank, Eibe und Hall, Mark A. *Data Mining: Practical Machine Learning Tools and Techniques*. 4th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.

[180] Siebertz, Karl; Hochkirchen, Thomas und Bebber, D van. *Statistische versuchsplanung*. Springer, 2010.

[181] Wang, G. Gary und Shan, S. „Review of Metamodeling Techniques in Support of Engineering Design Optimization“. *Journal of Mechanical Design* 129.4 (Mai 2006), S. 370–380.

[182] Papadrakakis, Manolis; Lagaros, Nikos D und Tsompanakis, Yiannis. „Structural optimization using evolution strategies and neural networks“. *Computer methods in applied mechanics and engineering* 156.1-4 (1998), S. 309–333.

[183] Simpson, Timothy W; Mauery, Timothy M; Korte, John J und Mistree, Farrokh. „Kriging models for global approximation in simulation-based multidisciplinary design optimization". *AIAA journal* 39.12 (2001), S. 2233–2241.

[184] Wang, Liping; Beeson, Don; Akkaram, Srikanth und Wiggs, Gene. „Gaussian process meta-models for efficient probabilistic design in complex engineering design spaces". *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Bd. 4739. 2005, S. 785–798.

[185] Simpson, Timothy W; Poplinski, JD; Koch, Patrick N und Allen, Janet K. „Metamodels for computer-based engineering design: survey and recommendations". *Engineering with computers* 17.2 (2001), S. 129–150.

[186] Chiarello, Filippo; Belingheri, Paola und Fantoni, Gualtiero. „Data science for engineering design: State of the art and future directions". *Computers in Industry* 129 (2021), S. 103447.

[187] Pan, Feng; Zhu, Ping und Zhang, Yu. „Metamodel-based lightweight design of B-pillar with TWB structure via support vector regression". *Computers & Structures* 88.1 (2010), S. 36–44.

[188] Duan, Libin; Jiang, Haobin; Geng, Guoqing; Zhang, Xuerong und Li, Zhanjiang. „Parametric modeling and multiobjective crashworthiness design optimization of a new front longitudinal beam". *Structural and Multidisciplinary Optimization* 59.5 (2019), S. 1789–1812.

[189] Zhu, Ping; Pan, Feng; Chen, Wei und Zhang, Siliang. „Use of support vector regression in structural optimization: Application to vehicle crashworthiness design". *Mathematics and Computers in Simulation* 86 (2012), S. 21–31.

[190] Huang, Zhangjun; Wang, Chengen; Chen, Jian und Tian, Hong. „Optimal design of aeroengine turbine disc based on kriging surrogate models". *Computers & Structures* 89.1 (2011), S. 27–37.

[191] Palar, Pramudita Satria und Shimoyama, Koji. „Efficient global optimization with ensemble and selection of kernel functions for engineering design". *Structural and Multidisciplinary Optimization* 59.1 (2019), S. 93–116.

[192] Nagendra, S; Staubach, JB; Suydam, AJ; Ghunakikar, SJ und Akula, VR. „Optimal rapid multidisciplinary response networks: RAPIDDISK". *Structural and Multidisciplinary Optimization* 29.3 (2005), S. 213–231.

[193] Lee, Jongsoo; Jeong, Heeseok und Kang, Seongkyu. „Derivative and GA-based methods in metamodeling of back-propagation neural networks for constrained approximate optimization". *Structural and Multidisciplinary Optimization* 35.1 (2008), S. 29–40.

[194] Wu, Jiajun; Zhang, Chengkai; Xue, Tianfan; Freeman, William T und Tenenbaum, Joshua B. „Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, S. 82–90.

[195] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron und Bengio, Yoshua. „Generative adversarial nets". *Advances in neural information processing systems* 27 (2014).

[196] Zhao, Dong und Xue, Deyi. „A comparative study of metamodeling methods considering sample quality merits". *Structural and Multidisciplinary Optimization* 42.6 (2010), S. 923–938.

[197] Trunk, G. V. „A Problem of Dimensionality: A Simple Example". *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.3 (1979), S. 306–307.

[198] Burggräf, Peter; Wagner, Johannes und Weißer, Tim. „Knowledge-based problem solving in physical product development—-A methodological review". *Expert Systems with Applications: X* 5 (2020), S. 1–14.

[199] Watkins, Christopher JCH und Dayan, Peter. „Q-learning". *Machine learning* 8.3-4 (1992), S. 279–292.

[200] Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Rusu, Andrei A; Veness, Joel; Bellemare, Marc G; Graves, Alex; Riedmiller, Martin; Fidjeland, Andreas K; Ostrovski, Georg u. a. „Human-level control through deep reinforcement learning". *nature* 518.7540 (2015), S. 529–533.

[201] Goodfellow, Ian; Bengio, Yoshua und Courville, Aaron. *Deep Learning*. MIT Press, 2016.

[202] Rumelhart, David E; Hinton, Geoffrey E und Williams, Ronald J. „Learning representations by back-propagating errors". *nature* 323.6088 (1986), S. 533–536.

[203] Rabault, Jean; Kuchta, Miroslav; Jensen, Atle; Réglade, Ulysse und Cerardi, Nicolas. „Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control". *Journal of fluid mechanics* 865 (2019), S. 281–302.

[204] Rabault, Jean und Kuhnle, Alexander. „Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach". *Physics of Fluids* 31.9 (2019), S. 094105.

[205] Tang, Hongwei; Rabault, Jean; Kuhnle, Alexander; Wang, Yan und Wang, Tongguang. „Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning". *Physics of Fluids* 32.5 (2020), S. 053605.

[206] Viquerat, Jonathan; Rabault, Jean; Kuhnle, Alexander; Ghraieb, Hassan; Larcher, Aurélien und Hachem, Elie. „Direct shape optimization through deep reinforcement learning". *Journal of Computational Physics* 428 (2021), S. 110080.

[207] Garnier, Paul; Viquerat, Jonathan; Rabault, Jean; Larcher, Aurélien; Kuhnle, Alexander und Hachem, Elie. „A review on deep reinforcement learning for fluid mechanics". *Computers & Fluids* 225 (2021), S. 104973.

[208] Yonekura, Kazuo und Hattori, Hitoshi. „Framework for design optimization using deep reinforcement learning". *Structural and Multidisciplinary Optimization* 60.4 (2019), S. 1709–1713.

[209] Yan, Xinghui; Zhu, Jihong; Kuang, Minchi und Wang, Xiangyang. „Aerodynamic shape optimization using a novel optimizer based on machine learning techniques". *Aerospace Science and Technology* 86 (2019), S. 826–835.

[210] Hayashi, Kazuki und Ohsaki, Makoto. „Reinforcement learning and graph embedding for binary truss topology optimization under stress and displacement constraints". *Frontiers in Built Environment* 6 (2020), S. 59.

[211] Hayashi, Kazuki und Ohsaki, Makoto. „Reinforcement learning for optimum design of a plane frame under static loads". *Engineering with Computers* 37 (2020), S. 1–13.

[212] Sun, Hongbo und Ma, Ling. „Generative Design by Using Exploration Approaches of Reinforcement Learning in Density-Based Structural Topology Optimization". *Designs* 4.2 (Mai 2020), S. 10.

[213] Stocker, Cosima; Schmid, Marc und Reinhart, Gunther. „Reinforcement learning–based design of orienting devices for vibratory bowl feeders". *The International Journal of Advanced Manufacturing Technology* 105.9 (2019), S. 3631–3642.

[214] Zirngibl, Christoph; Schleich, Benjamin und Wartzack, Sandro. „Approach for the automated and Data-Based Design of Mechanical Joints". *Proceedings of the Design Society* 1 (2021), S. 521–530.

[215] Donoho, David L u. a. „High-dimensional data analysis: The curses and blessings of dimensionality". *AMS math challenges lecture* 1.2000 (2000), S. 1–32.

[216] Rahman, Molla Hafizur; Xie, Charles und Sha, Zhenghui. „A Deep Learning Based Approach to Predict Sequential Design Decisions". *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Bd. Volume 1: 39th Computers and Information in Engineering Conference. Aug. 2019, S. 12.

[217] Chapman, Pete; Clinton, Julian; Kerber, Randy; Khabaza, Thomas; Reinartz, Thomas; Shearer, Colin; Wirth, Rudiger u. a. „CRISP-DM 1.0: Step-by-step data mining guide". *SPSS inc* 9 (2000), S. 13.

[218] Bergstra, James und Bengio, Yoshua. „Random Search for Hyper-Parameter Optimization". *Journal of Machine Learning Research* 13.10 (2012), S. 281–305.

[219] Bergstra, James; Komer, Brent; Eliasmith, Chris; Yamins, Dan und Cox, David D. „Hyperopt: a Python library for model selection and hyperparameter optimization". *Computational Science & Discovery* 8.1 (Juli 2015), S. 014008. https://doi.org/10.1088/1749-4699/8/1/014008.

[220] Thrun, Sebastian B. „Efficient exploration in reinforcement learning". *Technical Report CMU-CS-92-102* (1992).

[221] Tokic, Michel. „Adaptive ϵ-Greedy Exploration in Reinforcement Learning Based on Value Differences". *KI 2010: Advances in Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, Sep. 2010, S. 203–210.

[222] Hessel, Matteo; Modayil, Joseph; Van Hasselt, Hado; Schaul, Tom; Ostrovski, Georg; Dabney, Will; Horgan, Dan; Piot, Bilal; Azar, Mohammad und Silver, David. „Rainbow: Combining improvements in deep reinforcement learning". *Thirty-second AAAI conference on artificial intelligence*. 2018.

[223] Petrik, Marek und Scherrer, Bruno. „Biasing approximate dynamic programming with a lower discount factor". *Advances in neural information processing systems*. Hrsg. von Koller, D.; Schuurmans, D.; Bengio, Y. und Bottou, L. Bd. 21. Curran Associates, Inc., 2009, S. 1265–1272.

[224] Fogliatto, Flavio S.; da Silveira, Giovani J.C. und Borenstein, Denis. „The mass customization decade: An updated review of the literature". *International Journal of Production Economics* 138.1 (2012), S. 14–25.

[225] Bini, Rodrigo R und Carpes, Felipe P. *Biomechanics of cycling*. Springer, 2014.

[226] *Fahrräder - Sicherheitstechnische Anforderungen an Fahrräder - Teil 2: Anforderungen für City- und Trekkingfahrräder, Jugendfahrräder, Geländefahrräder*. Norm. Dez. 2015.

[227] *Fahrräder - Sicherheitstechnische Anforderungen an Fahrräder - Teil 4: Prüfverfahren für Bremsen*. Norm. Jan. 2015.

[228] Lapan, Maxim. *Deep Reinforcement Learning Hands-On: Apply modern RL methods to practical problems of chatbots, robotics, discrete optimization, web automation, and more*. Packt Publishing Ltd, 2020.

Verzeichnis promotionsbezogener, eigener Publikationen

[P1] Kügler, Patricia; Dworschak, Fabian; Schleich, Benjamin und Wartzack, Sandro. „The evolution of knowledge-based engineering from a design research perspective: Literature review 2012–2021". *Advanced Engineering Informatics* 55 (Jan. 2023).

[P2] Dworschak, Fabian; Zirngibl, Christoph; Schleich, Benjamin und Wartzack, Sandro. „Konzept für den MBSE-Einsatz zur automatisierten Individualisierung von komplexen Produkten". *Beiträge zum 30. DfX-Symposium* (Jesteburg). Hrsg. von Dieter Krause Kristin Paetzold, Sandro Wartzack. Hamburg: TuTech Verlag, –19. Sep. 2019, S. 279–290.

[P3] Dworschak, Fabian; Tüchsen, Johann; Pop, Adrian-Cornel; Pinto, Diogo; Schleich, Benjamin und Wartzack, Sandro. „On Simulation Knowledge Acquisition Using Gaussian Processes for the Design of Electric Motors". *Procedia CIRP* 84 (2019), S. 737–742.

[P4] Dworschak, Fabian; Küstner, Christof; Schleich, Benjamin und Wartzack, Sandro. „Szenario-Analyse zur Digitalisierung und Automatisierung des Produktentwicklungsprozesses". *Tagungsband 16. Gemeinsames Kolloquium Konstruktionstechnik* (Bayreuth). –12. Okt. 2018, S. 61–70.

[P5] Zirngibl, Christoph; Dworschak, Fabian; Schleich, Benjamin und Wartzack, Sandro. „Application of reinforcement learning for the optimization of clinch joint characteristics". *Production Engineering* (2021).

[P6] Dworschak, Fabian; Kügler, Patricia; Schleich, Benjamin und Wartzack, Sandro. „Model and Knowledge Representation for the Reuse of Design Process Knowledge Supporting Design Automation in Mass Customization". *Applied Sciences* (2021).

[P7] Dworschak, Fabian; Kügler, Patricia; Schleich, Benjamin und Wartzack, Sandro. „Integrating The Mechanical Domain Into Seed Approach". *Proceedings of the 22nd International Conference on Engineering Design (ICED19)*. Cambridge, United Kingdom: Cambridge University Press, 2019, S. 2587–2596.

[P8] Dworschak, Fabian; Sauer, Christopher; Schleich, Benjamin und Wartzack, Sandro. „Reinforcement Learning as an Alternative for Parameter Prediction in Design for Sheet Bulk Metal Forming". *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (St. Louis, Missouri). 2022, S. 2587–2596.

[P9] Dworschak, Fabian; Dietze, Sebastian; Wittmann, Maximilian; Schleich, Benjamin und Wartzack, Sandro. „Reinforcement Learning for Engineering Design Automation". *Advanced Engineering Informatics* 52 (2022), S. 101612.

Verzeichnis promotionsbezogener, studentischer Arbeiten

[S1] Dietze, Sebastian. „Reinforcement Learning for Geometric Design Optimization in the Virtual Product Development“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2022.

[S2] Dechert, Dennis. „Anwendung von Reinforcement Learning in der Produktentwicklung“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2019.

[S3] Scholz, Felix. „Entwicklung einer Methode für die semantische Beschreibung von Wirkprinzipien für die automatisierte Konstruktion“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2020.

[S4] Schulte, David. „Konzeptfürdiesemantische Integration von Anforderungen in die automatisierte Anpassungskonstruktion“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2020.

[S5] Wittmann, Maximilian. „Development and Implementation of Reinforcement Learning Algorithms in Engineering Design“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2020.

[S6] Zirngibl, Christoph. „Konzept eines durchgängig digitalisierten Produktmodells für die automatisierte Produktsynthese“. Masterarbeit. Friedrich-Alexander Universität Erlangen-Nürnberg, 2019.

Reihenübersicht

Koordination der Reihe (Stand 2024):
Geschäftsstelle Maschinenbau, Dr.-Ing. Oliver Kreis, www.mb.fau.de/diss/

Im Rahmen der Reihe sind bisher die nachfolgenden Bände erschienen.

Band 1 – 52
Fertigungstechnik – Erlangen
ISSN 1431-6226
Carl Hanser Verlag, München

Band 53 – 307
Fertigungstechnik – Erlangen
ISSN 1431-6226
Meisenbach Verlag, Bamberg

ab Band 308
FAU Studien aus dem Maschinenbau
ISSN 2625-9974
FAU University Press, Erlangen

Die Zugehörigkeit zu den jeweiligen Lehrstühlen ist wie folgt gekennzeichnet:

Lehrstühle:

FAPS	Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik
FMT	Lehrstuhl für Fertigungsmesstechnik
KTmfk	Lehrstuhl für Konstruktionstechnik
LFT	Lehrstuhl für Fertigungstechnologie
LGT	Lehrstuhl für Gießereitechnik
LPT	Lehrstuhl für Photonische Technologien
REP	Lehrstuhl für Ressourcen- und Energieeffiziente Produktionsmaschinen

Band 1: Andreas Hemberger
Innovationspotentiale in der rechnerintegrierten Produktion durch wissensbasierte Systeme
FAPS, 208 Seiten, 107 Bilder. 1988.
ISBN 3-446-15234-2.

Band 2: Detlef Classe
Beitrag zur Steigerung der Flexibilität automatisierter Montagesysteme durch Sensorintegration und erweiterte Steuerungskonzepte
FAPS, 194 Seiten, 70 Bilder. 1988.
ISBN 3-446-15529-5.

Band 3: Friedrich-Wilhelm Nolting
Projektierung von Montagesystemen
FAPS, 201 Seiten, 107 Bilder, 1 Tab. 1989.ISBN 3-446-15541-4.

Band 4: Karsten Schlüter
Nutzungsgradsteigerung von Montagesystemen durch den Einsatz der Simulationstechnik
FAPS, 177 Seiten, 97 Bilder. 1989.
ISBN 3-446-15542-2.

Band 5: Shir-Kuan Lin
Aufbau von Modellen zur Lageregelung von Industrierobotern
FAPS, 168 Seiten, 46 Bilder. 1989.
ISBN 3-446-15546-5.

Band 6: Rudolf Nuss
Untersuchungen zur Bearbeitungsqualität im Fertigungssystem Laserstrahlschneiden
LFT, 206 Seiten, 115 Bilder, 6 Tab. 1989. ISBN 3-446-15783-2.

Band 7: Wolfgang Scholz
Modell zur datenbankgestützten Planung automatisierter Montageanlagen
FAPS, 194 Seiten, 89 Bilder. 1989.
ISBN 3-446-15825-1.

Band 8: Hans-Jürgen Wißmeier
Beitrag zur Beurteilung des Bruchverhaltens von Hartmetall-Fließpreßmatrizen
LFT, 179 Seiten, 99 Bilder, 9 Tab. 1989. ISBN 3-446-15921-5.

Band 9: Rainer Eisele
Konzeption und Wirtschaftlichkeit von Planungssystemen in der Produktion
FAPS, 183 Seiten, 86 Bilder. 1990.
ISBN 3-446-16107-4.

Band 10: Rolf Pfeiffer
Technologisch orientierte Montageplanung am Beispiel der Schraubtechnik
FAPS, 216 Seiten, 102 Bilder, 16 Tab. 1990. ISBN 3-446-16161-9.

Band 11: Herbert Fischer
Verteilte Planungssysteme zur Flexibilitätssteigerung der rechnerintegrierten Teilefertigung
FAPS, 201 Seiten, 82 Bilder. 1990.
ISBN 3-446-16105-8.

Band 12: Gerhard Kleineidam
CAD/CAP: Rechnergestützte Montagefeinplanung
FAPS, 203 Seiten, 107 Bilder. 1990.
ISBN 3-446-16112-0.

Band 13: Frank Vollertsen
Pulvermetallurgische Verarbeitung eines übereutektoiden verschleißfesten Stahls
LFT, XIII u. 217 Seiten, 67 Bilder, 34 Tab. 1990.
ISBN 3-446-16133-3.

Band 14: Stephan Biermann
Untersuchungen zur Anlagen- und Prozeßdiagnostik für das Schneiden mit CO_2-Hochleistungslasern
LFT, VIII u. 170 Seiten, 93 Bilder, 4 Tab. 1991. ISBN 3-446-16269-0.

Band 15: Uwe Geißler
Material- und Datenfluß in einer flexiblen Blechbearbeitungszelle
LFT, 124 Seiten, 41 Bilder, 7 Tab. 1991. ISBN 3-446-16358-1.

Band 16: Frank Oswald Hake
Entwicklung eines rechnergestützten Diagnosesystems für automatisierte Montagezellen
FAPS, XIV u. 166 Seiten, 77 Bilder. 1991. ISBN 3-446-16428-6.

Band 17: Herbert Reichel
Optimierung der Werkzeugbereitstellung durch rechnergestützte Arbeitsfolgenbestimmung
FAPS, 198 Seiten, 73 Bilder, 2 Tab. 1991. ISBN 3-446-16453-7.

Band 18: Josef Scheller
Modellierung und Einsatz von Softwaresystemen für rechnergeführte Montagezellen
FAPS, 198 Seiten, 65 Bilder. 1991.
ISBN 3-446-16454-5.

Band 19: Arnold vom Ende
Untersuchungen zum Biegeumformen mit elastischer Matrize
LFT, 166 Seiten, 55 Bilder, 13 Tab. 1991. ISBN 3-446-16493-6.

Band 20: Joachim Schmid
Beitrag zum automatisierten Bearbeiten von Keramikguß mit Industrierobotern
FAPS, XIV u. 176 Seiten, 111 Bilder, 6 Tab. 1991.
ISBN 3-446-16560-6.

Band 21: Egon Sommer
Multiprozessorsteuerung für kooperierende Industrieroboter in Montagezellen
FAPS, 188 Seiten, 102 Bilder. 1991.
ISBN 3-446-17062-6.

Band 22: Georg Geyer
Entwicklung problemspezifischer Verfahrensketten in der Montage
FAPS, 192 Seiten, 112 Bilder. 1991.
ISBN 3-446-16552-5.

Band 23: Rainer Flohr
Beitrag zur optimalen Verbindungstechnik in der Oberflächenmontage (SMT)
FAPS, 186 Seiten, 79 Bilder. 1991.
ISBN 3-446-16568-1.

Band 24: Alfons Rief
Untersuchungen zur Verfahrensfolge Laserstrahlschneiden und -schweißen in der Rohkarosseriefertigung
LFT, VI u. 145 Seiten, 58 Bilder, 5 Tab. 1991. ISBN 3-446-16593-2.

Band 25: Christoph Thim
Rechnerunterstützte Optimierung von Materialflußstrukturen in der Elektronikmontage durch Simulation
FAPS, 188 Seiten, 74 Bilder. 1992.
ISBN 3-446-17118-5.

Band 26: Roland Müller
CO_2 -Laserstrahlschneiden von kurzglasverstärkten Verbundwerkstoffen
LFT, 141 Seiten, 107 Bilder, 4 Tab. 1992. ISBN 3-446-17104-5.

Band 27: Günther Schäfer
Integrierte Informationsverarbeitung bei der Montageplanung
FAPS, 195 Seiten, 76 Bilder. 1992.
ISBN 3-446-17117-7.

Band 28: Martin Hoffmann
Entwicklung einerCAD/CAM-Prozeßkette für die Herstellung von Blechbiegeteilen
LFT, 149 Seiten, 89 Bilder. 1992. ISBN 3-446-17154-1.

Band 29: Peter Hoffmann
Verfahrensfolge Laserstrahlschneiden und –schweißen: Prozeßführung und Systemtechnik in der 3D-Laserstrahlbearbeitung von Blechformteilen
LFT, 186 Seiten, 92 Bilder, 10 Tab. 1992. ISBN 3-446-17153-3.

Band 30: Olaf Schrödel
Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen
FAPS, 180 Seiten, 84 Bilder. 1992. ISBN 3-446-17242-4.

Band 31: Hubert Reinisch
Planungs- und Steuerungswerkzeuge zur impliziten Gerätepro-grammierung in Roboterzellen
FAPS, XI u. 212 Seiten, 112 Bilder. 1992. ISBN 3-446-17380-3.

Band 32: Brigitte Bärnreuther
Ein Beitrag zur Bewertung des Kommunikationsverhaltens von Automatisierungsgeräten in flexiblen Produktionszellen
FAPS, XI u. 179 Seiten, 71 Bilder. 1992. ISBN 3-446-17451-6.

Band 33: Joachim Hutfless
Laserstrahlregelung und Optikdiagnostik in der Strahlführung einer CO_2-Hochleistungslaseranlage
LFT, 175 Seiten, 70 Bilder, 17 Tab. 1993. ISBN 3-446-17532-6.

Band 34: Uwe Günzel
Entwicklung und Einsatz eines Simula-tionsverfahrens für operative undstrategische Probleme der Produktionsplanung und -steuerung
FAPS, XIV u. 170 Seiten, 66 Bilder, 5 Tab. 1993. ISBN 3-446-17604-7.

Band 35: Bertram Ehmann
Operatives Fertigungscontrolling durch Optimierung auftragsbezogener Bearbeitungsabläufe in der Elektronikfertigung
FAPS, XV u. 167 Seiten, 114 Bilder. 1993. ISBN 3-446-17658-6.

Band 36: Harald Kolléra
Entwicklung eines benutzerorientierten Werkstattprogrammiersystems für das Laserstrahlschneiden
LFT, 129 Seiten, 66 Bilder, 1 Tab. 1993. ISBN 3-446-17719-1.

Band 37: Stephanie Abels
Modellierung und Optimierung von Montageanlagen in einem integrierten Simulationssystem
FAPS, 188 Seiten, 88 Bilder. 1993. ISBN 3-446-17731-0.

Band 38: Robert Schmidt-Hebbel
Laserstrahlbohren durchflußbestimmender Durchgangslöcher
LFT, 145 Seiten, 63 Bilder, 11 Tab. 1993. ISBN 3-446-17778-7.

Band 39: Norbert Lutz
Oberflächenfeinbearbeitung keramischer Werkstoffe mit XeCl-Excimerlaserstrahlung
LFT, 187 Seiten, 98 Bilder, 29 Tab. 1994. ISBN 3-446-17970-4.

Band 40: Konrad Grampp
Rechnerunterstützung bei Test und Schulung an Steuerungssoftware von SMD-Bestücklinien
FAPS, 178 Seiten, 88 Bilder. 1995. ISBN 3-446-18173-3.

Band 41: Martin Koch
Wissensbasierte Unterstützung derAngebotsbearbeitung in der Investitionsgüterindustrie
FAPS, 169 Seiten, 68 Bilder. 1995. ISBN 3-446-18174-1.

Band 42: Armin Gropp
Anlagen- und Prozeßdiagnostik beim Schneiden mit einem gepulsten Nd:YAG-Laser
LFT, 160 Seiten, 88 Bilder, 7 Tab. 1995. ISBN 3-446-18241-1.

Band 43: Werner Heckel
Optische 3D-Konturerfassung und on-line Biegewinkelmessung mit dem Lichtschnittverfahren
LFT, 149 Seiten, 43 Bilder, 11 Tab. 1995. ISBN 3-446-18243-8.

Band 44: Armin Rothhaupt
Modulares Planungssystem zur Optimierung der Elektronikfertigung
FAPS, 180 Seiten, 101 Bilder. 1995. ISBN 3-446-18307-8.

Band 45: Bernd Zöllner
Adaptive Diagnose in der Elektronikproduktion
FAPS, 195 Seiten, 74 Bilder, 3 Tab. 1995. ISBN 3-446-18308-6.

Band 46: Bodo Vormann
Beitrag zur automatisierten Handhabungsplanung komplexer Blechbiegeteile
LFT, 126 Seiten, 89 Bilder, 3 Tab. 1995. ISBN 3-446-18345-0.

Band 47: Peter Schnepf
Zielkostenorientierte Montageplanung
FAPS, 144 Seiten, 75 Bilder. 1995. ISBN 3-446-18397-3.

Band 48: Rainer Klotzbücher
Konzept zur rechnerintegrierten Materialversorgung in flexiblen Fertigungssystemen
FAPS, 156 Seiten, 62 Bilder. 1995. ISBN 3-446-18412-0.

Band 49: Wolfgang Greska
Wissensbasierte Analyse undKlassifizierung von Blechteilen
LFT, 144 Seiten, 96 Bilder. 1995. ISBN 3-446-18462-7.

Band 50: Jörg Franke
Integrierte Entwicklung neuer Produkt- und Produktionstechnologien für räumliche spritzgegossene Schaltungsträger (3-D MID)
FAPS, 196 Seiten, 86 Bilder, 4 Tab. 1995. ISBN 3-446-18448-1.

Band 51: Franz-Josef Zeller
Sensorplanung und schnelle Sensorregelung für Industrieroboter
FAPS, 190 Seiten, 102 Bilder, 9 Tab. 1995. ISBN 3-446-18601-8.

Band 52: Michael Solvie
Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen
FAPS, 200 Seiten, 87 Bilder, 35 Tab. 1996. ISBN 3-446-18607-7.

Band 53: Robert Hopperdietzel
Reengineering in der Elektro- und Elektronikindustrie
FAPS, 180 Seiten, 109 Bilder, 1 Tab. 1996. ISBN 3-87525-070-2.

Band 54: Thomas Rebhahn
Beitrag zur Mikromaterialbearbeitung mit Excimerlasern - Systemkomponenten und Verfahrensoptimierungen
LFT, 148 Seiten, 61 Bilder, 10 Tab. 1996. ISBN 3-87525-075-3.

Band 55: Henning Hanebuth
Laserstrahlhartlöten mit Zweistrahltechnik
LFT, 157 Seiten, 58 Bilder, 11 Tab. 1996. ISBN 3-87525-074-5.

Band 56: Uwe Schönherr
Steuerung und Sensordatenintegration für flexible Fertigungszellen mitkooperierenden Robotern
FAPS, 188 Seiten, 116 Bilder, 3 Tab. 1996. ISBN 3-87525-076-1.

Band 57: Stefan Holzer
Berührungslose Formgebung mit-Laserstrahlung
LFT, 162 Seiten, 69 Bilder, 11 Tab. 1996. ISBN 3-87525-079-6.

Band 58: Markus Schultz
Fertigungsqualität beim 3D-Laserstrahlschweißen vonBlechformteilen
LFT, 165 Seiten, 88 Bilder, 9 Tab. 1997. ISBN 3-87525-080-X.

Band 59: Thomas Krebs
Integration elektromechanischer CA-Anwendungen über einem STEP-Produktmodell
FAPS, 198 Seiten, 58 Bilder, 8 Tab. 1997. ISBN 3-87525-081-8.

Band 60: Jürgen Sturm
Prozeßintegrierte Qualitätssicherung in der Elektronikproduktion
FAPS, 167 Seiten, 112 Bilder, 5 Tab. 1997. ISBN 3-87525-082-6.

Band 61: Andreas Brand
Prozesse und Systeme zur Bestückung räumlicher elektronischer Baugruppen (3D-MID)
FAPS, 182 Seiten, 100 Bilder. 1997. ISBN 3-87525-087-7.

Band 62: Michael Kauf
Regelung der Laserstrahlleistung und der Fokusparameter einer CO_2-Hochleistungslaseranlage
LFT, 140 Seiten, 70 Bilder, 5 Tab. 1997. ISBN 3-87525-083-4.

Band 63: Peter Steinwasser
Modulares Informationsmanagement in der integrierten Produkt- und Prozeßplanung
FAPS, 190 Seiten, 87 Bilder. 1997. ISBN 3-87525-084-2.

Band 64: Georg Liedl
Integriertes Automatisierungskonzept für den flexiblen Materialfluß in der Elektronikproduktion
FAPS, 196 Seiten, 96 Bilder, 3 Tab. 1997. ISBN 3-87525-086-9.

Band 65: Andreas Otto
Transiente Prozesse beim Laserstrahlschweißen
LFT, 132 Seiten, 62 Bilder, 1 Tab. 1997. ISBN 3-87525-089-3.

Band 66: Wolfgang Blöchl
Erweiterte Informationsbereitstellung an offenen CNC-Steuerungen zur Prozeß- und Programmoptimierung
FAPS, 168 Seiten, 96 Bilder. 1997. ISBN 3-87525-091-5.

Band 67: Klaus-Uwe Wolf
Verbesserte Prozeßführung und Prozeßplanung zur Leistungs- und Qualitätssteigerung beim Spulenwickeln
FAPS, 186 Seiten, 125 Bilder. 1997. ISBN 3-87525-092-3.

Band 68: Frank Backes
Technologieorientierte Bahnplanung für die 3D-Laserstrahlbearbeitung
LFT, 138 Seiten, 71 Bilder, 2 Tab. 1997. ISBN 3-87525-093-1.

Band 69: Jürgen Kraus
Laserstrahlumformen von Profilen
LFT, 137 Seiten, 72 Bilder, 8 Tab. 1997. ISBN 3-87525-094-X.

Band 70: Norbert Neubauer
Adaptive Strahlführungen für CO_2-Laseranlagen
LFT, 120 Seiten, 50 Bilder, 3 Tab. 1997. ISBN 3-87525-095-8.

Band 71: Michael Steber
Prozeßoptimierter Betrieb flexibler Schraubstationen in der automatisierten Montage
FAPS, 168 Seiten, 78 Bilder, 3 Tab. 1997. ISBN 3-87525-096-6.

Band 72: Markus Pfestorf
Funktionale 3D-Oberflächenkenngrößen in der Umformtechnik
LFT, 162 Seiten, 84 Bilder, 15 Tab. 1997. ISBN 3-87525-097-4.

Band 73: Volker Franke
Integrierte Planung und Konstruktion von Werkzeugen für die Biegebearbeitung
LFT, 143 Seiten, 81 Bilder. 1998. ISBN 3-87525-098-2.

Band 74: Herbert Scheller
Automatisierte Demontagesysteme und recyclinggerechte Produktgestaltung elektronischer Baugruppen
FAPS, 184 Seiten, 104 Bilder, 17 Tab. 1998. ISBN 3-87525-099-0.

Band 75: Arthur Meßner
Kaltmassivumformung metallischer Kleinstteile – Werkstoffverhalten, Wirkflächenreibung, Prozeßauslegung
LFT, 164 Seiten, 92 Bilder, 14 Tab. 1998. ISBN 3-87525-100-8.

Band 76: Mathias Glasmacher
Prozeß- und Systemtechnik zum Laserstrahl-Mikroschweißen
LFT, 184 Seiten, 104 Bilder, 12 Tab. 1998. ISBN 3-87525-101-6.

Band 77: Michael Schwind
Zerstörungsfreie Ermittlung mechanischer Eigenschaften von Feinblechen mit dem Wirbelstromverfahren
LFT, 124 Seiten, 68 Bilder, 8 Tab. 1998. ISBN 3-87525-102-4.

Band 78: Manfred Gerhard
Qualitätssteigerung in der Elektronikproduktion durch Optimierung der Prozeßführung beim Löten komplexer Baugruppen
FAPS, 179 Seiten, 113 Bilder, 7 Tab. 1998. ISBN 3-87525-103-2.

Band 79: Elke Rauh
Methodische Einbindung der Simulation in die betrieblichen Planungs- und Entscheidungsabläufe
FAPS, 192 Seiten, 114 Bilder, 4 Tab. 1998. ISBN 3-87525-104-0.

Band 80: Sorin Niederkorn
Meßeinrichtung zur Untersuchung der Wirkflächenreibung bei umformtechnischen Prozessen
LFT, 99 Seiten, 46 Bilder, 6 Tab. 1998. ISBN 3-87525-105-9.

Band 81: Stefan Schuberth
Regelung der Fokuslage beim Schweißen mit CO_2-Hochleistungslasern unter Einsatz von adaptiven Optiken
LFT, 140 Seiten, 64 Bilder, 3 Tab. 1998. ISBN 3-87525-106-7.

Band 82: Armando Walter Colombo
Development and Implementation of Hierarchical Control Structures of Flexible Production Systems Using High Level Petri Nets
FAPS, 216 Seiten, 86 Bilder. 1998. ISBN 3-87525-109-1.

Band 83: Otto Meedt
Effizienzsteigerung bei Demontage und Recycling durch flexible Demontagetechnologien und optimierte Produktgestaltung
FAPS, 186 Seiten, 103 Bilder. 1998. ISBN 3-87525-108-3.

Band 84: Knuth Götz
Modelle und effiziente Modellbildung zur Qualitätssicherung in der Elektronikproduktion
FAPS, 212 Seiten, 129 Bilder, 24 Tab. 1998. ISBN 3-87525-112-1.

Band 85: Ralf Luchs
Einsatzmöglichkeiten leitender Klebstoffe zur zuverlässigen Kontaktierung elektronischer Bauelemente in der SMT
FAPS, 176 Seiten, 126 Bilder, 30 Tab. 1998. ISBN 3-87525-113-7.

Band 86: Frank Pöhlau
Entscheidungsgrundlagen zur Einführung räumlicher spritzgegossener Schaltungsträger (3-D MID)
FAPS, 144 Seiten, 99 Bilder. 1999. ISBN 3-87525-114-8.

Band 87: Roland T. A. Kals
Fundamentals on the miniaturization of sheet metal working processes
LFT, 128 Seiten, 58 Bilder, 11 Tab. 1999. ISBN 3-87525-115-6.

Band 88: Gerhard Luhn
Implizites Wissen und technisches Handeln am Beispiel der Elektronikproduktion
FAPS, 252 Seiten, 61 Bilder, 1 Tab. 1999. ISBN 3-87525-116-4.

Band 89: Axel Sprenger
Adaptives Streckbiegen von Aluminium-Strangpreßprofilen
LFT, 114 Seiten, 63 Bilder, 4 Tab. 1999. ISBN 3-87525-117-2.

Band 90: Hans-Jörg Pucher
Untersuchungen zur Prozeßfolge Umformen, Bestücken und Laserstrahllöten von Mikrokontakten
LFT, 158 Seiten, 69 Bilder, 9 Tab. 1999. ISBN 3-87525-119-9.

Band 91: Horst Arnet
Profilbiegen mit kinematischer Gestalterzeugung
LFT, 128 Seiten, 67 Bilder, 7 Tab. 1999. ISBN 3-87525-120-2.

Band 92: Doris Schubart
Prozeßmodellierung und Technologieentwicklung beim Abtragen mit CO_2-Laserstrahlung
LFT, 133 Seiten, 57 Bilder, 13 Tab. 1999. ISBN 3-87525-122-9.

Band 93: Adrianus L. P. Coremans
Laserstrahlsintern von Metallpulver - Prozeßmodellierung, Systemtechnik, Eigenschaften laserstrahlgesinterter Metallkörper
LFT, 184 Seiten, 108 Bilder, 12 Tab. 1999. ISBN 3-87525-124-5.

Band 94: Hans-Martin Biehler
Optimierungskonzepte für Qualitätsdatenverarbeitung und Informationsbereitstellung in der Elektronikfertigung
FAPS, 194 Seiten, 105 Bilder. 1999. ISBN 3-87525-126-1.

Band 95: Wolfgang Becker
Oberflächenausbildung und tribologische Eigenschaften excimerlaserstrahlbearbeiteter Hochleistungskeramiken
LFT, 175 Seiten, 71 Bilder, 3 Tab. 1999. ISBN 3-87525-127-X.

Band 96: Philipp Hein
Innenhochdruck-Umformen von Blechpaaren: Modellierung, Prozeßauslegung und Prozeßführung
LFT, 129 Seiten, 57 Bilder, 7 Tab. 1999. ISBN 3-87525-128-8.

Band 97: Gunter Beitinger
Herstellungs- und Prüfverfahren für thermoplastische Schaltungsträger
FAPS, 169 Seiten, 92 Bilder, 20 Tab. 1999. ISBN 3-87525-129-6.

Band 98: Jürgen Knoblach
Beitrag zur rechnerunterstützten verursachungsgerechten Angebotskalkulation von Blechteilen mit Hilfe wissensbasierter Methoden
LFT, 155 Seiten, 53 Bilder, 26 Tab. 1999. ISBN 3-87525-130-X.

Band 99: Frank Breitenbach
Bildverarbeitungssystem zur Erfassung der Anschlußgeometrie elektronischer SMT-Bauelemente
LFT, 147 Seiten, 92 Bilder, 12 Tab. 2000. ISBN 3-87525-131-8.

Band 100: Bernd Falk
Simulationsbasierte Lebensdauervorhersage für Werkzeuge der Kaltmassivumformung
LFT, 134 Seiten, 44 Bilder, 15 Tab. 2000. ISBN 3-87525-136-9.

Band 101: Wolfgang Schlögl
Integriertes Simulationsdaten-Management für Maschinenentwicklung und Anlagenplanung
FAPS, 169 Seiten, 101 Bilder, 20 Tab. 2000. ISBN 3-87525-137-7.

Band 102: Christian Hinsel
Ermüdungsbruchversagen hartstoffbeschichteter Werkzeugstähle in der Kaltmassivumformung
LFT, 130 Seiten, 80 Bilder, 14 Tab. 2000. ISBN 3-87525-138-5.

Band 103: Stefan Bobbert
Simulationsgestützte Prozessauslegung für das Innenhochdruck-Umformen von Blechpaaren
LFT, 123 Seiten, 77 Bilder. 2000. ISBN 3-87525-145-8.

Band 104: Harald Rottbauer
Modulares Planungswerkzeug zum Produktionsmanagement in der Elektronikproduktion
FAPS, 166 Seiten, 106 Bilder. 2001.
ISBN 3-87525-139-3.

Band 105: Thomas Hennige
Flexible Formgebung von Blechen durch Laserstrahlumformen
LFT, 119 Seiten, 50 Bilder. 2001.
ISBN 3-87525-140-7.

Band 106: Thomas Menzel
Wissensbasierte Methoden für die rechnergestützte Charakterisierung und Bewertung innovativer Fertigungsprozesse
LFT, 152 Seiten, 71 Bilder. 2001.
ISBN 3-87525-142-3.

Band 107: Thomas Stöckel
Kommunikationstechnische Integration der Prozeßebene in Produktionssysteme durch Middleware-Frameworks
FAPS, 147 Seiten, 65 Bilder, 5 Tab.
2001. ISBN 3-87525-143-1.

Band 108: Frank Pitter
Verfügbarkeitssteigerung von Werkzeugmaschinen durch Einsatz mechatronischer Sensorlösungen
FAPS, 158 Seiten, 131 Bilder, 8 Tab.
2001. ISBN 3-87525-144-X.

Band 109: Markus Korneli
Integration lokaler CAP-Systeme in einen globalen Fertigungsdatenverbund
FAPS, 121 Seiten, 53 Bilder, 11 Tab.
2001. ISBN 3-87525-146-6.

Band 110: Burkhard Müller
Laserstrahljustieren mit Excimer-Lasern - Prozeßparameter und Modelle zur Aktorkonstruktion
LFT, 128 Seiten, 36 Bilder, 9 Tab.
2001. ISBN 3-87525-159-8.

Band 111: Jürgen Göhringer
Integrierte Telediagnose via Internet zum effizienten Service von Produktionssystemen
FAPS, 178 Seiten, 98 Bilder, 5 Tab.
2001. ISBN 3-87525-147-4.

Band 112: Robert Feuerstein
Qualitäts- und kosteneffiziente Integration neuer Bauelementetechnologien in die Flachbaugruppenfertigung
FAPS, 161 Seiten, 99 Bilder, 10 Tab.
2001. ISBN 3-87525-151-2.

Band 113: Marcus Reichenberger
Eigenschaften und Einsatzmöglichkeiten alternativer Elektroniklote in der Oberflächenmontage (SMT)
FAPS, 165 Seiten, 97 Bilder, 18 Tab.
2001. ISBN 3-87525-152-0.

Band 114: Alexander Huber
Justieren vormontierter Systeme mit dem Nd:YAG-Laser unter Einsatz von Aktoren
LFT, 122 Seiten, 58 Bilder, 5 Tab.
2001. ISBN 3-87525-153-9.

Band 115: Sami Krimi
Analyse und Optimierung von Montagesystemen in der Elektronikproduktion
FAPS, 155 Seiten, 88 Bilder, 3 Tab.
2001. ISBN 3-87525-157-1.

Band 116: Marion Merklein
Laserstrahlumformen von Aluminiumwerkstoffen - Beeinflussung der Mikrostruktur und der mechanischen Eigenschaften
LFT, 122 Seiten, 65 Bilder, 15 Tab.
2001. ISBN 3-87525-156-3.

Band 117: Thomas Collisi
Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten
FAPS, 181 Seiten, 105 Bilder, 7 Tab.
2002. ISBN 3-87525-164-4.

Band 118: Markus Koch
Rationalisierung und ergonomische Optimierung im Innenausbau durch den Einsatz moderner Automatisierungstechnik
FAPS, 176 Seiten, 98 Bilder, 9 Tab.
2002. ISBN 3-87525-165-2.

Band 119: Michael Schmidt
Prozeßregelung für das Laserstrahl-Punktschweißen in der Elektronikproduktion
LFT, 152 Seiten, 71 Bilder, 3 Tab.
2002. ISBN 3-87525-166-0.

Band 120: Nicolas Tiesler
Grundlegende Untersuchungen zum Fließpressen metallischer Kleinstteile
LFT, 126 Seiten, 78 Bilder, 12 Tab.
2002. ISBN 3-87525-175-X.

Band 121: Lars Pursche
Methoden zur technologieorientierten Programmierung für die 3D-Lasermikrobearbeitung
LFT, 111 Seiten, 39 Bilder, 0 Tab.
2002. ISBN 3-87525-183-0.

Band 122: Jan-Oliver Brassel
Prozeßkontrolle beim Laserstrahl-Mikroschweißen
LFT, 148 Seiten, 72 Bilder, 12 Tab.
2002. ISBN 3-87525-181-4.

Band 123: Mark Geisel
Prozeßkontrolle und -steuerung beim Laserstrahlschweißen mit den Methoden der nichtlinearen Dynamik
LFT, 135 Seiten, 46 Bilder, 2 Tab.
2002. ISBN 3-87525-180-6.

Band 124: Gerd Eßer
Laserstrahlunterstützte Erzeugung metallischer Leiterstrukturen auf Thermoplastsubstraten für die MID-Technik
LFT, 148 Seiten, 60 Bilder, 6 Tab.
2002. ISBN 3-87525-171-7.

Band 125: Marc Fleckenstein
Qualität laserstrahl-gefügter Mikroverbindungen elektronischer Kontakte
LFT, 159 Seiten, 77 Bilder, 7 Tab.
2002. ISBN 3-87525-170-9.

Band 126: Stefan Kaufmann
Grundlegende Untersuchungen zum Nd:YAG- Laserstrahlfügen von Silizium für Komponenten der Optoelektronik
LFT, 159 Seiten, 100 Bilder, 6 Tab.
2002. ISBN 3-87525-172-5.

Band 127: Thomas Fröhlich
Simultanes Löten von Anschlußkontakten elektronischer Bauelemente mit Diodenlaserstrahlung
LFT, 143 Seiten, 75 Bilder, 6 Tab.
2002. ISBN 3-87525-186-5.

Band 128: Achim Hofmann
Erweiterung der Formgebungsgrenzen beim Umformen von Aluminiumwerkstoffen durch den Einsatz prozessangepasster Platinen
LFT, 113 Seiten, 58 Bilder, 4 Tab. 2002. ISBN 3-87525-182-2.

Band 129: Ingo Kriebitzsch
3 - D MID Technologie in der Automobilelektronik
FAPS, 129 Seiten, 102 Bilder, 10 Tab. 2002. ISBN 3-87525-169-5.

Band 130: Thomas Pohl
Fertigungsqualität und Umformbarkeit laserstrahlgeschweißter Formplatinen aus Aluminiumlegierungen
LFT, 133 Seiten, 93 Bilder, 12 Tab. 2002. ISBN 3-87525-173-3.

Band 131: Matthias Wenk
Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industrierobotern
FAPS, 167 Seiten, 85 Bilder, 1 Tab. 2002. ISBN 3-87525-174-1.

Band 132: Matthias Negendanck
Neue Sensorik und Aktorik für Bearbeitungsköpfe zum Laserstrahlschweißen
LFT, 116 Seiten, 60 Bilder, 14 Tab. 2002. ISBN 3-87525-184-9.

Band 133: Oliver Kreis
Integrierte Fertigung - Verfahrensintegration durch Innenhochdruck-Umformen, Trennen und Laserstrahlschweißen in einem Werkzeug sowie ihre tele- und multimediale Präsentation
LFT, 167 Seiten, 90 Bilder, 43 Tab. 2002. ISBN 3-87525-176-8.

Band 134: Stefan Trautner
Technische Umsetzung produktbezogener Instrumente der Umweltpolitik bei Elektro- und Elektronikgeräten
FAPS, 179 Seiten, 92 Bilder, 11 Tab. 2002. ISBN 3-87525-177-6.

Band 135: Roland Meier
Strategien für einen produktorientierten Einsatz räumlicher spritzgegossener Schaltungsträger (3-D MID)
FAPS, 155 Seiten, 88 Bilder, 14 Tab. 2002. ISBN 3-87525-178-4.

Band 136: Jürgen Wunderlich
Kostensimulation - Simulationsbasierte Wirtschaftlichkeitsregelung komplexer Produktionssysteme
FAPS, 202 Seiten, 119 Bilder, 17 Tab. 2002. ISBN 3-87525-179-2.

Band 137: Stefan Novotny
Innenhochdruck-Umformen von Blechen aus Aluminium- und Magnesiumlegierungen bei erhöhter Temperatur
LFT, 132 Seiten, 82 Bilder, 6 Tab. 2002. ISBN 3-87525-185-7.

Band 138: Andreas Licha
Flexible Montageautomatisierung zur Komplettmontage flächenhafter Produktstrukturen durch kooperierende Industrieroboter
FAPS, 158 Seiten, 87 Bilder, 8 Tab. 2003. ISBN 3-87525-189-X.

Band 139: Michael Eisenbarth
Beitrag zur Optimierung der Aufbau- und Verbindungstechnik für mechatronische Baugruppen
FAPS, 207 Seiten, 141 Bilder, 9 Tab. 2003. ISBN 3-87525-190-3.

Band 140: Frank Christoph
Durchgängige simulationsgestützte Planung von Fertigungseinrichtungen der Elektronikproduktion
FAPS, 187 Seiten, 107 Bilder, 9 Tab. 2003. ISBN 3-87525-191-1.

Band 141: Hinnerk Hagenah
Simulationsbasierte Bestimmung der zu erwartenden Maßhaltigkeit für das Blechbiegen
LFT, 131 Seiten, 36 Bilder, 26 Tab. 2003. ISBN 3-87525-192-X.

Band 142: Ralf Eckstein
Scherschneiden und Biegen metallischer Kleinstteile - Materialeinfluss und Materialverhalten
LFT, 148 Seiten, 71 Bilder, 19 Tab. 2003. ISBN 3-87525-193-8.

Band 143: Frank H. Meyer-Pittroff
Excimerlaserstrahlbiegen dünner metallischer Folien mit homogener Lichtlinie
LFT, 138 Seiten, 60 Bilder, 16 Tab. 2003. ISBN 3-87525-196-2.

Band 144: Andreas Kach
Rechnergestützte Anpassung von Laserstrahlschneidbahnen an Bauteilabweichungen
LFT, 139 Seiten, 69 Bilder, 11 Tab. 2004. ISBN 3-87525-197-0.

Band 145: Stefan Hierl
System- und Prozeßtechnik für das simultane Löten mit Diodenlaserstrahlung von elektronischen Bauelementen
LFT, 124 Seiten, 66 Bilder, 4 Tab. 2004. ISBN 3-87525-198-9.

Band 146: Thomas Neudecker
Tribologische Eigenschaften keramischer Blechumformwerkzeuge-Einfluss einer Oberflächenendbearbeitung mittels Excimerlaserstrahlung
LFT, 166 Seiten, 75 Bilder, 26 Tab. 2004. ISBN 3-87525-200-4.

Band 147: Ulrich Wenger
Prozessoptimierung in der Wickeltechnik durch innovative maschinenbauliche und regelungstechnische Ansätze
FAPS, 132 Seiten, 88 Bilder, 0 Tab. 2004. ISBN 3-87525-203-9.

Band 148: Stefan Slama
Effizienzsteigerung in der Montage durch marktorientierte Montagestrukturen und erweiterte Mitarbeiterkompetenz
FAPS, 188 Seiten, 125 Bilder, 0 Tab. 2004. ISBN 3-87525-204-7.

Band 149: Thomas Wurm
Laserstrahljustieren mittels Aktoren-Entwicklung von Konzepten und Methoden für die rechnerunterstützte Modellierung und Optimierung von komplexen Aktorsystemen in der Mikrotechnik
LFT, 122 Seiten, 51 Bilder, 9 Tab. 2004. ISBN 3-87525-206-3.

Band 150: Martino Celeghini
Wirkmedienbasierte Blechumformung: Grundlagenuntersuchungen zum Einfluss von Werkstoff und Bauteilgeometrie
LFT, 146 Seiten, 77 Bilder, 6 Tab.
2004. ISBN 3-87525-207-1.

Band 151: Ralph Hohenstein
Entwurf hochdynamischer Sensor- und Regelsysteme für die adaptiveLaserbearbeitung
LFT, 282 Seiten, 63 Bilder, 16 Tab.
2004. ISBN 3-87525-210-1.

Band 152: Angelika Hutterer
Entwicklung prozessüberwachender Regelkreise für flexible Formgebungsprozesse
LFT, 149 Seiten, 57 Bilder, 2 Tab.
2005. ISBN 3-87525-212-8.

Band 153: Emil Egerer
Massivumformen metallischer Kleinstteile bei erhöhter Prozesstemperatur
LFT, 158 Seiten, 87 Bilder, 10 Tab.
2005. ISBN 3-87525-213-6.

Band 154: Rüdiger Holzmann
Strategien zur nachhaltigen Optimierung von Qualität und Zuverlässigkeit in der Fertigung hochintegrierter Flachbaugruppen
FAPS, 186 Seiten, 99 Bilder, 19 Tab.
2005. ISBN 3-87525-217-9.

Band 155: Marco Nock
Biegeumformen mit Elastomerwerkzeugen Modellierung, Prozessauslegung und Abgrenzung des Verfahrens am Beispiel des Rohrbiegens
LFT, 164 Seiten, 85 Bilder, 13 Tab.
2005. ISBN 3-87525-218-7.

Band 156: Frank Niebling
Qualifizierung einer Prozesskette zum Laserstrahlsintern metallischer Bauteile
LFT, 148 Seiten, 89 Bilder, 3 Tab.
2005. ISBN 3-87525-219-5.

Band 157: Markus Meiler
Großserientauglichkeit trockenschmierstoffbeschichteter Aluminiumbleche im Presswerk Grundlegende Untersuchungen zur Tribologie, zum Umformverhalten und Bauteilversuche
LFT, 104 Seiten, 57 Bilder, 21 Tab.
2005. ISBN 3-87525-221-7.

Band 158: Agus Sutanto
Solution Approaches for Planning of Assembly Systems in Three-Dimensional Virtual Environments
FAPS, 169 Seiten, 98 Bilder, 3 Tab.
2005. ISBN 3-87525-220-9.

Band 159: Matthias Boiger
Hochleistungssysteme für die Fertigung elektronischer Baugruppen auf der Basis flexibler Schaltungsträger
FAPS, 175 Seiten, 111 Bilder, 8 Tab.
2005. ISBN 3-87525-222-5.

Band 160: Matthias Pitz
Laserunterstütztes Biegen höchstfester Mehrphasenstähle
LFT, 120 Seiten, 73 Bilder, 11 Tab.
2005. ISBN 3-87525-223-3.

Band 161: Meik Vahl
Beitrag zur gezielten Beeinflussung des Werkstoffflusses beim Innenhochdruck-Umformen von Blechen
LFT, 165 Seiten, 94 Bilder, 15 Tab.
2005. ISBN 3-87525-224-1.

Band 162: Peter K. Kraus
Plattformstrategien - Realisierung einer varianz- und kostenoptimierten Wertschöpfung
FAPS, 181 Seiten, 95 Bilder, 0 Tab.
2005. ISBN 3-87525-226-8.

Band 163: Adrienn Cser
Laserstrahlschmelzabtrag - Prozessanalyse und -modellierung
LFT, 146 Seiten, 79 Bilder, 3 Tab.
2005. ISBN 3-87525-227-6.

Band 164: Markus C. Hahn
Grundlegende Untersuchungen zur Herstellung von Leichtbauverbundstrukturen mit Aluminiumschaumkern
LFT, 143 Seiten, 60 Bilder, 16 Tab.
2005. ISBN 3-87525-228-4.

Band 165: Gordana Michos
Mechatronische Ansätze zur Optimierung von Vorschubachsen
FAPS, 146 Seiten, 87 Bilder, 17 Tab.
2005. ISBN 3-87525-230-6.

Band 166: Markus Stark
Auslegung und Fertigung hochpräziser Faser-Kollimator-Arrays
LFT, 158 Seiten, 115 Bilder, 11 Tab.
2005. ISBN 3-87525-231-4.

Band 167: Yurong Zhou
Kollaboratives Engineering Management in der integrierten virtuellen Entwicklung der Anlagen für die Elektronikproduktion
FAPS, 156 Seiten, 84 Bilder, 6 Tab.
2005. ISBN 3-87525-232-2.

Band 168: Werner Enser
Neue Formen permanenter und lösbarer elektrischer Kontaktierungen für mechatronische Baugruppen
FAPS, 190 Seiten, 112 Bilder, 5 Tab.
2005. ISBN 3-87525-233-0.

Band 169: Katrin Melzer
Integrierte Produktpolitik bei elektrischen und elektronischen Geräten zur Optimierung des Product-Life-Cycle
FAPS, 155 Seiten, 91 Bilder, 17 Tab.
2005. ISBN 3-87525-234-9.

Band 170: Alexander Putz
Grundlegende Untersuchungen zur Erfassung der realen Vorspannung von armierten Kaltfließpresswerkzeugen mittels Ultraschall
LFT, 137 Seiten, 71 Bilder, 15 Tab.
2006. ISBN 3-87525-237-3.

Band 171: Martin Prechtl
Automatisiertes Schichtverfahren für metallische Folien - System- und Prozesstechnik
LFT, 154 Seiten, 45 Bilder, 7 Tab.
2006. ISBN 3-87525-238-1.

Band 172: Markus Meidert
Beitrag zur deterministischen Lebensdauerabschätzung von Werkzeugen der Kaltmassivumformung
LFT, 131 Seiten, 78 Bilder, 9 Tab.
2006. ISBN 3-87525-239-X.

Band 173: Bernd Müller
Robuste, automatisierte Montagesysteme durch adaptive Prozessführung und montageübergreifende Fehlerprävention am Beispiel flächiger Leichtbauteile
FAPS, 147 Seiten, 77 Bilder, 0 Tab.
2006. ISBN 3-87525-240-3.

Band 174: Alexander Hofmann
Hybrides Laserdurchstrahlschweißen von Kunststoffen
LFT, 136 Seiten, 72 Bilder, 4 Tab.
2006. ISBN 978-3-87525-243-9.

Band 175: Peter Wölflick
Innovative Substrate und Prozesse mit feinsten Strukturen für bleifreie Mechatronik-Anwendungen
FAPS, 177 Seiten, 148 Bilder, 24 Tab. 2006.
ISBN 978-3-87525-246-0.

Band 176: Attila Komlodi
Detection and Prevention of Hot Cracks during Laser Welding of Aluminium Alloys Using Advanced Simulation Methods
LFT, 155 Seiten, 89 Bilder, 14 Tab. 2006. ISBN 978-3-87525-248-4.

Band 177: Uwe Popp
Grundlegende Untersuchungen zum Laserstrahlstrukturieren von Kaltmassivumformwerkzeugen
LFT, 140 Seiten, 67 Bilder, 16 Tab. 2006. ISBN 978-3-87525-249-1.

Band 178: Veit Rückel
Rechnergestützte Ablaufplanung und Bahngenerierung Für kooperierende Industrieroboter
FAPS, 148 Seiten, 75 Bilder, 7 Tab. 2006. ISBN 978-3-87525-250-7.

Band 179: Manfred Dirscherl
Nicht-thermische Mikrojustiertechnik mittels ultrakurzer Laserpulse
LFT, 154 Seiten, 69 Bilder, 10 Tab. 2007. ISBN 978-3-87525-251-4.

Band 180: Yong Zhuo
Entwurf eines rechnergestützten integrierten Systems für Konstruktion und Fertigungsplanung räumlicher spritzgegossener Schaltungsträger (3D-MID)
FAPS, 181 Seiten, 95 Bilder, 5 Tab. 2007. ISBN 978-3-87525-253-8.

Band 181: Stefan Lang
Durchgängige Mitarbeiterinformation zur Steigerung von Effizienz und Prozesssicherheit in der Produktion
FAPS, 172 Seiten, 93 Bilder. 2007. ISBN 978-3-87525-257-6.

Band 182: Hans-Joachim Krauß
Laserstrahlinduzierte Pyrolyse präkeramischer Polymere
LFT, 171 Seiten, 100 Bilder. 2007. ISBN 978-3-87525-258-3.

Band 183: Stefan Junker
Technologien und Systemlösungen für die flexibel automatisierte Bestückung permanent erregter Läufer mit oberflächenmontierten Dauermagneten
FAPS, 173 Seiten, 75 Bilder. 2007. ISBN 978-3-87525-259-0.

Band 184: Rainer Kohlbauer
Wissensbasierte Methoden für die simulationsgestützte Auslegung wirkmedienbasierter Blechumformprozesse
LFT, 135 Seiten, 50 Bilder. 2007. ISBN 978-3-87525-260-6.

Band 185: Klaus Lamprecht
Wirkmedienbasierte Umformung tiefgezogener Vorformen unter besonderer Berücksichtigung maßgeschneiderter Halbzeuge
LFT, 137 Seiten, 81 Bilder. 2007. ISBN 978-3-87525-265-1.

Band 186: Bernd Zolleiß
Optimierte Prozesse und Systeme für die Bestückung mechatronischerBaugruppen
FAPS, 180 Seiten, 117 Bilder. 2007. ISBN 978-3-87525-266-8.

Band 187: Michael Kerausch
Simulationsgestützte Prozessauslegung für das Umformen lokal wärmebehandelter Aluminiumplatinen
LFT, 146 Seiten, 76 Bilder, 7 Tab. 2007. ISBN 978-3-87525-267-5.

Band 188: Matthias Weber
Unterstützung der Wandlungsfähigkeit von Produktionsanlagen durch innovative Softwaresysteme
FAPS, 183 Seiten, 122 Bilder, 3 Tab. 2007. ISBN 978-3-87525-269-9.

Band 189: Thomas Frick
Untersuchung der prozessbestimmenden Strahl-Stoff-Wechselwirkungen beim Laserstrahlschweißen von Kunststoffen
LFT, 104 Seiten, 62 Bilder, 8 Tab. 2007. ISBN 978-3-87525-268-2.

Band 190: Joachim Hecht
Werkstoffcharakterisierung und Prozessauslegung für die wirkmedienbasierte Doppelblech-Umformung von Magnesiumlegierungen
LFT, 107 Seiten, 91 Bilder, 2 Tab. 2007. ISBN 978-3-87525-270-5.

Band 191: Ralf Völkl
Stochastische Simulation zur Werkzeuglebensdaueroptimierung und Präzisionsfertigung in der Kaltmassivumformung
LFT, 178 Seiten, 75 Bilder, 12 Tab. 2008. ISBN 978-3-87525-272-9.

Band 192: Massimo Tolazzi
Innenhochdruck-Umformen verstärkter Blech-Rahmenstrukturen
LFT, 164 Seiten, 85 Bilder, 7 Tab. 2008. ISBN 978-3-87525-273-6.

Band 193: Cornelia Hoff
Untersuchung der Prozesseinflussgrößen beim Presshärten des höchstfesten Vergütungsstahls 22MnB5
LFT, 133 Seiten, 92 Bilder, 5 Tab. 2008. ISBN 978-3-87525-275-0.

Band 194: Christian Alvarez
Simulationsgestützte Methoden zur effizienten Gestaltung von Lötprozessen in der Elektronikproduktion
FAPS, 149 Seiten, 86 Bilder, 8 Tab. 2008. ISBN 978-3-87525-277-4.

Band 195: Andreas Kunze
Automatisierte Montage von makromechatronischen Modulen zur flexiblen Integration in hybride Pkw-Bordnetzsysteme
FAPS, 160 Seiten, 90 Bilder, 14 Tab. 2008.
ISBN 978-3-87525-278-1.

Band 196: Wolfgang Hußnätter
Grundlegende Untersuchungen zur experimentellen Ermittlung und zur Modellierung von Fließortkurven bei erhöhten Temperaturen
LFT, 152 Seiten, 73 Bilder, 21 Tab. 2008. ISBN 978-3-87525-279-8.

Band 197: Thomas Bigl
Entwicklung, angepasste Herstellungsverfahren und erweiterte Qualitätssicherung von einsatzgerechten elektronischen Baugruppen
FAPS, 175 Seiten, 107 Bilder, 14 Tab. 2008.
ISBN 978-3-87525-280-4.

Band 198: Stephan Roth
Grundlegende Untersuchungen zum Excimerlaserstrahl-Abtragen unter Flüssigkeitsfilmen
LFT, 113 Seiten, 47 Bilder, 14 Tab. 2008. ISBN 978-3-87525-281-1.

Band 199: Artur Giera
Prozesstechnische Untersuchungen zum Rührreibschweißen metallischer Werkstoffe
LFT, 179 Seiten, 104 Bilder, 36 Tab. 2008. ISBN 978-3-87525-282-8.

Band 200: Jürgen Lechler
Beschreibung und Modellierung des Werkstoffverhaltens von presshärtbaren Bor-Manganstählen
LFT, 154 Seiten, 75 Bilder, 12 Tab. 2009. ISBN 978-3-87525-286-6.

Band 201: Andreas Blankl
Untersuchungen zur Erhöhung der Prozessrobustheit bei der Innenhochdruck-Umformung von flächigen Halbzeugen mit vor- bzw. nachgeschalteten Laserstrahlfügeoperationen
LFT, 120 Seiten, 68 Bilder, 9 Tab. 2009. ISBN 978-3-87525-287-3.

Band 202: Andreas Schaller
Modellierung eines nachfrageorientierten Produktionskonzeptes für mobile Telekommunikationsgeräte
FAPS, 120 Seiten, 79 Bilder, 0 Tab. 2009. ISBN 978-3-87525-289-7.

Band 203: Claudius Schimpf
Optimierung von Zuverlässigkeitsuntersuchungen, Prüfabläufen und Nacharbeitsprozessen in der Elektronikproduktion
FAPS, 162 Seiten, 90 Bilder, 14 Tab. 2009.
ISBN 978-3-87525-290-3.

Band 204: Simon Dietrich
Sensoriken zur Schwerpunktslagebestimmung der optischen Prozessemissionen beim Laserstrahltiefschweißen
LFT, 138 Seiten, 70 Bilder, 5 Tab. 2009. ISBN 978-3-87525-292-7.

Band 205: Wolfgang Wolf
Entwicklung eines agentenbasierten Steuerungssystems zur Materialflussorganisation im wandelbaren Produktionsumfeld
FAPS, 167 Seiten, 98 Bilder. 2009. ISBN 978-3-87525-293-4.

Band 206: Steffen Polster
Laserdurchstrahlschweißen transparenter Polymerbauteile
LFT, 160 Seiten, 92 Bilder, 13 Tab. 2009. ISBN 978-3-87525-294-1.

Band 207: Stephan Manuel Dörfler
Rührreibschweißen von walzplattiertem Halbzeug und Aluminiumblech zur Herstellung flächiger Aluminiumschaum-Sandwich-Verbundstrukturen
LFT, 190 Seiten, 98 Bilder, 5 Tab. 2009. ISBN 978-3-87525-295-8.

Band 208: Uwe Vogt
Seriennahe Auslegung von Aluminium Tailored Heat Treated Blanks
LFT, 151 Seiten, 68 Bilder, 26 Tab. 2009. ISBN 978-3-87525-296-5.

Band 209: Till Laumann
Qualitative und quantitative Bewertung der Crashtauglichkeit von höchstfesten Stählen
LFT, 117 Seiten, 69 Bilder, 7 Tab. 2009. ISBN 978-3-87525-299-6.

Band 210: Alexander Diehl
Größeneffekte bei Biegeprozessen-Entwicklung einer Methodik zur Identifikation und Quantifizierung
LFT, 180 Seiten, 92 Bilder, 12 Tab. 2010. ISBN 978-3-87525-302-3.

Band 211: Detlev Staud
Effiziente Prozesskettenauslegung für das Umformen lokal wärmebehandelter und geschweißter Aluminiumbleche
LFT, 164 Seiten, 72 Bilder, 12 Tab. 2010. ISBN 978-3-87525-303-0.

Band 212: Jens Ackermann
Prozesssicherung beim Laserdurchstrahlschweißen thermoplastischer Kunststoffe
LPT, 129 Seiten, 74 Bilder, 13 Tab. 2010. ISBN 978-3-87525-305-4.

Band 213: Stephan Weidel
Grundlegende Untersuchungen zum Kontaktzustand zwischen Werkstück und Werkzeug bei umformtechnischen Prozessen unter tribologischen Gesichtspunkten
LFT, 144 Seiten, 67 Bilder, 11 Tab. 2010. ISBN 978-3-87525-307-8.

Band 214: Stefan Geißdörfer
Entwicklung eines mesoskopischen Modells zur Abbildung von Größeneffekten in der Kaltmassivumformung mit Methoden der FE-Simulation
LFT, 133 Seiten, 83 Bilder, 11 Tab. 2010. ISBN 978-3-87525-308-5.

Band 215: Christian Matzner
Konzeption produktspezifischer Lösungen zur Robustheitssteigerung elektronischer Systeme gegen die Einwirkung von Betauung im Automobil
FAPS, 165 Seiten, 93 Bilder, 14 Tab. 2010. ISBN 978-3-87525-309-2.

Band 216: Florian Schüßler
Verbindungs- und Systemtechnik für thermisch hochbeanspruchte und miniaturisierte elektronische Baugruppen
FAPS, 184 Seiten, 93 Bilder, 18 Tab. 2010.
ISBN 978-3-87525-310-8.

Band 217: Massimo Cojutti
Strategien zur Erweiterung der Prozessgrenzen bei der Innhochdruck-Umformung von Rohren und Blechpaaren
LFT, 125 Seiten, 56 Bilder, 9 Tab. 2010. ISBN 978-3-87525-312-2.

Band 218: Raoul Plettke
Mehrkriterielle Optimierung komplexer Aktorsysteme für das Laserstrahljustieren
LFT, 152 Seiten, 25 Bilder, 3 Tab. 2010. ISBN 978-3-87525-315-3.

Band 219: Andreas Dobroschke
Flexible Automatisierungslösungen für die Fertigung wickeltechnischer Produkte
FAPS, 184 Seiten, 109 Bilder, 18 Tab. 2011.
ISBN 978-3-87525-317-7.

Band 220: Azhar Zam
Optical Tissue Differentiation for Sensor-Controlled Tissue-Specific Laser Surgery
LPT, 99 Seiten, 45 Bilder, 8 Tab. 2011. ISBN 978-3-87525-318-4.

Band 221: Michael Rösch
Potenziale und Strategien zur Optimierung des Schablonendruckprozesses in der Elektronikproduktion
FAPS, 192 Seiten, 127 Bilder, 19 Tab. 2011.
ISBN 978-3-87525-319-1.

Band 222: Thomas Rechtenwald
Quasi-isothermes Laserstrahlsintern von Hochtemperatur-Thermoplasten - Eine Betrachtung werkstoff-prozessspezifischer Aspekte am Beispiel PEEK
LPT, 150 Seiten, 62 Bilder, 8 Tab. 2011. ISBN 978-3-87525-320-7.

Band 223: Daniel Craiovan
Prozesse und Systemlösungen für die SMT-Montage optischer Bauelemente auf Substrate mit integrierten Lichtwellenleitern
FAPS, 165 Seiten, 85 Bilder, 8 Tab. 2011. ISBN 978-3-87525-324-5.

Band 224: Kay Wagner
Beanspruchungsangepasste Kaltmassivumformwerkzeuge durch lokal optimierte Werkzeugoberflächen
LFT, 147 Seiten, 103 Bilder, 17 Tab. 2011. ISBN 978-3-87525-325-2.

Band 225: Martin Brandhuber
Verbesserung der Prognosegüte des Versagens von Punktschweißverbindungen bei höchstfesten Stahlgüten
LFT, 155 Seiten, 91 Bilder, 19 Tab. 2011. ISBN 978-3-87525-327-6.

Band 226: Peter Sebastian Feuser
Ein Ansatz zur Herstellung von pressgehärteten Karosseriekomponenten mit maßgeschneiderten mechanischen Eigenschaften: Temperierte Umformwerkzeuge. Prozessfenster, Prozesssimuation und funktionale Untersuchung
LFT, 195 Seiten, 97 Bilder, 60 Tab. 2012. ISBN 978-3-87525-328-3.

Band 227: Murat Arbak
Material Adapted Design of Cold Forging Tools Exemplified by Powder Metallurgical Tool Steels and Ceramics
LFT, 109 Seiten, 56 Bilder, 8 Tab. 2012. ISBN 978-3-87525-330-6.

Band 228: Indra Pitz
Beschleunigte Simulation des Laserstrahlumformens von Aluminiumblechen
LPT, 137 Seiten, 45 Bilder, 27 Tab. 2012. ISBN 978-3-87525-333-7.

Band 229: Alexander Grimm
Prozessanalyse und -überwachung des Laserstrahlhartlötens mittels optischer Sensorik
LPT, 125 Seiten, 61 Bilder, 5 Tab. 2012. ISBN 978-3-87525-334-4.

Band 230: Markus Kaupper
Biegen von höhenfesten Stahlblechwerkstoffen - Umformverhalten und Grenzen der Biegbarkeit
LFT, 160 Seiten, 57 Bilder, 10 Tab. 2012. ISBN 978-3-87525-339-9.

Band 231: Thomas Kroiß
Modellbasierte Prozessauslegung für die Kaltmassivumformung unter Brücksichtigung der Werkzeug- und Pressenauffederung
LFT, 169 Seiten, 50 Bilder, 19 Tab. 2012. ISBN 978-3-87525-341-2.

Band 232: Christian Goth
Analyse und Optimierung der Entwicklung und Zuverlässigkeit räumlicher Schaltungsträger (3D-MID)
FAPS, 176 Seiten, 102 Bilder, 22 Tab. 2012.
ISBN 978-3-87525-340-5.

Band 233: Christian Ziegler
Ganzheitliche Automatisierung mechatronischer Systeme in der Medizin am Beispiel Strahlentherapie
FAPS, 170 Seiten, 71 Bilder, 19 Tab. 2012. ISBN 978-3-87525-342-9.

Band 234: Florian Albert
Automatisiertes Laserstrahllöten und -reparaturlöten elektronischer Baugruppen
LPT, 127 Seiten, 78 Bilder, 11 Tab. 2012. ISBN 978-3-87525-344-3.

Band 235: Thomas Stöhr
Analyse und Beschreibung des mechanischen Werkstoffverhaltens von presshärtbaren Bor-Manganstählen
LFT, 118 Seiten, 74 Bilder, 18 Tab. 2013. ISBN 978-3-87525-346-7.

Band 236: Christian Kägeler
Prozessdynamik beim Laserstrahlschweißen verzinkter Stahlbleche im Überlappstoß
LPT, 145 Seiten, 80 Bilder, 3 Tab. 2013. ISBN 978-3-87525-347-4.

Band 237: Andreas Sulzberger
Seriennahe Auslegung der Prozesskette zur wärmeunterstützten Umformung von Aluminiumblechwerkstoffen
LFT, 153 Seiten, 87 Bilder, 17 Tab. 2013. ISBN 978-3-87525-349-8.

Band 238: Simon Opel
Herstellung prozessangepasster Halbzeuge mit variabler Blechdicke durch die Anwendung von Verfahren der Blechmassivumformung
LFT, 165 Seiten, 108 Bilder, 27 Tab. 2013. ISBN 978-3-87525-350-4.

Band 239: Rajesh Kanawade
In-vivo Monitoring of Epithelium Vessel and Capillary Density for the Application of Detection of Clinical Shock and Early Signs of Cancer Development
LPT, 124 Seiten, 58 Bilder, 15 Tab. 2013. ISBN 978-3-87525-351-1.

Band 240: Stephan Busse
Entwicklung und Qualifizierung eines Schneidclinchverfahrens
LFT, 119 Seiten, 86 Bilder, 20 Tab. 2013. ISBN 978-3-87525-352-8.

Band 241: Karl-Heinz Leitz
Mikro- und Nanostrukturierung mit kurz und ultrakurz gepulster Laserstrahlung
LPT, 154 Seiten, 71 Bilder, 9 Tab. 2013. ISBN 978-3-87525-355-9.

Band 242: Markus Michl
Webbasierte Ansätze zur ganzheitlichen technischen Diagnose
FAPS, 182 Seiten, 62 Bilder, 20 Tab. 2013.
ISBN 978-3-87525-356-6.

Band 243: Vera Sturm
Einfluss von Chargenschwankungen auf die Verarbeitungsgrenzen von Stahlwerkstoffen
LFT, 113 Seiten, 58 Bilder, 9 Tab. 2013. ISBN 978-3-87525-357-3.

Band 244: Christian Neudel
Mikrostrukturelle und mechanisch-technologische Eigenschaften widerstandspunktgeschweißter Aluminium-Stahl-Verbindungen für den Fahrzeugbau
LFT, 178 Seiten, 171 Bilder, 31 Tab. 2014. ISBN 978-3-87525-358-0.

Band 245: Anja Neumann
Konzept zur Beherrschung der Prozessschwankungen im Presswerk
LFT, 162 Seiten, 68 Bilder, 15 Tab. 2014. ISBN 978-3-87525-360-3.

Band 246: Ulf-Hermann Quentin
Laserbasierte Nanostrukturierung mit optisch positionierten Mikrolinsen
LPT, 137 Seiten, 89 Bilder, 6 Tab. 2014. ISBN 978-3-87525-361-0.

Band 247: Erik Lamprecht
Der Einfluss der Fertigungsverfahren auf die Wirbelstromverluste von Stator-Einzelzahnblechpaketen für den Einsatz in Hybrid- und Elektrofahrzeugen
FAPS, 148 Seiten, 138 Bilder, 4 Tab. 2014. ISBN 978-3-87525-362-7.

Band 248: Sebastian Rösel
Wirkmedienbasierte Umformung von Blechhalbzeugen unter Anwendung magnetorheologischer Flüssigkeiten als kombiniertes Wirk- und Dichtmedium
LFT, 148 Seiten, 61 Bilder, 12 Tab. 2014. ISBN 978-3-87525-363-4.

Band 249: Paul Hippchen
Simulative Prognose der Geometrie indirekt pressgehärteter Karosseriebauteile für die industrielle Anwendung
LFT, 163 Seiten, 89 Bilder, 12 Tab. 2014. ISBN 978-3-87525-364-1.

Band 250: Martin Zubeil
Versagensprognose bei der Prozesssimulation von Biegeumform- und Falzverfahren
LFT, 171 Seiten, 90 Bilder, 5 Tab. 2014. ISBN 978-3-87525-365-8.

Band 251: Alexander Kühl
Flexible Automatisierung der Statorenmontage mit Hilfe einer universellen ambidexteren Kinematik
FAPS, 142 Seiten, 60 Bilder, 26 Tab. 2014.
ISBN 978-3-87525-367-2.

Band 252: Thomas Albrecht
Optimierte Fertigungstechnologien für Rotoren getriebeintegrierter PM-Synchronmotoren von Hybridfahrzeugen
FAPS, 198 Seiten, 130 Bilder, 38 Tab. 2014.
ISBN 978-3-87525-368-9.

Band 253: Florian Risch
Planning and Production Concepts for Contactless Power Transfer Systems for Electric Vehicles
FAPS, 185 Seiten, 125 Bilder, 13 Tab. 2014.
ISBN 978-3-87525-369-6.

Band 254: Markus Weigl
Laserstrahlschweißen von Mischverbindungen aus austenitischen und ferritischen korrosionsbeständigen Stahlwerkstoffen
LPT, 184 Seiten, 110 Bilder, 6 Tab. 2014. ISBN 978-3-87525-370-2.

Band 255: Johannes Noneder
Beanspruchungserfassung für die Validierung von FE-Modellen zur Auslegung von Massivumformwerkzeugen
LFT, 161 Seiten, 65 Bilder, 14 Tab. 2014. ISBN 978-3-87525-371-9.

Band 256: Andreas Reinhardt
Ressourceneffiziente Prozess- und Produktionstechnologie für flexible Schaltungsträger
FAPS, 123 Seiten, 69 Bilder, 19 Tab. 2014. ISBN 978-3-87525-373-3.

Band 257: Tobias Schmuck
Ein Beitrag zur effizienten Gestaltung globaler Produktions- und Logistiknetzwerke mittels Simulation
FAPS, 151 Seiten, 74 Bilder. 2014.
ISBN 978-3-87525-374-0.

Band 258: Bernd Eichenhüller
Untersuchungen der Effekte und Wechselwirkungen charakteristischer Einflussgrößen auf das Umformverhalten bei Mikroumformprozessen
LFT, 127 Seiten, 29 Bilder, 9 Tab. 2014. ISBN 978-3-87525-375-7.

Band 259: Felix Lütteke
Vielseitiges autonomes Transportsystem basierend auf Weltmodellerstellung mittels Datenfusion von Deckenkameras und Fahrzeugsensoren
FAPS, 152 Seiten, 54 Bilder, 20 Tab. 2014.
ISBN 978-3-87525-376-4.

Band 260: Martin Grüner
Hochdruck-Blechumformung mit formlos festen Stoffen als Wirkmedium
LFT, 144 Seiten, 66 Bilder, 29 Tab. 2014. ISBN 978-3-87525-379-5.

Band 261: Christian Brock
Analyse und Regelung des Laserstrahltiefschweißprozesses durch Detektion der Metalldampffackelposition
LPT, 126 Seiten, 65 Bilder, 3 Tab. 2015. ISBN 978-3-87525-380-1.

Band 262: Peter Vatter
Sensitivitätsanalyse des 3-Rollen-Schubbiegens auf Basis der Finite Elemente Methode
LFT, 145 Seiten, 57 Bilder, 26 Tab. 2015. ISBN 978-3-87525-381-8.

Band 263: Florian Klämpfl
Planung von Laserbestrahlungen durch simulationsbasierte Optimierung
LPT, 169 Seiten, 78 Bilder, 32 Tab. 2015. ISBN 978-3-87525-384-9.

Band 264: Matthias Domke
Transiente physikalische Mechanismen bei der Laserablation von dünnen Metallschichten
LPT, 133 Seiten, 43 Bilder, 3 Tab.
2015. ISBN 978-3-87525-385-6.

Band 265: Johannes Götz
Community-basierte Optimierung des Anlagenengineerings
FAPS, 177 Seiten, 80 Bilder, 30 Tab.
2015.
ISBN 978-3-87525-386-3.

Band 266: Hung Nguyen
Qualifizierung des Potentials von Verfestigungseffekten zur Erweiterung des Umformvermögens aushärtbarer Aluminiumlegierungen
LFT, 137 Seiten, 57 Bilder, 16 Tab.
2015. ISBN 978-3-87525-387-0.

Band 267: Andreas Kuppert
Erweiterung und Verbesserung von Versuchs- und Auswertetechniken für die Bestimmung von Grenzformänderungskurven
LFT, 138 Seiten, 82 Bilder, 2 Tab.
2015. ISBN 978-3-87525-388-7.

Band 268: Kathleen Klaus
Erstellung eines Werkstofforientierten Fertigungsprozessfensters zur Steigerung des Formgebungsvermögens von Alumi-niumlegierungen unter Anwendung einer zwischengeschalteten Wärmebehandlung
LFT, 154 Seiten, 70 Bilder, 8 Tab.
2015. ISBN 978-3-87525-391-7.

Band 269: Thomas Svec
Untersuchungen zur Herstellung von funktionsoptimierten Bauteilen im partiellen Presshärtprozess mittels lokal unterschiedlich temperierter Werkzeuge
LFT, 166 Seiten, 87 Bilder, 15 Tab.
2015. ISBN 978-3-87525-392-4.

Band 270: Tobias Schrader
Grundlegende Untersuchungen zur Verschleißcharakterisierung beschichteter Kaltmassivumformwerkzeuge
LFT, 164 Seiten, 55 Bilder, 11 Tab.
2015. ISBN 978-3-87525-393-1.

Band 271: Matthäus Brela
Untersuchung von Magnetfeld-Messmethoden zur ganzheitlichen Wertschöpfungsoptimierung und Fehlerdetektion an magnetischen Aktoren
FAPS, 170 Seiten, 97 Bilder, 4 Tab.
2015. ISBN 978-3-87525-394-8.

Band 272: Michael Wieland
Entwicklung einer Methode zur Prognose adhäsiven Verschleißes an Werkzeugen für das direkte Presshärten
LFT, 156 Seiten, 84 Bilder, 9 Tab.
2015. ISBN 978-3-87525-395-5.

Band 273: René Schramm
Strukturierte additive Metallisierung durch kaltaktives Atmosphärendruckplasma
FAPS, 136 Seiten, 62 Bilder, 15 Tab.
2015. ISBN 978-3-87525-396-2.

Band 274: Michael Lechner
Herstellung beanspruchungsangepasster Aluminiumblechhalbzeuge durch eine maßgeschneiderte Variation der Abkühlgeschwindigkeit nach Lösungsglühen
LFT, 136 Seiten, 62 Bilder, 15 Tab.
2015. ISBN 978-3-87525-397-9.

Band 275: Kolja Andreas
Einfluss der Oberflächenbeschaffenheit auf das Werkzeugeinsatzverhalten beim Kaltfließpressen
LFT, 169 Seiten, 76 Bilder, 4 Tab.
2015. ISBN 978-3-87525-398-6.

Band 276: Marcus Baum
Laser Consolidation of ITO Nanoparticles for the Generation of Thin Conductive Layers on Transparent Substrates
LPT, 158 Seiten, 75 Bilder, 3 Tab.
2015. ISBN 978-3-87525-399-3.

Band 277: Thomas Schneider
Umformtechnische Herstellung dünnwandiger Funktionsbauteile aus Feinblech durch Verfahren der Blechmassivumformung
LFT, 188 Seiten, 95 Bilder, 7 Tab.
2015. ISBN 978-3-87525-401-3.

Band 278: Jochen Merhof
Sematische Modellierung automatisierter Produktionssysteme zur Verbesserung der IT-Integration zwischen Anlagen-Engineering und Steuerungsebene
FAPS, 157 Seiten, 88 Bilder, 8 Tab.
2015. ISBN 978-3-87525-402-0.

Band 279: Fabian Zöller
Erarbeitung von Grundlagen zur Abbildung des tribologischen Systems in der Umformsimulation
LFT, 126 Seiten, 51 Bilder, 3 Tab.
2016. ISBN 978-3-87525-403-7.

Band 280: Christian Hezler
Einsatz technologischer Versuche zur Erweiterung der Versagensvorhersage bei Karosseriebauteilen aus höchstfesten Stählen
LFT, 147 Seiten, 63 Bilder, 44 Tab.
2016. ISBN 978-3-87525-404-4.

Band 281: Jochen Bönig
Integration des Systemverhaltens von Automobil-Hochvoltleitungen in die virtuelle Absicherung durch strukturmechanische Simulation
FAPS, 177 Seiten, 107 Bilder, 17 Tab.
2016.
ISBN 978-3-87525-405-1.

Band 282: Johannes Kohl
Automatisierte Datenerfassung für diskret ereignisorientierte Simulationen in der energieflexibelen Fabrik
FAPS, 160 Seiten, 80 Bilder, 27 Tab.
2016.
ISBN 978-3-87525-406-8.

Band 283: Peter Bechtold
Mikroschockwellenumformung mittels ultrakurzer Laserpulse
LPT, 155 Seiten, 59 Bilder, 10 Tab.
2016. ISBN 978-3-87525-407-5.

Band 284: Stefan Berger
Laserstrahlschweißen thermoplastischer Kohlenstofffaserverbundwerkstoffe mit spezifischem Zusatzdraht
LPT, 118 Seiten, 68 Bilder, 9 Tab.
2016. ISBN 978-3-87525-408-2.

Band 285: Martin Bornschlegl
Methods-Energy Measurement - Eine Methode zur Energieplanung für Fügeverfahren im Karosseriebau
FAPS, 136 Seiten, 72 Bilder, 46 Tab. 2016.
ISBN 978-3-87525-409-9.

Band 286: Tobias Rackow
Erweiterung des Unternehmenscontrollings um die Dimension Energie
FAPS, 164 Seiten, 82 Bilder, 29 Tab. 2016.
ISBN 978-3-87525-410-5.

Band 287: Johannes Koch
Grundlegende Untersuchungen zur Herstellung zyklisch-symmetrischer Bauteile mit Nebenformelementen durch Blechmassivumformung
LFT, 125 Seiten, 49 Bilder, 17 Tab. 2016. ISBN 978-3-87525-411-2.

Band 288: Hans Ulrich Vierzigmann
Beitrag zur Untersuchung der tribologischen Bedingungen in der Blechmassivumformung - Bereitstellung von tribologischen Modellversuchen und Realisierung von Tailored Surfaces
LFT, 174 Seiten, 102 Bilder, 34 Tab. 2016. ISBN 978-3-87525-412-9.

Band 289: Thomas Senner
Methodik zur virtuellen Absicherung der formgebenden Operation des Nasspressprozesses von Gelege-Mehrschichtverbunden
LFT, 156 Seiten, 96 Bilder, 21 Tab. 2016. ISBN 978-3-87525-414-3.

Band 290: Sven Kreitlein
Der grundoperationsspezifische Mindestenergiebedarf als Referenzwert zur Bewertung der Energieeffizienz in der Produktion
FAPS, 185 Seiten, 64 Bilder, 30 Tab. 2016.
ISBN 978-3-87525-415-0.

Band 291: Christian Roos
Remote-Laserstrahlschweißen verzinkter Stahlbleche in Kehlnahtgeometrie
LPT, 123 Seiten, 52 Bilder, 0 Tab. 2016. ISBN 978-3-87525-416-7.

Band 292: Alexander Kahrimanidis
Thermisch unterstützte Umformung von Aluminiumblechen
LFT, 165 Seiten, 103 Bilder, 18 Tab. 2016. ISBN 978-3-87525-417-4.

Band 293: Jan Tremel
Flexible Systems for Permanent Magnet Assembly and Magnetic Rotor Measurement / Flexible Systeme zur Montage von Permanentmagneten und zur Messung magnetischer Rotoren
FAPS, 152 Seiten, 91 Bilder, 12 Tab. 2016. ISBN 978-3-87525-419-8.

Band 294: Ioannis Tsoupis
Schädigungs- und Versagensverhalten hochfester Leichtbauwerkstoffe unter Biegebeanspruchung
LFT, 176 Seiten, 51 Bilder, 6 Tab. 2017. ISBN 978-3-87525-420-4.

Band 295: Sven Hildering
Grundlegende Untersuchungen zum Prozessverhalten von Silizium als Werkzeugwerkstoff für das Mikroscherschneiden metallischer Folien
LFT, 177 Seiten, 74 Bilder, 17 Tab. 2017. ISBN 978-3-87525-422-8.

Band 296: Sasia Mareike Hertweck
Zeitliche Pulsformung in der Lasermikromaterialbearbeitung – Grundlegende Untersuchungen und Anwendungen
LPT, 146 Seiten, 67 Bilder, 5 Tab. 2017. ISBN 978-3-87525-423-5.

Band 297: Paryanto
Mechatronic Simulation Approach for the Process Planning of Energy-Efficient Handling Systems
FAPS, 162 Seiten, 86 Bilder, 13 Tab. 2017. ISBN 978-3-87525-424-2.

Band 298: Peer Stenzel
Großserientaugliche Nadelwickeltechnik für verteilte Wicklungen im Anwendungsfall der E-Traktionsantriebe
FAPS, 239 Seiten, 147 Bilder, 20 Tab. 2017.
ISBN 978-3-87525-425-9.

Band 299: Mario Lušić
Ein Vorgehensmodell zur Erstellung montageführender Werkerinformationssysteme simultan zum Produktentstehungsprozess
FAPS, 174 Seiten, 79 Bilder, 22 Tab. 2017.
ISBN 978-3-87525-426-6.

Band 300: Arnd Buschhaus
Hochpräzise adaptive Steuerung und Regelung robotergeführter Prozesse
FAPS, 202 Seiten, 96 Bilder, 4 Tab. 2017. ISBN 978-3-87525-427-3.

Band 301: Tobias Laumer
Erzeugung von thermoplastischen Werkstoffverbunden mittels simultanem, intensitätsselektivem Laserstrahlschmelzen
LPT, 140 Seiten, 82 Bilder, 0 Tab. 2017. ISBN 978-3-87525-428-0.

Band 302: Nora Unger
Untersuchung einer thermisch unterstützten Fertigungskette zur Herstellung umgeformter Bauteile aus der höherfesten Aluminiumlegierung EN AW-7020
LFT, 142 Seiten, 53 Bilder, 8 Tab. 2017. ISBN 978-3-87525-429-7.

Band 303: Tommaso Stellin
Design of Manufacturing Processes for the Cold Bulk Forming of Small Metal Components from Metal Strip
LFT, 146 Seiten, 67 Bilder, 7 Tab. 2017. ISBN 978-3-87525-430-3.

Band 304: Bassim Bachy
Experimental Investigation, Modeling, Simulation and Optimization of Molded Interconnect Devices (MID) Based on Laser Direct Structuring (LDS) / Experimentelle Untersuchung, Modellierung, Simulation und Optimierung von Molded Interconnect Devices (MID) basierend auf Laser Direktstrukturierung (LDS)
FAPS, 168 Seiten, 120 Bilder, 26 Tab. 2017.
ISBN 978-3-87525-431-0.

Band 305: Michael Spahr
Automatisierte Kontaktierungsverfahren für flachleiterbasierte Pkw-Bordnetzsysteme
FAPS, 197 Seiten, 98 Bilder, 17 Tab. 2017. ISBN 978-3-87525-432-7.

Band 306: Sebastian Suttner
Charakterisierung und Modellierung des spannungszustandsabhängigen Werkstoffverhaltens der Magnesiumlegierung AZ31B für die numerische Prozessauslegung
LFT, 150 Seiten, 84 Bilder, 19 Tab. 2017. ISBN 978-3-87525-433-4.

Band 307: Bhargav Potdar
A reliable methodology to deduce thermo-mechanical flow behaviour of hot stamping steels
LFT, 203 Seiten, 98 Bilder, 27 Tab. 2017. ISBN 978-3-87525-436-5.

Band 308: Maria Löffler
Steuerung von Blechmassivumformprozessen durch maßgeschneiderte tribologische Systeme
LFT, viii u. 166 Seiten, 90 Bilder, 5 Tab. 2018. ISBN 978-3-96147-133-1.

Band 309: Martin Müller
Untersuchung des kombinierten Trenn- und Umformprozesses beim Fügen artungleicher Werkstoffe mittels Schneidclinchverfahren
LFT, xi u. 149 Seiten, 89 Bilder, 6 Tab. 2018.
ISBN: 978-3-96147-135-5.

Band 310: Christopher Kästle
Qualifizierung der Kupfer-Drahtbondtechnologie für integrierte Leistungsmodule in harschen Umgebungsbedingungen
FAPS, xii u. 167 Seiten, 70 Bilder, 18 Tab. 2018.
ISBN 978-3-96147-145-4.

Band 311: Daniel Vipavc
Eine Simulationsmethode für das 3-Rollen-Schubbiegen
LFT, xiii u. 121 Seiten, 56 Bilder, 17 Tab. 2018. ISBN 978-3-96147-147-8.

Band 312: Christina Ramer
Arbeitsraumüberwachung und autonome Bahnplanung für ein sicheres und flexibles Roboter-Assistenzsystem in der Fertigung
FAPS, xiv u. 188 Seiten, 57 Bilder, 9 Tab. 2018.
ISBN 978-3-96147-153-9.

Band 313: Miriam Rauer
Der Einfluss von Poren auf die Zuverlässigkeit der Lötverbindungen von Hochleistungs-Leuchtdioden
FAPS, xii u. 209 Seiten, 108 Bilder, 21 Tab. 2018.
ISBN 978-3-96147-157-7.

Band 314: Felix Tenner
Kamerabasierte Untersuchungen der Schmelze und Gasströmungen beim Laserstrahlschweißen verzinkter Stahlbleche
LPT, xxiii u. 184 Seiten, 94 Bilder, 7 Tab. 2018.
ISBN 978-3-96147-160-7.

Band 315: Aarief Syed-Khaja
Diffusion Soldering for High-temperature Packaging of Power Electronics
FAPS, x u. 202 Seiten, 144 Bilder, 32 Tab. 2018.
ISBN 978-3-87525-162-1.

Band 316: Adam Schaub
Grundlagenwissenschaftliche Untersuchung der kombinierten Prozesskette aus Umformen und Additive Fertigung
LFT, xi u. 192 Seiten, 72 Bilder, 27 Tab. 2019.
ISBN 978-3-96147-166-9.

Band 317: Daniel Gröbel
Herstellung von Nebenformelementen unterschiedlicher Geometrie an Blechen mittels Fließpressverfahren der Blechmassivumformung
LFT, x u. 165 Seiten, 96 Bilder, 13 Tab. 2019. ISBN 978-3-96147-168-3.

Band 318: Philipp Hildenbrand
Entwicklung einer Methodik zur Herstellung von Tailored Blanks mit definierten Halbzeugeigenschaften durch einen Taumelprozess
LFT, ix u. 153 Seiten, 77 Bilder, 4 Tab. 2019. ISBN 978-3-96147-174-4.

Band 319: Tobias Konrad
Simulative Auslegung der Spann- und Fixierkonzepte im Karosserierohbau: Bewertung der Baugruppenmaßhaltigkeit unter Berücksichtigung schwankender Einflussgrößen
LFT, x u. 203 Seiten, 134 Bilder, 32 Tab. 2019.
ISBN 978-3-96147-176-8.

Band 320: David Meinel
Architektur applikationsspezifischer Multi-Physics-Simulationskonfiguratoren am Beispiel modularer Triebzüge
FAPS, xii u. 166 Seiten, 82 Bilder, 25 Tab. 2019.
ISBN 978-3-96147-184-3.

Band 321: Andrea Zimmermann
Grundlegende Untersuchungen zum Einfluss fertigungsbedingter Eigenschaften auf die Ermüdungsfestigkeit kaltmassivumgeformter Bauteile
LFT, ix u. 160 Seiten, 66 Bilder, 5 Tab. 2019.
ISBN 978-3-96147-190-4.

Band 322: Christoph Amann
Simulative Prognose der Geometrie nassgepresster Karosseriebauteile aus Gelege-Mehrschichtverbunden
LFT, xvi u. 169 Seiten, 80 Bilder, 13 Tab. 2019.
ISBN 978-3-96147-194-2.

Band 323: Jennifer Tenner
Realisierung schmierstofffreier Tiefziehprozesse durch maßgeschneiderte Werkzeugoberflächen
LFT, x u. 187 Seiten, 68 Bilder, 13 Tab. 2019.
ISBN 978-3-96147-196-6.

Band 324: Susan Zöller
Mapping Individual Subjective Values to Product Design
KTmfk, xi u. 223 Seiten, 81 Bilder, 25 Tab. 2019.
ISBN 978-3-96147-202-4.

Band 325: Stefan Lutz
Erarbeitung einer Methodik zur semiempirischen Ermittlung der Umwandlungskinetik durchhärtender Wälzlagerstähle für die Wärmebehandlungssimulation
LFT, xiv u. 189 Seiten, 75 Bilder, 32 Tab. 2019.
ISBN 978-3-96147-209-3.

Band 326: Tobias Gnibl
Modellbasierte Prozesskettenabbildung rührreibgeschweißter Aluminiumhalbzeuge zur umformtechnischen Herstellung höchstfester Leichtbau-strukturteile
LFT, xii u. 167 Seiten, 68 Bilder, 17 Tab. 2019.
ISBN 978-3-96147-217-8.

Band 327: Johannes Bürner
Technisch-wirtschaftliche Optionen zur Lastflexibilisierung durch intelligente elektrische Wärmespeicher
FAPS, xiv u. 233 Seiten, 89 Bilder, 27 Tab. 2019.
ISBN 978-3-96147-219-2.

Band 328: Wolfgang Böhm
Verbesserung des Umformverhaltens von mehrlagigen Aluminiumblechwerkstoffen mit ultrafeinkörnigem Gefüge
LFT, ix u. 160 Seiten, 88 Bilder, 14 Tab. 2019.
ISBN 978-3-96147-227-7.

Band 329: Stefan Landkammer
Grundsatzuntersuchungen, mathematische Modellierung und Ableitung einer Auslegungsmethodik für Gelenkantriebe nach dem Spinnenbeinprinzip
LFT, xii u. 200 Seiten, 83 Bilder, 13 Tab. 2019.
ISBN 978-3-96147-229-1.

Band 330: Stephan Rapp
Pump-Probe-Ellipsometrie zur Messung transienter optischer Materialeigen-schaften bei der Ultrakurzpuls-Lasermaterialbearbeitung
LPT, xi u. 143 Seiten, 49 Bilder, 2 Tab. 2019.
ISBN 978-3-96147-235-2.

Band 331: Michael Scholz
Intralogistics Execution System mit integrierten autonomen, servicebasierten Transportentitäten
FAPS, xi u. 195 Seiten, 55 Bilder, 11 Tab. 2019.
ISBN 978-3-96147-237-6.

Band 332: Eva Bogner
Strategien der Produktindividualisierung in der produzierenden Industrie im Kontext der Digitalisierung
FAPS, ix u. 201 Seiten, 55 Bilder, 28 Tab. 2019.
ISBN 978-3-96147-246-8.

Band 333: Daniel Benjamin Krüger
Ein Ansatz zur CAD-integrierten muskuloskelettalen Analyse der Mensch-Maschine-Interaktion
KTmfk, x u. 217 Seiten, 102 Bilder, 7 Tab. 2019.
ISBN 978-3-96147-250-5.

Band 334: Thomas Kuhn
Qualität und Zuverlässigkeit laserdirektstrukturierter mechatronisch integrierter Baugruppen (LDS-MID)
FAPS, ix u. 152 Seiten, 69 Bilder, 12 Tab. 2019.
ISBN: 978-3-96147-252-9.

Band 335: Hans Fleischmann
Modellbasierte Zustands- und Prozessüberwachung auf Basis sozio-cyber-physischer Systeme
FAPS, xi u. 214 Seiten, 111 Bilder, 18 Tab. 2019.
ISBN: 978-3-96147-256-7.

Band 336: Markus Michalski
Grundlegende Untersuchungen zum Prozess- und Werkstoffverhalten bei schwingungsüberlagerter Umformung
LFT, xii u. 197 Seiten, 93 Bilder, 11 Tab. 2019.
ISBN: 978-3-96147-270-3.

Band 337: Markus Brandmeier
Ganzheitliches ontologiebasiertes Wissensmanagement im Umfeld der industriellen Produktion
FAPS, xi u. 255 Seiten, 77 Bilder, 33 Tab. 2020.
ISBN: 978-3-96147-275-8.

Band 338: Stephan Purr
Datenerfassung für die Anwendung lernender Algorithmen bei der Herstellung von Blechformteilen
LFT, ix u. 165 Seiten, 48 Bilder, 4 Tab. 2020.
ISBN: 978-3-96147-281-9.

Band 339: Christoph Kiener
Kaltfließpressen von gerad- und schrägverzahnten Zahnrädern
LFT, viii u. 151 Seiten, 81 Bilder, 3 Tab. 2020.
ISBN 978-3-96147-287-1.

Band 340: Simon Spreng
Numerische, analytische und empirische Modellierung des Heißcrimpprozesses
FAPS, xix u. 204 Seiten, 91 Bilder, 27 Tab. 2020.
ISBN 978-3-96147-293-2.

Band 341: Patrik Schwingenschlögl
Erarbeitung eines Prozessverständnisses zur Verbesserung der tribologischen Bedingungen beim Presshärten
LFT, x u. 177 Seiten, 81 Bilder, 8 Tab. 2020.
ISBN 978-3-96147-297-0.

Band 342: Emanuela Affronti
Evaluation of failure behaviour of sheet metals
LFT, ix u. 136 Seiten, 57 Bilder, 20 Tab. 2020.
ISBN 978-3-96147-303-8.

Band 343: Julia Degner
Grundlegende Untersuchungen zur Herstellung hochfester Aluminiumblechbauteile in einem kombinierten Umform- und Abschreckprozess
LFT, x u. 172 Seiten, 61 Bilder, 9 Tab. 2020.
ISBN 978-3-96147-307-6.

Band 344: Maximilian Wagner
Automatische Bahnplanung für die Aufteilung von Prozessbewegungen in synchrone Werkstück- und Werkzeugbewegungen mittels Multi-Roboter-Systemen
FAPS, xxi u. 181 Seiten, 111 Bilder, 15 Tab. 2020.
ISBN 978-3-96147-309-0.

Band 345: Stefan Härter
Qualifizierung des Montageprozesses hochminiaturisierter elektronischer Bauelemente
FAPS, ix u. 194 Seiten, 97 Bilder, 28 Tab. 2020.
ISBN 978-3-96147-314-4.

Band 346: Toni Donhauser
Ressourcenorientierte Auftragsregelung in einer hybriden Produktion mittels betriebsbegleitender Simulation
FAPS, xix u. 242 Seiten, 97 Bilder, 17 Tab. 2020.
ISBN 978-3-96147-316-8.

Band 347: Philipp Amend
Laserbasiertes Schmelzkleben von Thermoplasten mit Metallen
LPT, xv u. 154 Seiten, 67 Bilder. 2020. ISBN 978-3-96147-326-7.

Band 348: Matthias Ehlert
Simulationsunterstützte funktionale Grenzlagenabsicherung
KTmfk, xvi u. 300 Seiten, 101 Bilder, 73 Tab. 2020.
ISBN 978-3-96147-328-1.

Band 349: Thomas Sander
Ein Beitrag zur Charakterisierung und Auslegung des Verbundes von Kunststoffsubstraten mit harten Dünnschichten
KTmfk, xiv u. 178 Seiten, 88 Bilder, 21 Tab. 2020.
ISBN 978-3-96147-330-4.

Band 350: Florian Pilz
Fließpressen von Verzahnungselementen an Blechen
LFT, x u. 170 Seiten, 103Bilder, 4 Tab. 2020.
ISBN 978-3-96147-332-8.

Band 351: Sebastian Josef Katona
Evaluation und Aufbereitung von Produktsimulationen mittels abweichungsbehafteter Geometriemodelle
KTmfk, ix u. 147 Seiten, 73 Bilder, 11 Tab. 2020.
ISBN 978-3-96147-336-6.

Band 352: Jürgen Herrmann
Kumulatives Walzplattieren. Bewertung der Umformeigenschaften mehrlagiger Blechwerkstoffe der ausscheidungshärtbaren Legierung AA6014
LFT, x u. 157 Seiten, 64 Bilder, 5 Tab. 2020.
ISBN 978-3-96147-344-1.

Band 353: Christof Küstner
Assistenzsystem zur Unterstützung der datengetriebenen Produktentwicklung
KTmfk, xii u. 219 Seiten, 63 Bilder, 14 Tab. 2020.
ISBN 978-3-96147-348-9.

Band 354: Tobias Gläßel
Prozessketten zum Laserstrahlschweißen von flachleiterbasierten Formspulenwicklungen für automobile Traktionsantriebe
FAPS, xiv u. 206 Seiten, 89 Bilder, 11 Tab. 2020.
ISBN 978-3-96147-356-4.

Band 355: Andreas Meinel
Experimentelle Untersuchung der Auswirkungen von Axialschwingungen auf Reibung und Verschleiß in Zylinderrol-lenlagern
KTmfk, xii u. 162 Seiten, 56 Bilder, 7 Tab. 2020.
ISBN 978-3-96147-358-8.

Band 356: Hannah Riedle
Haptische, generische Modelle weicher anatomischer Strukturen für die chirurgische Simulation
FAPS, xxx u. 179 Seiten, 82 Bilder, 35 Tab. 2020.
ISBN 978-3-96147-367-0.

Band 357: Maximilian Landgraf
Leistungselektronik für den Einsatz dielektrischer Elastomere in aktorischen, sensorischen und integrierten sensomotorischen Systemen
FAPS, xxiii u. 166 Seiten, 71 Bilder, 10 Tab. 2020.
ISBN 978-3-96147-380-9.

Band 358: Alireza Esfandyari
Multi-Objective Process Optimization for Overpressure Reflow Soldering in Electronics Production
FAPS, xviii u. 175 Seiten, 57 Bilder, 23 Tab. 2020.
ISBN 978-3-96147-382-3.

Band 359: Christian Sand
Prozessübergreifende Analyse komplexer Montageprozessketten mittels Data Mining
FAPS, XV u. 168 Seiten, 61 Bilder, 12 Tab. 2021.
ISBN 978-3-96147-398-4.

Band 360: Ralf Merkl
Closed-Loop Control of a Storage-Supported Hybrid Compensation System for Improving the Power Quality in Medium Voltage Networks
FAPS, xxvii u. 200 Seiten, 102 Bilder, 2 Tab. 2021.
ISBN 978-3-96147-402-8.

Band 361: Thomas Reitberger
Additive Fertigung polymerer optischer Wellenleiter im Aerosol-Jet-Verfahren
FAPS, xix u. 141 Seiten, 65 Bilder, 11 Tab. 2021.
ISBN 978-3-96147-400-4.

Band 362: Marius Christian Fechter
Modellierung von Vorentwürfen in der virtuellen Realität mit natürlicher Fingerinteraktion
KTmfk, x u. 188 Seiten, 67 Bilder, 19 Tab. 2021.
ISBN 978-3-96147-404-2.

Band 363: Franziska Neubauer
Oberflächenmodifizierung und Entwicklung einer Auswertemethodik zur Verschleißcharakterisierung im Presshärteprozess
LFT, ix u. 177 Seiten, 42 Bilder, 6 Tab. 2021.
ISBN 978-3-96147-406-6.

Band 364: Eike Wolfram Schäffer
Web- und wissensbasierter Engineering-Konfigurator für roboterzentrierte Automatisierungslösungen
FAPS, xxiv u. 195 Seiten, 108 Bilder, 25 Tab. 2021.
ISBN 978-3-96147-410-3.

Band 365: Daniel Gross
Untersuchungen zur kohlenstoffdioxidbasierten kryogenen Minimalmengenschmierung
REP, xii u. 184 Seiten, 56 Bilder, 18 Tab. 2021.
ISBN 978-3-96147-412-7.

Band 366: Daniel Junker
Qualifizierung laser-additiv gefertigter Komponenten für den Einsatz im Werkzeugbau der Massivumformung
LFT, vii u. 142 Seiten, 62 Bilder, 5 Tab. 2021.
ISBN 978-3-96147-416-5.

Band 367: Tallal Javied
Totally Integrated Ecology Management for Resource Efficient and Eco-Friendly Production
FAPS, xv u. 160 Seiten, 60 Bilder, 13 Tab. 2021.
ISBN 978-3-96147-418-9.

Band 368: David Marco Hochrein
Wälzlager im Beschleunigungsfeld – Eine Analysestrategie zur Bestimmung des Reibungs-, Axialschub- und Temperaturverhaltens von Nadelkränzen –
KTmfk, xiii u. 279 Seiten, 108 Bilder, 39 Tab. 2021.
ISBN 978-3-96147-420-2.

Band 369: Daniel Gräf
Funktionalisierung technischer Oberflächen mittels prozessüberwachter aerosolbasierter Drucktechnologie
FAPS, xxii u. 175 Seiten, 97 Bilder, 6 Tab. 2021.
ISBN 978-3-96147-433-2.

Band 370: Andreas Gröschl
Hochfrequent fokusabstandsmodulierte Konfokalsensoren für die Nanokoordinatenmesstechnik
FMT, x u. 144 Seiten, 98 Bilder, 6 Tab. 2021.
ISBN 978-3-96147-435-6.

Band 371: Johann Tüchsen
Konzeption, Entwicklung und Einführung des Assistenzsystems D-DAS für die Produktentwicklung elektrischer Motoren
KTmfk, xii u. 178 Seiten, 92 Bilder, 12 Tab. 2021.
ISBN 978-3-96147-437-0.

Band 372: Max Marian
Numerische Auslegung von Oberflächenmikrotexturen für geschmierte tribologische Kontakte
KTmfk, xviii u. 276 Seiten, 85 Bilder, 45 Tab. 2021.
ISBN 978-3-96147-439-4.

Band 373: Johannes Strauß
Die akustooptische Strahlformung in der Lasermaterialbearbeitung
LPT, xvi u. 113 Seiten, 48 Bilder. 2021. ISBN 978-3-96147-441-7.

Band 374: Martin Hohmann
Machine learning and hyper spectral imaging: Multi Spectral Endoscopy in the Gastro Intestinal Tract towards Hyper Spectral Endoscopy
LPT, x u. 137 Seiten, 62 Bilder, 29 Tab. 2021.
ISBN 978-3-96147-445-5.

Band 375: Timo Kordaß
Lasergestütztes Verfahren zur selektiven Metallisierung von epoxidharzbasierten Duromeren zur Steigerung der Integrationsdichte für dreidimensionale mechatronische Package-Baugruppen
FAPS, xviii u. 198 Seiten, 92 Bilder, 24 Tab. 2021.
ISBN 978-3-96147-443-1.

Band 376: Philipp Kestel
Assistenzsystem für den wissensbasierten Aufbau konstruktionsbegleitender Finite-Elemente-Analysen
KTmfk, xviii u. 209 Seiten, 57 Bilder, 17 Tab. 2021.
ISBN 978-3-96147-457-8.

Band 377: Martin Lerchen
Messverfahren für die pulverbettbasierte additive Fertigung zur Sicherstellung der Konformität mit geometrischen Produktspezifikationen
FMT, x u. 150 Seiten, 60 Bilder, 9 Tab. 2021.
ISBN 978-3- 96147-463-9.

Band 378: Michael Schneider
Inline-Prüfung der Permeabilität in weichmagnetischen Komponenten
FAPS, xxii u. 189 Seiten, 79 Bilder, 14 Tab. 2021.
ISBN 978-3-96147-465-3.

Band 379: Tobias Sprügel
Sphärische Detektorflächen als Unterstützung der Produktentwicklung zur Datenanalyse im Rahmen des Digital Engineering
KTmfk, xiii u. 213 Seiten, 84 Bilder, 33 Tab. 2021.
ISBN 978-3-96147-475-2.

Band 380: Tom Häfner
Multipulseffekte beim Mikro-Materialabtrag von Stahllegierungen mit Pikosekunden-Laserpulsen
LPT, xxviii u. 159 Seiten, 57 Bilder, 13 Tab. 2021.
ISBN 978-3-96147-479-0.

Band 381: Björn Heling
Einsatz und Validierung virtueller Absicherungsmethoden für abweichungs-behaftete Mechanismen im Kontext des Robust Design
KTmfk, xi u. 169 Seiten, 63 Bilder, 27 Tab. 2021.
ISBN 978-3-96147-487-5.

Band 382: Tobias Kolb
Laserstrahl-Schmelzen von Metallen mit einer Serienanlage – Prozesscharakterisierung und Erweiterung eines Überwachungssystems
LPT, xv u. 170 Seiten, 128 Bilder, 16 Tab. 2021.
ISBN 978-3-96147-491-2.

Band 383: Mario Meinhardt
Widerstandselementschweißen mit gestauchten Hilfsfügeelementen - Umformtechnische Wirkzusammenhänge zur Beeinflussung der Verbindungsfestigkeit
LFT, xii u. 189 Seiten, 87 Bilder, 4 Tab. 2022.
ISBN 978-3-96147-473-8.

Band 384: Felix Bauer
Ein Beitrag zur digitalen Auslegung von Fügeprozessen im Karosseriebau mit Fokus auf das Remote-Laserstrahlschweißen unter Einsatz flexibler Spanntechnik
LFT, xi u. 185 Seiten, 74 Bilder, 12 Tab. 2022.
ISBN 978-3-96147-498-1.

Band 385: Jochen Zeitler
Konzeption eines rechnergestützten Konstruktionssystems für optomechatronische Baugruppen
FAPS, xix u. 172 Seiten, 88 Bilder, 11 Tab. 2022.
ISBN 978-3-96147-499-8.

Band 386: Vincent Mann
Einfluss von Strahloszillation auf das Laserstrahlschweißen hochfester Stähle
LPT, xiii u. 172 Seiten, 103 Bilder, 18 Tab. 2022.
ISBN 978-3-96147-503-2.

Band 387: Chen Chen
Skin-equivalent opto-/elastofluidic in-vitro microphysiological vascular models for translational studies of optical biopsies
LPT, xx u. 126 Seiten, 60 Bilder, 10 Tab. 2022.
ISBN 978-3-96147-505-6.

Band 388: Stefan Stein
Laser drop on demand joining as bonding method for electronics assembly and packaging with high thermal requirements
LPT, x u. 112 Seiten, 54 Bilder, 10 Tab. 2022.
ISBN 978-3-96147-507-0

Band 389: Nikolaus Urban
Untersuchung des Laserstrahlschmelzens von Neodym-Eisen-Bor zur additiven Herstellung von Permanentmagneten
FAPS, x u. 174 Seiten, 88 Bilder, 18 Tab. 2022.
ISBN 978-3-96147-501-8.

Band 390: Yiting Wu
Großflächige Topographiemessungen mit einem Weißlichtinterferenzmikroskop und einem metrologischen Rasterkraftmikroskop
FMT, xii u. 142 Seiten, 68 Bilder, 11 Tab. 2022.
ISBN: 978-3-96147-513-1.

Band 391: Thomas Papke
Untersuchungen zur Umformbarkeit hybrider Bauteile aus Blechgrundkörper und additiv gefertigter Struktur
LFT, xii u. 194 Seiten, 71 Bilder, 16 Tab. 2022.
ISBN 978-3-96147-515-5.

Band 392: Bastian Zimmermann
Einfluss des Vormaterials auf die mehrstufige Kaltumformung vom Draht
LFT, xi u. 182 Seiten, 36 Bilder, 6 Tab. 2022.
ISBN 978-3-96147-519-3.

Band 393: Harald Völkl
Ein simulationsbasierter Ansatz zur Auslegung additiv gefertigter FLM-Faserverbundstrukturen
KTmfk, xx u. 204 Seiten, 95 Bilder, 22 Tab. 2022.
ISBN 978-3-96147-523-0.

Band 394: Robert Schulte
Auslegung und Anwendung prozessangepasster Halbzeuge für Verfahren der Blechmassivumformung
LFT, x u. 163 Seiten, 93 Bilder, 5 Tab. 2022.
ISBN 978-3-96147-525-4.

Band 395: Philipp Frey
Umformtechnische Strukturierung metallischer Einleger im Folgeverbund für mediendichte Kunststoff-Metall-Hybridbauteile
LFT, ix u. 180 Seiten, 83 Bilder, 7 Tab. 2022.
ISBN 978-3-96147-534-6.

Band 396: Thomas Johann Luft
Komplexitätsmanagement in der Produktentwicklung - Holistische Modellierung, Analyse, Visualisierung und Bewertung komplexer Systeme
KTmfk, xiii u. 510 Seiten, 166 Bilder, 16 Tab. 2022.
ISBN 978-3-96147-540-7.

Band 397: Li Wang
Evaluierung der Einsetzbarkeit des lasergestützten Verfahrens zur selektiven Metallisierung für die Verbesserung passiver Intermodulation in Hochfrequenzanwendungen
FAPS, xxii u.151 Seiten, 72 Bilder, 22 Tab. 2022.
ISBN 978-3-96147-542-1.

Band 398: Sebastian Reitelshöfer
Der Aerosol-Jet-Druck Dielektrischer Elastomere als additives Fertigungsverfahren für elastische mechatronische Komponenten
FAPS, xxv u. 206 Seiten, 87 Bilder, 13 Tab. 2022.
ISBN 978-3-96147-547-6.

Band 399: Alexander Meyer
Selektive Magnetmontage zur Verringerung des Rastmomentes permanenterregter Synchronmotoren
FAPS, xv u. 164 Seiten, 90 Bilder, 18 Tab. 2022.
ISBN 978-3-96147-555-1.

Band 400: Rong Zhao
Design verschleißreduzierender amorpher Kohlenstoffschichtsysteme für trockene tribologische Gleitkontakte
KTmfk, x u. 148 Seiten, 69 Bilder, 14 Tab. 2022.
ISBN 978-3-96147-557-5.

Band 401: Christian P. J. Schwarzer
Kupfersintern als Fügetechnologie für Leistungselektronik
FAPS, xxvii u. 234 Seiten, 125 Bilder, 24 Tab. 2022.
ISBN 978-3-96147-566-7.

Band 402: Alexander Horn
Grundlegende Untersuchungen zur Gradierung der mechanischen Eigenschaften pressgehärteter Bauteile durch eine örtlich begrenzte Aufkohlung
LFT, xii u. 204 Seiten, 58 Bilder, 6 Tab. 2022.
ISBN 978-3-96147-568-1.

Band 403: Artur Klos
Werkstoff- und umformtechnische Bewertung von hochfesten Aluminiumblechwerkstoffen für den Karosseriebau
LFT, x u. 192 Seiten, 73 Bilder, 12 Tab. 2022.
ISBN 978-3-96147-572-8.

Band 404: Harald Schmid
Ganzheitliche Erarbeitung eines Prozessverständnisses von Tiefziehprozessen mit Ziehsicken auf Basis mechanischer und tribologischer Analysen
LFT, xiii u. 211 Seiten, 78 Bilder, 5 Tab. 2022.
ISBN 978-3-96147-577-3.

Band 405: Johannes Henneberg
Blechmassivumformung von Funktionsbauteilen aus Bandmaterial
LFT, viii u. 176 Seiten, 101 Bilder, 2 Tab. 2022.
ISBN 978-3-96147-579-7.

Band 406: Anton Schmailzl
Festigkeits- und zeitoptimierte Prozessführung beim quasi-simultanen Laser-Durchstrahlschweißen
LPT, xiii u. 157 Seiten, 84 Bilder, 7 Tab. 2022.
ISBN 978-3-96147-583-4.

Band 407: Alexander Wolf
Modellierung und Vorhersage menschlichen Interaktionsverhaltens zur Analyse der Mensch-Produkt Interaktion
KTmfk, x u. 207 Seiten, 69 Bilder, 10 Tab. 2022.
ISBN 978-3-96147-585-8.

Band 408: Tim Weikert
Modifikationen amorpher Kohlenstoffschichten zur Anpassung der Reibungsbedingungen und zur Erhöhung des Verschleißschutzes
KTmfk, xvii u. 258 Seiten, 91 Bilder, 9 Tab. 2022.
ISBN 978-3-96147-589-6.

Band 409: Stefan Götz
Frühzeitiges konstruktionsbegleitendes Toleranzmanagement
KTmfk, ix u. 276 Seiten, 127 Bilder, 13 Tab. 2022.
ISBN 978-3-96147-593-3.

Band 410: Markus Hubert
Einsatzpotenziale der Rotationsschneidtechnologie in der Verarbeitung von metallischen Funktionsfolien für mechatronische Produkte
FAPS, xviii u. 139 Seiten, 86 Bilder, 7 Tab. 2022.
ISBN 978-3-96147-603-9.

Band 411: Manfred Vogel
Grundlagenuntersuchungen und Erarbeitung einer Methodik zur Herstellung maßgeschneiderter Halbzeuge auf Basis eines neuartigen flexiblen Walzprozesses
LFT, ix u. 176 Seiten, 61 Bilder, 11 Tab. 2022.
ISBN 978-3-96147-605-3.

Band 412: Michael Weigelt
Multidimensionale Optionenanalyse alternativer Antriebskonzepte für die individuelle Langstreckenmobilität
FAPS, xv u. 222 Seiten, 89 Bilder, 38 Tab. 2022.
ISBN 978-3-96147-607-7.

Band 413: Frank Bodendorf
Machine Learning im Cost Engineering des Supply Managements
FAPS, xiii u. 165 Seiten, 75 Bilder, 13 Tab. 2023.
ISBN 978-3-96147-609-1.

Band 414: Maximilian Metzner
Planung und Simulation taktiler, intelligenter und kollaborativer Roboterfähigkeiten in der Montage
FAPS, xix u. 174 Seiten, 72 Bilder, 3 Tab. 2023.
ISBN 978-3-96147-611-4.

Band 415: Tina Buker
Ein Ansatz zur Reduktion produktinduzierter Nutzerstigmatisierung durch Förderung einer gleichermaßen gebrauchstauglichen wie emotionalen Produktgestalt
KTmfk, x u. 236 Seiten, 54 Bilder, 44 Tab. 2022.
ISBN 978-3-96147-613-8.

Band 416: Marlene Kuhn
Model-based Traceability System Development for Complex Manufacturing Applying Blockchain and Graphs
FAPS, xv u. 167 Seiten, 63 Bilder, 10 Tab. 2022.
ISBN 978-3-96147-615-2.

Band 417: Benjamin Lengenfelder
Remote photoacoustic sensing using speckle-analysis for biomedical imaging
LPT, xv u. 124 Seiten, 86 Bilder, 10 Tab. 2023.
ISBN 978-3-96147-617-6.

Band 418: Benjamin Pohrer
Analyse des Zusammenhangs zwischen dem tribochemischen Aufbau von Grenzschichten und der Ausbildung von White Etching Crack-Schäden
KTmfk, xv u. 258 Seiten, 103 Bilder, 10 Tab. 2023.
ISBN 978-3-96147- 621-3.

Band 419: Matthias Friedlein
Zuverlässigkeitsmethoden zur Beschleunigung von Qualifizierungsuntersuchungen für Steckkontakte
FAPS, xxv u. 162 Seiten, 98 Bilder, 7 Tab. 2023.
ISBN 978-3-96147-625-1.

Band 420: Thomas Stoll
Laser Powder Bed Fusion von Kupfer auf Aluminiumoxid-Keramiken
FAPS, xxvii u. 236 Seiten, 103 Bilder, 11 Tab. 2023.
ISBN 978-3-96147-631-2.

Band 421: Eric Eschner
Relation of Particle Motion and Process Zone Formation as a Basis for Sensing Approaches within PBF-LB/M
LPT, xiv u. 143 Seiten, 87 Bilder, 0 Tab. 2023.
ISBN 978-3-96147-633-6.

Band 422: Fanuel Mehari
Laser-induced Breakdown Spectroscopy (LIBS) as a diagnostics tool for biological tissue analysis.
LPT, xv u. 145 Seiten, 68 Bilder, 12 Tab. 2023.
ISBN 978-3-96147-641-1.

Band 423: Uwe Leicht
Ultraschallüberlagertes Umformen und Verstemmen von Stahlwerkstoffen
LFT, xi u. 165 Seiten, 65 Bilder, 6 Tab. 2023.
ISBN 978-3-96147-643-5.

Band 424: Thomas Braun
Potenzialanalyse der plasmabasierten, strukturierten Metallisierung thermoaktiver Oberflächen im industriellen Hausbau
FAPS, xvii u. 152 Seiten, 72 Bilder, 11 Tab. 2023.
ISBN 978-3-96147-653-4.

Band 425: Reinhardt Seidel
Modellbasierte Optimierung des Selektivwellenlötprozesses
FAPS, xxii u. 167 Seiten, 73 Bilder, 23 Tab. 2023.
ISBN: 978-3-96147-651-0.

Band 426: Matthias Lenzen
Maßgeschneiderte Werkstoffcharakterisierung für die numerische Auslegung von Blechumformprozessen
LFT, xi u. 187 Seiten, 77 Bilder, 13 Tab. 2023.
ISBN: 978-3-96147-663-3.

Band 427: Matthias Graser
Analyse lokaler Kurzzeitwärmebehandlungsmethoden zur Verbesserung des Umformverhaltens und der Bauteileigenschaften von Aluminiumstrangpresshohlprofilen
LFT, xi u. 169 Seiten, 81 Bilder, 1 Tab. 2023.
ISBN: 978-3-96147-666-4.

Band 428: Markus Lieret
Sicheres autonomes Flugrobotersystem für den Einsatz im Produktions- und Logistikumfeld
FAPS, xix u. 198 Seiten, 54 Bilder, 7 Tab. 2023.
ISBN 978-3-96147-668-8.

Band 429: Petar Vukovic
Simulation komplexer Kommunikationssysteme in der Fertigungsautomatisierung
FAPS, xiv u. 163 Seiten, 57 Bilder, 21 Tab. 2023.
ISBN 978-3-96147-673-2.

Band 430: Fabian Knieps
Finite Elemente Simulation dünnster Verpackungsstähle: Entwicklung einer geeigneten Charakterisierungs- und Validierungsstrategie
LFT, xix, 189 Seiten, 122 Bilder, 17 Tab. 2023.
ISBN 978-3-96147-689-3

Band 431: Julian Seßner
Multimodale Bildsegmentierung gering strukturierter Umgebungen für die Navigation am Beispiel eines Assistenzsystems für sehbeeinträchtigte Personen
FAPS, xxv, 203 Seiten, 57 Bilder, 25 Tab. 2023.
ISBN 978-3-96147-697-8

Band 432: Benjamin Samuel Lutz
Smart Manufacturing System for Process Optimization Regarding Deviations among Material Batches
FAPS, xix, 208 Seiten, 77 Bilder, 14 Tab. 2023.
ISBN 978-3-96147-703-6

Band 433: Michael Jüttner
Bewertung von Kantenpressungen auf Basis von Simulationen mehrfach überrollter elasto-plastischer Kontakte
KTmfk, xii, 162 Seiten, 59 Bilder, 7 Tab. 2024.
ISBN 978-3-96147-713-5.

Band 434: Sebastian Wiesenmayer
Untersuchungen zur Stoffflusssteuerung beim Fügen durch Umformen von hochfesten Aluminiumlegierungen mittels lokaler Kurzzeitwärmebehandlung
LFT, xii u. 197 Seiten, 81 Bilder, 19 Tab. 2024.
ISBN 978-3-96147-715-9.

Band 435: Clara-Maria Kuball
Grundlegende Untersuchungen zur umformtechnischen Herstellung von Halbhohlstanznieten aus hochverfestigenden Werkstoffen
LFT, viii u. 180 Seiten, 64 Bilder, 13 Tab. 2024.
ISBN 978-3-96147-717-3.

Band 436: Martin Roth
Sampling-based Tolerance-Cost Optimization: The Key to Optimal Tolerance Allocation
KTmfk, xxxvii u. 337 Seiten, 97 Bilder, 56 Tab. 2024.
ISBN 978-3-96147-719-7.

Band 437: Stephan Schirdewahn
Verbesserung des tribologischen Einsatzverhaltens im Presshärteprozess durch Verwendung maßgeschneiderter laserimplantierter Werkzeuge
LFT, viii u. 177 Seiten, 63 Bilder, 7 Tab. 2024.
ISBN 978-3-96147-721-0.

Band 438: Andreas Rohrmoser
Erarbeitung eines grundlegenden Verständnisses zum Fließpressen betriebsangepasster Verzahnungen für den Einsatz in der Materialpaarung Metall-Kunststoff
LFT, x u. 166 Seiten, 94 Bilder, 6 Tab. 2024.
ISBN: 978-3-96147-723-4.

Band 439: Andreas Selmaier
DMAICS-Zyklus zur Digitalisierung in produzierenden Unternehmen
FAPS, xv u. 185 Seiten, 73 Bilder, 19 Tab. 2024.
ISBN: 978-3-96147-733-3.

Band 440: Thomas Kistner
Entwicklung von Modellen der Oberflächenform für die Messunsicherheitsbestimmung von taktilen Koordinatenmessungen durch Simulation
FMT, xiii u. 130 Seiten, 64 Bilder, 9 Tab. 2024.
ISBN 978-3-96147-735-7.

Band 441: Fabian Dworschak
Selbstverstärkendes Lernen als Beitrag zur Automatisierung der Anpassungskonstruktion
KTmfk, x, 205 Seiten, 87 Bilder, 16 Tab. 2024
ISBN 978-3-96147-739-5

Abstract

Automation describes the transfer of tasks from humans to machines. In the context of adaptation design, these tasks involve defining the product´s characteristics to meet new requirements while maintaining the same principal solution. They are particularly challenging if characteristics simultaneously influence competing properties. Here, the best possible compromise must be found. Depending on the context of the adaptation task, they can be automated by optimisation algorithms or by data-driven approaches. However, data-driven approaches require that the underlying database contains all relevant information for recognising the relationships between characteristics and resulting properties. For their part, optimisation algorithms must be brought to convergence whenever the boundary conditions change. For the intersection of adaptation tasks, for which the boundary conditions change regularly due to changes in requirements and at the same time no sufficient database is available, this work presents reinforcement learning as an extension of the current automation possibilities. Reinforcement learning is described as the learning of an agent based on experience with an environment. Here, the environment represents the tasks of the product developers in the context of adaptation design. In which, the agent is trained to take over tasks by adapting characteristics and receiving the product properties as feedback in return. This work does not to develop a new algorithm for reinforcement learning, but takes reinforcement learning into product development, defines the role of the product developers in its use and to research the transferability of the experience gained from one adaptation task to a subsequent one.

Die Automatisierung beschreibt die Übergabe von Aufgaben vom Menschen an eine Maschine. Die Aufgaben der Anpassungskonstruktion umfassen das Festlegen der Merkmalsausprägungen zum Erfüllen neuer Anforderungen bei gleichbleibender Wirkstruktur. Sie sind herausfordernd, wenn Merkmale gleichzeitig konkurrierende Eigenschaften beeinflussen. Je nach Kontext können sie durch Optimierungsalgorithmen oder datengetriebene Ansätze automatisiert werden. Datengetriebene Ansätze setzen eine Datenbasis voraus, die alle relevanten Informationen für das Erkennen der Zusammenhänge zwischen Merkmalen und resultierenden Eigenschaften enthält. Optimierungsalgorithmen müssen bei einer Änderung der Randbedingungen erneut zur Konvergenz gebracht werden. Für die Schnittmenge der Anpassungsaufgaben, für die sich die Randbedingungen durch Anforderungsänderungen regelmäßig ändern und gleichzeitig keine ausreichende Datenbasis vorliegt, stellt diese Arbeit das Selbstverstärkende Lernen vor. Es beschreibt das Lernen eines Agenten auf Basis von Erfahrungen mit einer Umgebung. Hier wird diese durch die verschiedenen Konstruktionsumgebungen der Produktentwickelnden im Rahmen der Anpassungskonstruktion repräsentiert. Der Agent passt die Merkmale an und erhält die Produkteigenschaften als Feedback. Ziel der Arbeit ist nicht das Entwickeln eines neuen Algorithmus für Selbstverstärkendes Lernen, sondern das Erforschen der Voraussetzungen für Selbstverstärkendes Lernen in der Produktentwicklung, das Definieren der Rolle der Produktentwickelnden bei dessen Einsatz sowie das Erforschen der Übertragbarkeit der gesammelten Erfahrungen von einer Anpassungsaufgabe auf nachfolgende.

ISBN 978-3-96147-739-5

FAU UNIVERSITY PRESS 2024