Round-Trip-Engineering im modellbasierten Ingenieurentwurf

Von der Fakultät für Luft- und Raumfahrttechnik und Geodäsie
der Universität Stuttgart zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
Dominik Joachim Walter Schopper
geboren in
Stuttgart-Bad Cannstatt

Hauptberichter:	PD Dr.-Ing. Stephan Rudolph
Erster Mitberichter:	Prof. Dr.-Ing. Oliver Riedel
Zweiter Mitberichter:	Prof. Dr.-Ing. Andreas Strohmayer
Tag der mündlichen Prüfung:	19.12.2023

Institut für Flugzeugbau
Universität Stuttgart
2023

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Arbeitsgruppe *Entwurfstheorie und Ähnlichkeitsmechanik* („π-Gruppe“) zunächst am Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen (ISD) und nach dem Umzug der Arbeitsgruppe später am Institut für Flugzeugbau (IFB) an der Universität Stuttgart.

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Anfertigung dieser Dissertation mit Rat und Tat unterstützt haben und auf diese Weise zum Gelingen dieser Arbeit beigetragen haben. Einigen bin ich dabei zu besonderem Dank verpflichtet:

Zuallererst möchte ich meinem Doktorvater, Herrn PD Dr.-Ing. Stephan Rudolph, für das mir entgegen gebrachte Vertrauen und sein Engagement während der Anfertigung dieser Arbeit danken. Die zahlreichen, oftmals weit über den Tellerrand der Arbeit hinaus reichenden Gespräche habe ich immer als Motivation und Inspiration empfunden. Auch dein stets offenes Ohr für Probleme abseits der Arbeit wird mir immer in besonderer Erinnerung bleiben.

Des Weiteren möchte mich bei Herrn Prof. Dr.-Ing. Oliver Riedel sowie Herrn Prof. Dr.-Ing. Andreas Strohmayer für die freundliche Übernahme des Mitberichts bedanken.

Ein ganz besonderer Dank gilt auch allen Kolleginnen und Kollegen aus der Arbeitsgruppe, die durch ihre unermüdliche Hilfe, wissenschaftliche und moralische Unterstützung sowie nicht zu vergessen auch Heiterkeit die fordernde und arbeitsreiche Promotionszeit zu einer wundervollen und rückblickend wahrscheinlich zur besten Zeit meines Lebens gemacht haben.

Ebenso möchte ich mich bei meinen Kolleginnen und Kollegen am IFB bedanken, die uns nach unserem Umzug herzlich am Institut aufgenommen haben. An die gemeinsame Zeit erinnere ich mich immer gerne zurück. Dies gilt daneben auch allen Kolleginnen und Kollegen des Forschungsprojekts „Digitaler Produktlebenszyklus (DiP)“ am Zentrum für angewandte Forschung an Hochschulen (ZAFH). Unsere gemeinsame Forschung im Projekt hat den Grundstein für die Anfertigung dieser Arbeit gelegt. Die zahlreichen Treffen habe ich immer als wissenschaftlich bereichernd, kollegial und freundschaftlich wahrgenommen.

Meinen Eltern und meiner gesamten Familie möchte ich von Herzen für ihre Unterstützung und Ermutigung während meines gesamten Studiums sowie insbesondere während der Promotionszeit danken. Euer unerschütterlicher Glaube an mich hat mir die Kraft und den Mut zur Anfertigung dieser Arbeit gegeben. Auch für alles was ihr sonst noch für mich getan habt, bin ich euch zutiefst dankbar und werde euch das nie vergessen.

Zu guter Letzt möchte ich meinen ganz besonderen Dank meiner Frau Claudia aussprechen. In all den Jahren haben wir jede Herausforderung, ob inner- oder außeruniversitär, *gemeinsam* gemeistert. Ich glaube nicht, dass vielen Menschen das Glück vergönnt ist, einen Menschen an ihrer Seite zu haben, der Arbeitskollege, bester Freund und Partner zugleich sein kann.

Stuttgart im Dezember 2023

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abk.	Bedeutung
ADT	Autogenetic Design Theory
BFO	Basic Formal Ontology
BOM	Bill of Materials
BREP	Boundary Representation Methode
CAD	Computer Aided Design
CAS	Computeralgebrasystem
CFD	Computational Fluid Dynamics
CIM	Computation Independent Model
CSG	Constructive Solid Geometry
CST	Classfunction Shapefunction Transformation
DC43	Design Cockpit 43®
DIN	Deutsche Industrienorm
DRM	Design Research Methodology
DSL	Domain Specific Language
EKP	Entwicklungs- und Konstruktionsprozess
FCA	Formal Concept Analysis
FDR	fest-diskrete Randbedingungen
FE	Forward-Engineering
FEM	Finite Elemente Methode
FF	Forschungsfrage
GBES	Graphenbasierte Entwurfssprache
GDA	Generative Design Approach
IILS mbH	Ingenieurgesellschaft für intelligente Lösungen und Systeme mbH
INCOSE	International Council on Systems Engineering
ISD	Institut für Statik und Dyanmik der Luft- und Raumfahrtkonstruktionen
ISO	Internationale Organisation für Normung
KBE	Knowledge-based Engineering
KDR	kontinuierlich-diskrete Randbedingungen
KI	Künstliche Intelligenz
KIF	Knowledge Interchange Format

LHS	Left-Hand Side
LPG	Lösungspfadgenerator
M2M	Modell-zu-Modell
M2T	Modell-zu-Text
MBS	Multibody Simulation
MBSE	Model-based Systems Engineering
MDA	Model-driven Architecture
MGS	mathematisches Gleichungssystem
MIT	Massachusetts Institute of Technology
NDIA	National Defense Industrial Association
ODM	Ontology Definition Metamodel
OMG	Object Management Group
OOP	objektorientierte Programmierung
OWL	Web Ontology Language
PEP	Produktentstehungsprozess
PGE	Product Generation Engineering
PIM	Platform Independent Model
PSM	Platform Specific Model
RDF	Resource Description Framework
RE	Reverse-Engineering
REE	Re-Engineering
RHS	Right-Hand Side
RTE	Round-Trip-Engineering
RTE_S	Round-Trip-Engineering in der Softwareentwicklung
SCR	Selective Catalytic Reduction
SE	Systems Engineering
SoS	System of Systems
STEP	Standard for the Exchange of Product Model Data
SVV	systematische Vorwärtsvariation
T2M	Text-zu-Modell
TK	Themenkomplex
TRL	Technology Readiness Level
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieuere
VDR	variabel-diskrete Randbedingungen

Kurzfassung

In Rahmen der vorliegenden Arbeit wird das ursprünglich aus der Informatik stammende Konzept des *Round-Trip-Engineerings* als Kombination der drei Vorgänge des Forward-, Reverse- und Re-Engineering konzeptuell auf den ingenieurwissenschaftlichen Produktentwurf übertragen und mittels graphenbasierter Entwurfssprachen als Framework implementiert.

Das Forward-Engineering tritt im Ingenieurentwurf bei klassischen Entwurfsaufgaben wie der Neu- oder Variantenkonstruktion zutage. In dieser Arbeit wird ein automatisiertes Forward-Engineering auf Basis einer domänenübergreifenden Wissensbasis definiert, bei dem aus Anforderungen durch eine *individuelle* Modelltransformationssequenz ein konkretes Modell, das das gewünschte Produkt repräsentiert, generiert wird. Zur einfacheren Definition der angesprochenen Wissensbasen wird in der Arbeit gezeigt, wie die formale Begriffsanalyse genutzt werden kann, um *objektorientierte Universalontologien* abzuleiten. Dies wird exemplarisch für die Domänen der Funktion, der Geometrie, des Materials, der Herstellung und der physikalischen Lasten – als wesentliche Einflussbereiche im mechanischen Produktentwurf – vorgestellt.

Das Reverse-Engineering folgt aus dem Kontext einer Konstruktionsaufgabe, deren Zielprodukt zwar in Form eines Entwurfsartefakts – z. B. ein CAD-Modell – bekannt ist, aber keine Modelle im Sinne eines modellbasierten Entwurfs existieren. Die Intention des Reverse-Engineering ist eine Rekonstruktion dieser abstrakten Modelle aus den vorhandenen Modellartefakten.

Hervorzuheben ist dabei die gewonnene Erkenntnis, dass hierbei ein zugrundeliegendes Mustererkennungsproblem gelöst und im Sinne der Vollständigkeit eine in der Model-driven Architecture bisher nicht definierte Text-zu-Modell-Transformation eingeführt werden muss. Für die Geometriedomäne wird detailliert beschrieben, wie ein solches Reverse-Engineering *automatisiert* modelliert werden kann. Diese Vorgehensweise wird durch eine Implementierung praktisch umgesetzt und anhand des Beispiels eines Satellitengehäuses validiert.

Das Re-Engineering wird im Kontext dieser Arbeit als Umkonstruktion im weitesten Sinne verstanden, wozu in der klassischen Konstruktionsphilosophie die Anpassungs- oder auch die Variantenkonstruktion zählen würden. In diesem Zusammenhang ist die Definition graphenbasierter *Spezifikations- und Abstraktionscasts* als Übertragung und Erweiterung des klassischen Casts aus der Informatik hervorzuheben. Des Weiteren ist in diesem Zuge die Definition einer Sammlung fertigungstechnischer und bauteiltheoretischer Prinziplösungen in einer Bibliothek generischer Graphentransformationen als Übertragung eines objektorientierten Entwurfsmusters auf das Ingenieurwesen zu nennen. Diese *graphenbasierten Entwurfsmuster* werden allgemein beschrieben und am Beispiel der Geometriedomäne in Form einer generischen Stoßdatenbank für Übergangsgeometrien konkret implementiert. Die Funktionsweise wird anhand eines Anwendungsbeispiels zu Strukturverbindungen dünnwandiger Bauteile validiert.

Insgesamt wird in der vorliegenden Arbeit dargelegt, wie für ein und dieselbe Konstruktionsaufgabe automatisiert Varianten bezüglich der bereits erwähnten, im Entwurf mechanischer Bauteile wichtigen Domänen generiert, bewertet und somit die Annäherung an ein globales

Optimum über die verschiedenen Domänen hinweg erzielt werden kann. Die Automatisierung wird dabei als Schlüssel zur Lösung des „Paradoxon der Konstruktion“ beschrieben, da nur mit ihrer Hilfe eine große Detailtiefe der Varianten in kürzest möglicher Zeit erreicht werden kann.

Darüber hinaus wird in dieser Arbeit ein Weg aufgezeigt, wie basierend auf der Vorstellung des modellbasierten Entwurfs alle identifizierten Teilprozesse des Round-Trip-Engineerings in ein und demselben Vorgang automatisiert werden können. Die Automatisierung basiert im Wesentlichen auf Modelltransformationen, die in einer intelligenten Formulierung den Weg für eine dynamische Reaktion auf geänderte Anforderungen vorbereiten. Hierfür werden zunächst verschiedene Arten von Randbedingungen identifiziert und anschließend ein generisches Aktivitätenschema vorgestellt, das in der Lage ist, die geforderte Flexibilität bei der Ausführung zu realisieren. Die Regeln müssen für diesen Zweck in einer automatisiert abgeleiteten und auf den Kontext abgestimmten Reihenfolge aufgerufen werden. Dies wird in der vorliegenden Arbeit als *dynamische Regelausführung* bezeichnet. In deren prototypischen Implementierung auf Basis graphenbasierter Entwurfssprachen spielen ineinandergeschachtelte Iterationsschleifen eine wesentliche Rolle, die als Anwendung eines Graphenalgorithmus zur Auflösung der Randbedingungen und Constraints zur Laufzeit der Entwurfssprache interpretiert werden können.

Für die Einbringung sogenannter *geometriewirksamer Randbedingungen* bei einer Konstruktionsaufgabe mit vorgegebener äußerer Form wird eine innovative Methode vorgestellt, wie sich auf die äußere Grundform auswirkende Randbedingungen durch eine BOOL’sche Schnittdefinition an der gewünschten Position im aktuellen Entwurfskontext berücksichtigen lassen. Durch einen oder mehrere solcher Schnitte werden sogenannte Konstruktionszonen definiert und einer musterbasierten Geometriemanipulation zugänglich gemacht. Dieser Vorgang tritt zum Beispiel bei der automatisierten Definition von Übergangs- oder Verstärkungsgeometrien auf und wird als Abschluss dieser Arbeit anhand des Entwurfs von von Hubschrauberstrukturelementen auch an einem industriell relevanten Anwendungsbeispiel aufgezeigt und validiert.

Abstract

In the context of the present thesis, the concept of *round-trip engineering*, which originally originated in computer science, as a combination of the three processes of forward, reverse, and re-engineering, is conceptually transferred to engineering product design and implemented as a framework using so-called graph-based design languages.

Forward engineering appears in engineering design in classical design tasks such as new or variant design. In this work, an automated forward engineering based on a domain-independent knowledge base is defined, in which requirements are transformed by an *individual* model transformation sequence into a concrete model representing the desired product. To simplify the definition of the addressed knowledge bases, the paper showed how formal conceptual analysis can be used to derive *object-oriented universal ontologies*. This is exemplified for the domains of function, geometry, material, manufacturing, and physical loads – as major areas of influence in mechanical product design.

Reverse engineering follows from the context of a design task whose target product is known only in the form of a design artifact – e.g. a CAD model – but no models exist in the sense of a model-based design. The intention of reverse engineering is to reconstruct these abstract models from the existing model artifacts. In this work it could be shown that this involves solving an underlying pattern recognition problem and the introduction of a text-to-model transformation not previously defined in model-driven architecture. For the geometry domain, it is described in detail how such reverse engineering can be patterned *automatically*. The implementation of the procedure is validated on the example of a satellite housing.

Re-engineering is described in the context of this thesis as redesign in the broadest sense, which in classical design philosophy would include adaptation or variant design. In this context, the definition of graph-based *specification and abstraction casts* as a transfer and extension of the classical casts from computer science is to be emphasized. Furthermore, the definition of a collection of manufacturing and building part theoretical principle solutions in a library of generic graph transformations as a transfer of an object-oriented design pattern to engineering should be mentioned in this context. These *graph-based design patterns* are described in general and implemented concretely on the example of the geometry domain in the form of a generic joint database for transition geometries. The functionality is validated using an application example on structural joints of thin-walled components.

All together, the present work demonstrates how, for one and the same design task, variants with respect to the aforementioned domains important in the design of mechanical components can be automatically generated and evaluated. Consequently the approximation of a global optimum across the different domains can be achieved. Automation is described here as the key to solving the „paradox of design“, since only with its help a great depth of detail of the variants can be achieved in the shortest possible time.

Furthermore, this thesis shows a way how, based on the notion of model-based design, all

identified sub-processes of round-trip engineering can be automated in one and the same process. The automation is essentially based on model transformations, which in an intelligent formulation pave the way for a dynamic response to changing requirements. For this purpose, different types of constraints are first identified and then a generic activity schema capable of realizing the required flexibility in execution is presented (*dynamic rule execution*). In their prototypical implementation based on graph-based design languages, nested iteration loops play an essential role, which can be interpreted as the application of a graph algorithm to resolve the boundary conditions and constraints at runtime of the design language.

For the introduction of so-called *geometry-effective boundary conditions* in a design task with a given outer shape, an innovative way is presented to introduce boundary conditions affecting the basic external shape into the current design context by means of a BOOL's section definition at the desired position. So-called construction zones are defined by one or more such sections and made accessible to pattern-based geometry manipulation. This process occurs, for example, in the automated definition of transition or reinforcement geometries and, as a conclusion to this work, is also demonstrated and validated in an industrially relevant application example using the design of helicopter structural elements.

1 Thematische Einführung

> *Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.“*
>
> **— Albert Einstein**
> (deutscher Physiker und Nobelpreisträger)

Kapitelübersicht

1.1 Motivation

Zu den Megatrends unserer Zeit zählen Nachhaltigkeit, Globalisierung[1], digitale Transformation und stark ansteigende Systemkomplexität [INCOSE, 2021, S. 4–9]. Angesichts dessen wird es für Unternehmen zusehends schwieriger, auf dem Markt langfristig kompetitiv zu bleiben. Sie reagieren mit einer Verkürzung der Innovations- und Produktlebenszyklen ihrer Produkte und erhoffen sich durch eine gesteigerte Variantenanzahl kundenindividueller und damit attraktiver aufgestellt zu sein [Ehrlenspiel und Meerkamm, 2013, S. 227, 255; VDI 5610-2, S. 2].

Diese Strategien bringen es mit sich, dass Produkte heutzutage schneller, häufiger und umfassender modifiziert, umkonstruiert oder gänzlich neu entworfen werden müssen als früher. Trotz des dadurch wachsenden Zeit- und Innovationsdrucks sollen die Produktqualität weiterhin hoch und die Entwicklungs- und Herstellungskosten möglichst niedrig sein. Für eine gute Marktpositionierung müssen die Unternehmen somit einen stetig hohen Innovations- und Qualitätsgrad bei geringen Kosten und in minimaler Entwicklungszeit erreichen. [VDI 2221, S. 24]

In den wenigsten Fällen ist der hierfür notwendige Umsetzungsweg a priori determinierbar. Besonders bei erhöhtem Innovationsgrad bedeutet Entwerfen ein Vorantasten vom rein Abstrakten hin zum Konkreten, wobei dies oftmals ein Zurückkehren zum Ausgangspunkt und erneutes Konkretisieren beinhaltet. Dieses iterative Vorgehen liegt im Wesen von „ill-structured“ [Dym und Brown, 2012] bzw. „wicked problems“ [Rittel und Webber, 1973], d. h. Problemstellungen, deren Anforderungen nicht vollständig bekannt und deren Lösung nicht eindeutig ist.

Gerade die frühe Phase des Entwurfsprozesses ist von dieser Unsicherheit unvollständigen

[1] Zum Zeitpunkt des Verfassens dieses Kapitels – nach einer über zweijährigen Phase der Covid 19-Pandemie und dem Beginn einer kriegerischen Auseinandersetzung in der Ukraine – ist das Voranschreiten dieses Megatrends der vergangenen drei Dekaden derzeit unvorhersagbar.

Wissens geprägt und trägt durch die in ihr getroffenen Architekturentscheidungen hierzu in besonderen Maße bei. Durch die Definition von Funktions- und Produktstruktur und deren Kopplung wird der Grundstein für eine mögliche Innovation im Sinne einer völlig neuen Produktfunktion oder einer Funktionserfüllung in bisher unbekannter Qualität [Grabowski et al., 1996] gelegt. Heutzutage wird eine solche neuartige Funktionalität oftmals durch Software, Elektronik oder die Kombination von Lösungen aus unterschiedlichen Ingenieursdomänen erreicht [Pahl und Beitz, 2013, S. 5–6]. Vor diesem Hintergrund lässt sich erklären, warum viele innovative Produkte von interdisziplinärer Natur sind.

In der Konsequenz muss Wissen aus verschiedenen Domänen bereitgestellt werden, das anschließend untereinander verknüpft und bei Bedarf an neue Anforderungen angepasst werden muss [Leemhuis, 2005, S. 1]. Das Dilemma dabei ist, dass das Produktwissen erst im Laufe des Entwurfes anwächst, sodass gerade in der Konzeptphase, in der Topologie, Parameter und Eigenschaften mit der größten Auswirkung auf das Produkt – und dadurch die wesentlichen Kostenfaktoren – festgelegt werden, eine unzureichende Informationsbasis vorherrscht.

Dieser Sachverhalt ist in der Literatur in nuancierten Abwandlungen unter dem Begriff „Paradoxon der Konstruktion“[2] bekannt, welcher in Abbildung 1.1 illustriert ist.

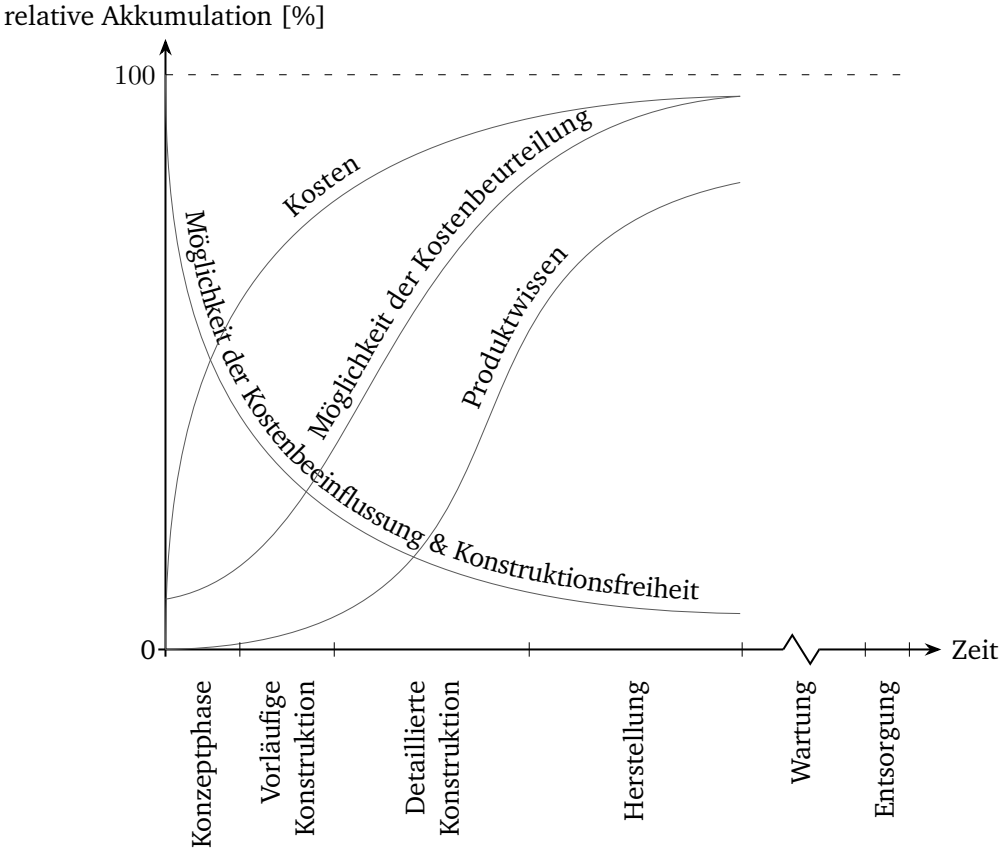


Abbildung 1.1: Kosten, Produktwissen, Möglichkeit der Kostenbeurteilung und -beeinflussung und Konstruktionsfreiheit akkumuliert über der Zeit – „Paradoxon der Konstruktion“ angelehnt an [Mavris et al., 1998, S. 3] und [VDI 2235, S. 5]

[2]Die Begriffsurheberschaft konnte nicht ermittelt werden. Er wird in einigen Werken unreferenziert verwendet (z. B. [VDI 2235, S. 5], [Ehrlenspiel und Meerkamm, 2013, S. 668] oder [Mavris et al., 1998, S. 3]).

Ein weiteres Paradoxon besteht in der frühen Phase der Konstruktion in der Diskrepanz zwischen der Möglichkeit der Kostenbeurteilung, die sehr niedrig ist, und der möglichen Kostenbeeinflussung, die zu diesem Zeitpunkt noch hoch ist. Viele Forschungsarbeiten widmen sich der Abmilderung der Auswirkungen dieser gegenläufigen Kurven und ihren Implikationen.

Ein Ansatz besteht darin, das Produktwissen früher und genauer bereitzustellen, beispielsweise durch die Anwendung von Knowledge-based Engineering (KBE)-Applikationen [Mavris et al., 1998, S. 3]. Eine weitere Möglichkeit stellt die Verfolgung mehrerer Produktvarianten im Rahmen einer Lösungsraumexploration (z. B. [Navinchandra, 1991], [Kang et al., 2011] oder [Lüdtke, 2016]) oder über einen längeren Zeitraum, z. B. mittels eines sogenannten „Set-based Concurrent Engineering“ [Sobek et al., 1999] dar. Auch ein agiler Entwicklungs- und Konstruktionsprozess, bei dem alle Phasen schnell, allerdings nicht in voller Tiefe durchlaufen werden, wird in der Literatur vorgeschlagen [Pahl und Beitz, 2013, S. 808–809].

Aus Sicht des Autors der vorliegenden Arbeit kann das Paradoxon der Konstruktion nur durch eine radikale Verkürzung der Konstruktionsdauer und die Beurteilung einer Konstruktion auf Basis einer vollständigen Produktbeschreibung im Sinne eines digitalen Zwillings am Ende der Detailkonstruktionsphase vollständig aufgelöst werden. Diese Aussage wird untermauert durch die Feststellung von RUDOLPH zur systematischen Bewertung von Konstruktionen, nachdem ein Beurteilungs- bzw. Bewertungsproblem genau dann gelöst ist, wenn eine vollständige Beschreibung inklusive aller Konstruktionsparameter vorliegt [Rudolph, 1995, S. 91–92].

Die einzige Möglichkeit, eine vollständige Produktbeschreibung in kürzestmöglicher Zeit zu erreichen, liegt in der durchgängigen Automatisierung des Konstruktionsvorgangs. Dadurch würde der Zeitaufwand zur Produktgenerierung – bzw. zur Generierung des zugehörigen digitalen Zwillings – auf die Laufzeit der Algorithmen und etwaiger Simulationen zusammenschrumpfen. Dies würde den Konstruktionsvorgang im Vergleich zum Status Quo einerseits massiv beschleunigen, andererseits aber eine qualitative und quantitative Konstruktionsevaluation am auskonstruierten Produkt ermöglichen. Die Zeitersparnis würde es darüber hinaus ermöglichen, viele unterschiedliche Varianten für eine Entwurfsraumexploration zu generieren. Aufgrund der Detailtiefe wäre der aufgespannte Lösungsraum gleichzeitig der adäquate Bewertungsraum, um ein domänenübergreifendes, globales Konstruktionsoptimum finden zu können.

In der vorliegenden Arbeit werden bestehende Konzepte aus der Informatik und der Ingenieurwissenschaft zusammengeführt, um zur Realisierung eben dieser Vision auf einem eingegrenzten thematischen Bereich einen theoretischen sowie praktischen Beitrag zu leisten.

1.2 Problemstellung und Zielsetzung

In der Literatur wurde die vollständige Automatisierung des Produktentwurfs bisher als reine Utopie wahrgenommen, die genauso unmöglich zu erreichen sei, wie die Quadratur des Kreises mit Zirkel und Lineal[3]. Wie sollte eine lose Menge von teilweise unscharfen Anforderungen maschinell in ein funktionsfähiges Produkt überführt werden können? Nicht einmal ein Beweis

[3] Die Unmöglichkeit der Lösung dieses Problems geht auf den Beweis der Transzendenz der Kreiszahl π durch LINDEMANN und WEIERSTRASS zurück [von Lindemann, 1882, S. 223].

der Lösungsunmöglichkeit, wie für die Kreisquadratur ableitbar, scheint auf dieser Grundlage führbar zu sein. Bereits Mitte der 1970er Jahre untersucht FRANKE in seiner Dissertation die Algorithmisierbarkeit des Konstruktionsprozesses und konstatiert, dass „*es keinen Algorithmus geben kann, der für alle Konstruktionsaufgaben eine Lösung (...) erzeugt*“ [Franke, 1976, S. 181].

Doch diese Aussage sollte nicht direkt als Showstopper verstanden werden. In der Technikgeschichte tauchen immer wieder Beispiele für die Überwindung scheinbar unlösbarer Probleme auf. Vor 100 Jahren hielt man es für unmöglich, dass einmal ein Mensch seinen Fuß auf den Mond setzen würde. Genauso unvorstellbar wären vor 50 Jahren selbstfahrende Automobile gewesen. Vor diesem Hintergrund sollte man zumindest den Gedanken zulassen, dass in Zukunft vielleicht doch eine technologische Umsetzung des vollautomatischen Produktentwurfs entwickelt werden könnte. Auch FRANKE konstatiert, dass eine Algorithmisierbarkeit nicht gänzlich ausgeschlossen sei, jedoch „*derzeit nicht möglich ist, da bisher keine logisch-inhaltliche Erfassung vieler konstruktiver Parameter vorliegt*“ [Franke, 1976, S. 181].

Die Problemstellung sollte demnach eingeschränkt werden, um einer Automatisierung leichter zugänglich gemacht zu werden. Dies deckt sich mit den Erkenntnissen von KOLLER, der in diesem Zusammenhang feststellt, dass „*Konstruktionsprozesse für erstmals zu konstruierende Produkte [. . .] von Menschen erst erdacht werden [müssen], ehe man diese beschreiben kann. Erst dann kann man diese auch programmieren und automatisieren. Nur für Produkte, welche immer wieder variiert werden müssen [. . .] ist es wirtschaftlich sinnvoll, deren Konstruktionsprozesse [. . .] zu automatisieren*“ [Koller, 1998, S. 463]. Er unterscheidet diesbezüglich in der Folge *originäre* und *nachvollzogene* (sekundäre) Arten von Konstruktionen [Koller, 1998, S. 464].

Im Rahmen der vorliegenden Arbeit wird daher eine Auswahl an Parametern im Entwurfsprozess als Freiheitsgrade angesehen, während die restlichen als feste Eingangsgrößen determiniert sind. Im Sinne KOLLERS werden somit vornehmlich sekundäre Konstruktionen betrachtet. Der Anspruch dieser Arbeit ist es nicht, ein industriell anwendbares Konstruktionstool zu beschreiben, das in der Lage ist, alle menschlichen Tätigkeiten bei sämtlichen Konstruktionsprozessen zu ersetzen. Es sollen vielmehr Modellierungstechniken, Modelle und Prozesse sowie sinnvoll automatisierbare Teilbereiche identifiziert werden, die den Grundstein dafür legen können.

In der Informatik oder genauer in der Softwareentwicklung haben sich bereits Techniken etabliert, die einen automatischen Softwareentwurf für einen eingeschränkten Anwendungsbereich ermöglichen. Da ein genereller Trend zu beobachten ist, dass sich durch die immer öfter durch Software ausgeführte Funktionserfüllung und die Nutzung digitaler Prototypen die Domänen des Ingenieurentwurfs und der Informatik immer weiter annähern [Pahl und Beitz, 2013, S. 809], soll in dieser Arbeit nach einer Möglichkeit der Übertragung etablierter softwaretechnischer Prinzipien auf den Ingenieurentwurf gesucht werden.

Daneben existiert aber auch eine Reihe fundamentaler Unterschiede, die die Automatisierung im Ingenieurentwurf zu einer deutlich komplexeren Operation werden lässt als in der Informatik und die in der gesuchten Übertragung softwaretechnischer Prinzipien auf den Ingenieurentwurf zwingend berücksichtigt werden muss. Hier ist die reale – nicht nur virtuelle – Funktionsfähigkeit des Produkts unter Wechselwirkungen mit der Umgebung und den physikalischen Einflüssen der größte Faktor. Auch die Gestaltungsphase, bei der die funktionale

und logische Struktur in eine geometrische Form überführt wird, hat kein direktes Äquivalent im Softwareentwicklungsprozess. Die wichtigsten Einflussgebiete und deren Schnittmengen, die die vorliegende Arbeit tangieren, sind in Abbildung 1.2 in Form eines Venn-Diagramms[4] grafisch dargestellt.

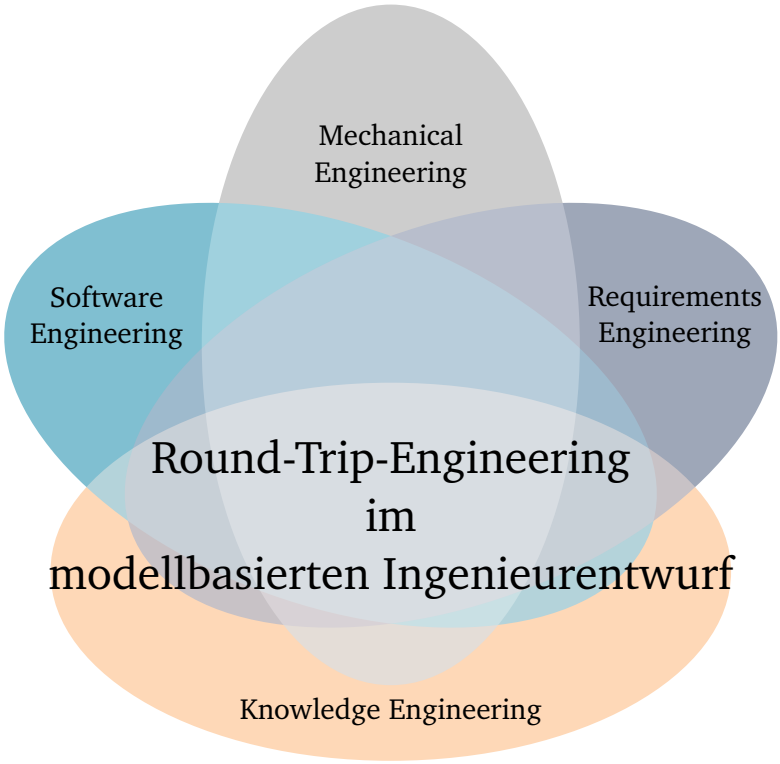


Abbildung 1.2: Für die vorliegende Arbeit „Round-Trip-Engineering im modellbasierten Ingenieurentwurf“ relevante Wissensgebiete und deren Schnittmengen

In Bezug auf Abbildung 1.2 soll im Bereich der Softwareentwicklung (Software Engineering) der Fokus auf Prinzipien der Objektorientierung, Entwurfsmuster und der modellbasierten Entwicklung liegen. Bei der Konstruktion mechanischer Bauteile (Mechanical Engineering) sollen die aus Sicht des Autors der vorliegenden Arbeit wichtigsten Teildomänen der Funktion, der Form, des Materials, der Herstellung und der physikalische Lasten berücksichtigt werden. Dem Gebiet der Anforderungen (Requirements Engineering) soll eine spezielle Bedeutung zukommen. Dabei sollen Anforderungen im Sinne des Entwurfs als vieldimensionales System als Randbedingung an die Konstruktion aufgefasst werden, die entweder variabel oder konkret vorgegeben werden können sollen. Die Lösungssequenz, in der das Produkt generiert wird, soll sich dabei dynamisch an die gewählten Randbedingungen anpassen können. Der letzte betrachtete Bereich befasst sich damit, wie Wissen formalisiert und maschinell lesbar aufbereitet werden kann (Knowledge Engineering). Dabei sollen insbesondere graphenbasierte Entwurfssprachen untersucht und deren verschiedene Wissensmodellierungsansätze genutzt werden, um eine statische und dynamische Wissensbasis zu beschreiben.

Insgesamt soll im weiteren Verlauf dieser Arbeit eine Vorgehensweise beschrieben werden,

[4]Benannt nach John Venn, der diese spezielle Form des Mengendiagramms in [Venn, 1880] vorgestellt hat.

die Aspekte aus allen vier Bereichen in der Automatisierung spezieller Konstruktionsvorgänge berücksichtigen können soll. Diese wird in Anlehnung an eine Bezeichnung aus der automatisierten Softwareentwicklung als *Round-Trip-Engineering* bezeichnet und trägt auch der Tatsache Rechnung, dass der Entwurfsprozess inhärent iterativer Natur ist, da Entscheidungsäste exploriert und dazu ausdetailliert werden müssen. Erkennt man dabei eine Sackgasse, wird am Ausgangspunkt eine andere Entscheidungsalternative gewählt. Darüber hinaus soll auch eine konkrete Umsetzung und Implementierung der erarbeiteten Konzepte angestrengt werden.

In [Kohli und Krishnamurti, 1989] klassifizieren die Autoren das „share-of-choices problem", das einen Teilbereich der hier angestrebten Vorgehensweise darstellt, als NP-schwer[5]. Infolgedessen müssen auch heuristische oder andere nicht analytische Vorgehensweisen in dieser Arbeit vorgesehen und integriert werden. Des Weiteren kann die angestrebte Variantenbildung, als kombinatorisches Problem verschiedener Produkteigenschaften aufgefasst werden, das über alle Grenzen hinaus wachsen und damit zu Speicher- und Laufzeitproblemen führen kann.

In [Durhuus und Eilers, 2014] wird beispielsweise am sehr einfach scheinenden Problem der möglichen räumlichen Kombination von sechs 2×4 LEGO®-Blöcken aufgezeigt, dass es hierfür bereits 915.103.765 verschiedene Möglichkeiten gibt [Durhuus und Eilers, 2014, S. 433]. Aus diesem Grund müssen im Rahmen dieser Arbeit ebenfalls Strategien zur Umgehung der Variantenexplosion gefunden werden. Die Zielsetzung der vorliegenden Arbeit soll an dieser Stelle auf fünf zentrale Forschungsfragen verdichtet werden, die in Tabelle 1.1 aufgelistet sind.

Tabelle 1.1: Forschungsfragen (FF), die in der vorliegenden Arbeit adressiert werden.

FF 1	Welche Modelle und Vorgehensweisen aus dem Softwareentwicklungsprozess lassen sich im Hinblick auf eine Automatisierung auf den ingenieurwissenschaftlichen Entwurf übertragen?
FF 2	Welche Prozesse der Produktentstehung eignen sich für eine Automatisierung?
FF 3	Wie kann der verschiedene Abstraktionsebenen überschreitende, domänenübergreifende, mehrfach iterative Prozess des Ingenieurentwurfs in einem automatisierten Framework abgebildet werden?
FF 4	Wie können Anforderungen an ein Produkt variabel definiert und in einem automatisierten Verarbeitungsprozess dynamisch zur Laufzeit interpretiert werden, sodass geänderte Anforderungen auf ein geändertes Produkt führen?
FF 5	Wie lassen sich Anforderungen, die sich auf die Form eines Produktes auswirken, im ingenieurwissenschaftlichen Entwurf allgemeingültig abbilden und einbringen?

Im weiteren Verlauf dieser Arbeit soll sich der Beantwortung der obigen Fragen angenommen werden. Hierfür wurde ein Forschungsansatz entwickelt und umgesetzt, der in der Folge auch auf die Struktur der Arbeit führt und im nächsten Abschnitt dieser Einführung vorgestellt wird.

[5] Ein Problem gilt als NP-schwer, wenn es nicht von einem deterministischen polynomiellen Algorithmus berechnet werden kann [Hromkovič, 2011, S. 232].

1.3 Forschungsansatz und Struktur der Arbeit

In der Literatur finden sich diverse generische Forschungsansätze, auf deren Basis sich wissenschaftliche Arbeiten systematisieren lassen (z. B. [Eckert et al., 2003], [Kothari, 2004] oder [Blessing und Chakrabarti, 2009]). Die Grundstruktur der vorliegenden Arbeit orientiert sich an der im Kontext des ingenieurwissenschaftlichen Produktentwurfs weitverbreiteten Design Research Methodology (DRM) nach BLESSING und CHAKRABARTI [Blessing und Chakrabarti, 2009]. Der allgemeine Ablauf der DRM ist in Abbildung 1.3 dargestellt.

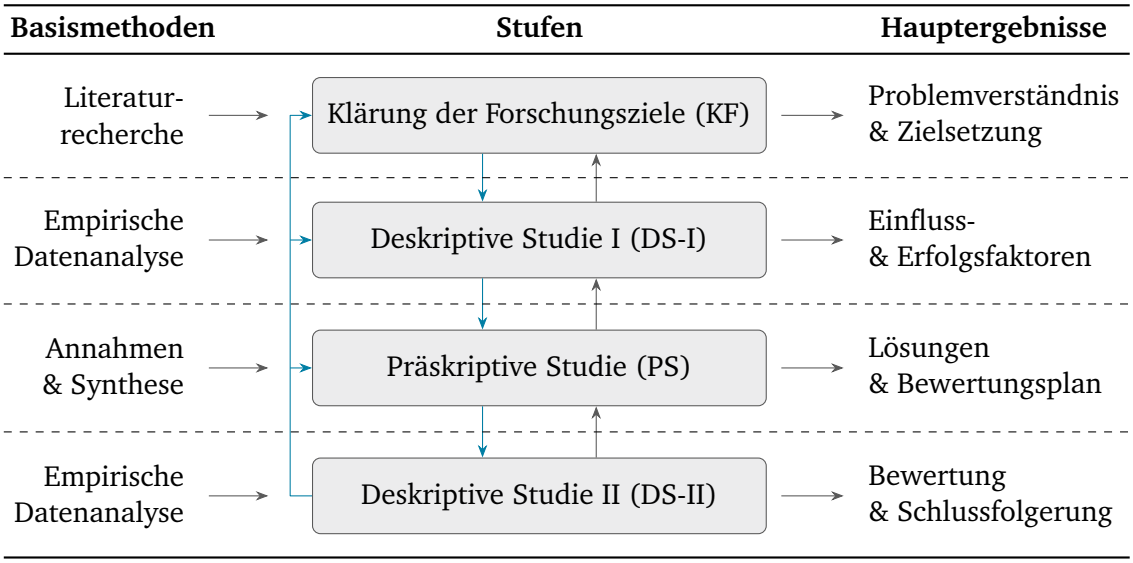


Abbildung 1.3: Vier Stufen der DRM [Blessing und Chakrabarti, 2009, S. 15]

Nach der DRM lässt sich ein Forschungszyklus in vier allgemeine Stufen (mittlere Spalte) unterteilen, die durch diverse Basismethoden (linke Spalte) unterstützt durchlaufen werden und dabei auf die Hauptergebnisse (rechte Spalte) führen sollen. In der DRM wird des Weiteren zwischen sieben unterschiedlichen Forschungsprojekttypen differenziert, die sich hauptsächlich in der Bearbeitungstiefe – initial, revisionsbasiert und umfassend – der einzelnen Stufen und der Anzahl an Iterationsschleifen unterscheiden[6]. [Blessing und Chakrabarti, 2009, S. 60–63]

In der vorliegenden Arbeit steht die Übertragung von Techniken aus der Softwareentwicklung auf Problemstellungen im Zusammenhang mit der Automatisierung des ingenieurwissenschaftlichen Entwurfs im Fokus der Forschungsaktivität. Hierfür sollen die bekannten graphenbasierten Entwurfssprachen über ihren bisherigen Anwendungsbereich hinaus theoretisch und praktisch erweitert werden, um als Grundlage für eine prototypische Implementierung zu dienen.

Ausgehend von einer Analyse des Zusammenhangs der Informatik und des Ingenieurwesens wird eine Unterstützungsmöglichkeit abgeleitet und initial evaluiert. Dieses Vorgehen zielt darauf ab, eine theoretische und praktische Unterstützung in der Automatisierung des Entwurfsprozesses zu erzielen, wobei das Verständnis über die wissenschaftliche Ausgangslage noch unvollständig ist. Ein solches hochinnovatives Vorhaben passt somit am besten zu einem

[6]Eine Ausnahme hiervon bilden die Forschungstypen eins und zwei, wobei Forschungstyp eins nur die ersten beiden und Forschungstyp zwei die ersten drei Stufen überstreicht [Blessing und Chakrabarti, 2009, S. 60]

Forschungsprojekt vom Typ fünf der DRM (vgl. [Blessing und Chakrabarti, 2009, S. 62]).

Nach dem Vorbild der DRM umfasst die vorliegende Arbeit die Herleitung des theoretischen Verständnisses und der Übertragung von Vorgehensweisen und Begriffen in Form einer deskriptiven Studie I. Darauf aufbauend wird mit der Beschreibung eines Entwurfsrahmenwerks eine detaillierte präskriptive Studie dargestellt. Wie bei jeder differenzierten Unterstützungsentwicklung folgt darauf eine initiale deskriptiven Studie II ([Blessing und Chakrabarti, 2009, S. 62]), hier in Form von Anwendungsbeispielen zur Validierung. Nachfolgend werden die in dieser Arbeit vorgenommenen Tätigkeiten und Ergebnisse in den einzelnen Stufen der DRM dargelegt.

Klärung der Forschungsziele (KF) Die Klärung der Forschungsziele erfolgt durch eine Literaturrecherche im Bereich der modernen, automatisierten Softwareentwicklung (Abschnitt 2.1) und der Theorien, Modelle und Vorgehensweisen im ingenieurwissenschaftlichen Produktentwurf (Abschnitt 2.2). Vor diesem Hintergrund werden die Forschungsfragen abgeleitet und Kriterien erarbeitet, die zur Bewertung des zu entwickelnden Entwurfsrahmenwerks herangezogen werden sollen. Die Kriterien werden ebenfalls für die deskriptive Studie I herangezogen.

Deskriptive Studie I (DS-I) In der Phase der deskriptiven Studie I wird auf Basis der vorgestellten Grundlagen eine Übersicht über den Stand des Wissens und der Technik in der Literatur im Umfeld der Forschungsfragen und der erarbeiteten Kriterien aus der KF gegeben (Abschnitt 2.3). Einerseits sollen auf diese Weise Defizite in der Modellanschauung, Vorgehensweise oder den Algorithmen analysiert werden, andererseits können bereits vorhandene positiv evaluierte Elemente im eigenen Forschungsansatz reflektiert und integriert werden. Die typischerweise an dieser Stelle angestrengte Untersuchung der Vorgehensweise menschlicher Konstrukteure wird nicht ausgeführt, da das Ziel eine zumindest auf einem speziellen Anwendungsbereich gültige, vollständige Automatisierung darstellt.

Präskriptive Studie (PS) In der Phase der präskriptiven Studie werden die in der KF und DS-I gesammelten Erkenntnisse genutzt, um eine Übertragung von Vorgehensweisen aus der Softwareentwicklung auf das Ingenieurwesen an den Stellen vorzunehmen, an denen es vielversprechend erscheint. Darauf aufbauend wird eine neuartige Konstruktionsphilosophie motiviert (Abschnitt 3.1), die schließlich in ein theoretisches Entwurfsrahmenwerk überführt wird, das die philosophischen Überlegungen zu realisieren in der Lage ist (Abschnitt 3.2). Schließlich wird eine prototypische Implementierung des Entwurfsrahmenwerks unter Verwendung der sogenannten graphenbasierter Entwurfssprachen vorgenommen (Kapitel 4).

Deskriptive Studie II (DS-II) In der Phase der deskriptiven Studie II erfolgt die Evaluierung und Validierung des in der PS entwickelten Entwurfsrahmenwerks anhand zahlreicher Beispiele, die jeweils einen Aspekt der Philosophiebetrachtungen in den Vordergrund stellen. Die prinzipiellen Vor- und Nachteile der neuartigen Sichtweise auf den Konstruktionsprozess wird anschließend diskutiert und vor dem Hintergrund der DS-I bewertet (Kapitel 5). Die Fundiertheit der aus dieser Arbeit resultierenden Erkenntnisse wird plausibilisiert, während

vor dem Hintergrund der Neuartigkeit der Theorie und der lediglich prototypisch erfolgten Implementierung die Auslotung der tatsächlichen Prozessverbesserungspotenziale in der Praxis im Ausblick der weiteren Forschungsaktivität anheimgestellt wird (Kapitel 6).

Struktur der Arbeit Aus dem vorangegangen vorgestellten Forschungsansatz ergibt sich eine natürliche Struktur für die vorliegenden Arbeit. Diese ist in Abbildung 1.4 dargestellt.

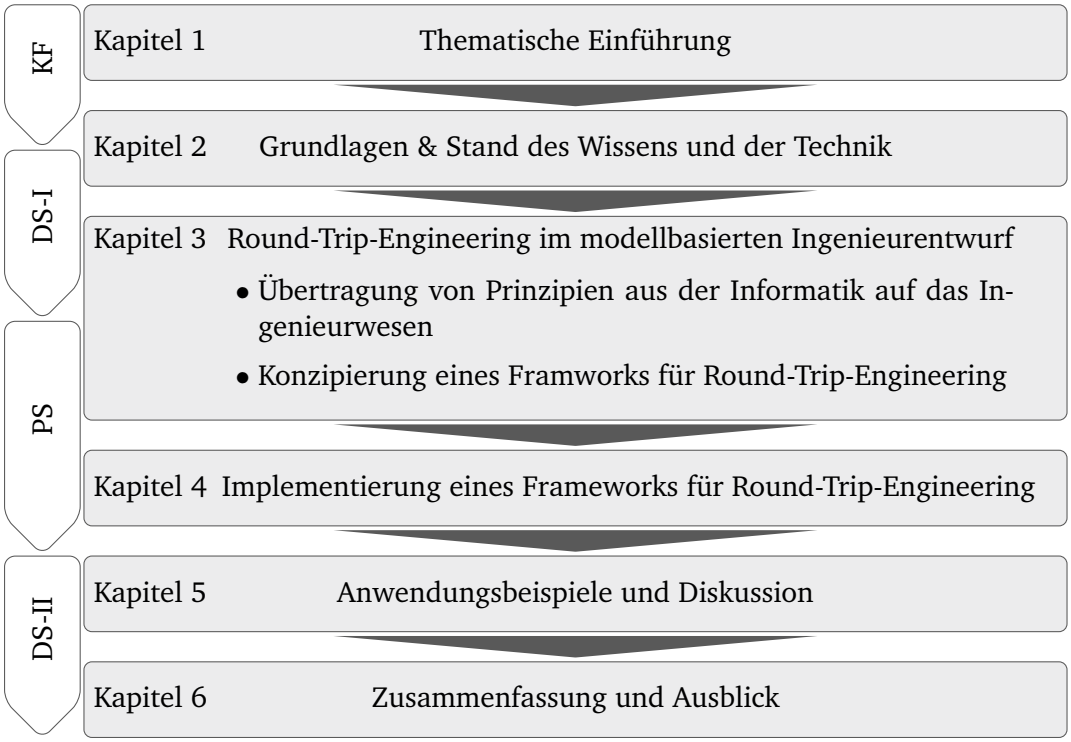


Abbildung 1.4: Struktur der vorliegenden Arbeit

Zunächst werden in Kapitel 2 die Grundlagen dargestellt, die den Rahmen der vorliegenden Arbeit bilden. Hierbei liegt der Fokus auf den Domänen der modernen Softwareentwicklung (siehe Abschnitt 2.1) und des ingenieurwissenschaftlichen Entwurfs mit seinen Begriffen, Theorien und Modellen (siehe Abschnitt 2.2). Es schließt eine Literaturübersicht des aktuellen Stands des Wissens und der Technik an (siehe Abschnitt 2.3), die eine Tour d'Horizon geben soll, auf welcher Wissensbasis aufbauend die vorgestellten Methoden und Werkzeuge hervorgegangen sind. Zur besseren Einordnung werden die Beiträge der genannten Quellen in Übersichtstabellen den fünf abgeleiteten Forschungsfragen aus Tabelle 1.1 gegenübergestellt.

Im darauffolgenden Kapitel 3 wird die Übertragung von Prinzipien und Vorgehensweisen aus der Informatik auf das Ingenieurwesen in Form des Round-Trip-Engineerings motiviert und dargelegt, welche prinzipiellen Vorteile, neuen Einsichten in den Entwurfsprozess und Möglichkeiten für ein zukünftiges Engineering damit einhergehen (siehe Abschnitt 3.1).

Es folgt die Vorstellung eines theoretischen Frameworks, das für einen eingegrenzten Anwendungsbereich die Anforderungen des philosophischen Konzepts erfüllt (siehe Abschnitt 3.2).

In Kapitel 4 wird eine prototypische Implementierung des zuvor rein theoretisch beschriebenen Entwurfsrahmenwerks demonstriert, die auf den bereits aus vielen Vorarbeiten bekannten

graphenbasierten Entwurfssprachen fußt. Einige Konzepte der Entwurfssprachen werden dazu erweitert (siehe Abschnitt 4.3 und Abschnitt 4.4).

Zur Prinzipdemonstration und Validierung des Entwurfsrahmenwerks werden in Kapitel 5 einige Anwendungsbeispiele präsentiert, die den Fokus jeweils auf bestimmte Aspekte der Entwurfsphilosophie legen. Am Ende dieses Kapitels folgt eine Reflexion und Diskussion der erarbeiteten Vorgehensweise. Die Arbeit schließt in Kapitel 6 mit einer Zusammenfassung der wichtigsten Ergebnisse und einem Ausblick für Ansatzpunkte zukünftiger Forschungsaktivitäten.

Stand des Wissens und der Technik

2

> *Die meisten Probleme entstehen bei ihrer Lösung.“*
>
> **— Leonardo da Vinci**
> (italienischer Maler, Bildhauer, Architekt, Anatom, Mechaniker, Ingenieur und Naturphilosoph)

In diesem Kapitel werden die wichtigsten Grundlagen und Zusammenhänge, die für ein besseres Verständnis der vorliegenden Arbeit förderlich sind, vorgestellt. Durch den domänenübergreifenden Charakter dieser Arbeit wird eine Vielzahl unterschiedlicher Wissensgebiete überstrichen. Die Themen der Software- und Produktentwicklung sowie deren Entstehungsprozesse und Modellvorstellungen stellen dabei zentrale Grundbausteine dar, die in den ersten beiden Sektionen näher beleuchtet werden. Das Kapitel schließt mit einem Überblick der wichtigsten Beiträge aus den verschiedenen Wissensgebieten, die als Grundlage oder Inspiration dieser Arbeit dienen. Aufgrund der Vielzahl an Themen und der schier unendlichen Menge an zugehöriger Literatur hat dieser Grundlagenteil nicht den Anspruch eines allumfassenden Überblicks und konzentriert sich auf die aus Sicht des Verfassers wesentlichen Aspekte im Stand des Wissens und der Technik. Nicht alle ausgewerteten Quellen haben Eingang in dieses Kapitel gefunden.

Kapitelübersicht

2.1 Aspekte moderner Softwareentwicklung

Die Entwurfsprinzipien und Modellierungstechniken der modernen Softwareentwicklung haben das in dieser Arbeit entwickelte Round-Trip-Engineering im modellbasierten Ingenieurentwurf stark geprägt. Wie im weiteren Verlauf der Arbeit noch genauer beschrieben wird, sind einige Prinzipien in die dahinter stehende Philosophie eingeflossen (vgl. Kapitel 3). Aus diesem Grund sollen im Folgenden die wichtigsten Einflussfaktoren aus der Informatik vorgestellt werden.

2.1.1 Objektorientierte Softwareentwicklung

Die objektorientierte Programmierung (OOP) stellt sowohl eine explizite als auch implizite Grundlage für die vorliegende Arbeit dar. Explizit, da in die als Entwicklungsumgebung dieser Arbeit zugrunde liegenden graphenbasierten Entwurfssprachen die meisten Techniken und Konzepte der OOP übernommen wurden (siehe hierzu [Vogel und Arnold, 2020]). Implizit, da sich die objektorientierte Denkweise und einige ihrer Modellierungstechniken in Teilen auf das Ingenieurwesen übertragen lassen. Nachfolgend sollen die wichtigsten Eigenschaften und Prinzipien der OOP dargestellt werden.

Die vier Pfeiler der Objektorientierung

In der OOP werden die vier Hauptprinzipien *Abstraktion*, *Datenkapselung*, *Vererbung* und *Polymorphismus* postuliert und in der Syntax und Grammatik der korrespondierenden Programmiersprachen konzeptuell berücksichtigt [Sharan und Davis, 2022, S. 13].

Abstraktion Allgemein bezeichnet Abstraktion das bewusste Präterieren aus dem aktuellen Blickpunkt heraus unbedeutender Details und die Fokussierung auf konstituierende Merkmale zur Komplexitätsreduktion. Ein Mittel der Abstraktion stellt die Modellbildung dar. [Broy, 2019, S. 10; Goll, 2014, S. IX; Sharan und Davis, 2022, S. 13]. Abstraktion im Rahmen der OOP wird unter anderem durch Vererbung und das Zusammenfassen von Methoden in Schnittstellen erreicht. „*Ein Objekt implementiert sein Verhalten in Schnittstellenmethoden, die außerhalb des Objekts nur als Abstraktion in Form der Methodenköpfe sichtbar sind*“ [Goll, 2014, S. 7].

Datenkapselung Unter dem Begriff Datenkapselung versteht man die Gruppierung von Daten und Operationen in einer gemeinsamen Kapsel. [Goll, 2014, S. XIV; Sharan und Davis, 2022, S. 13, 23] Die Kapsel selbst ist eine spezielle Entität der jeweiligen objektorientierten Programmiersprache. In Java können beispielsweise Klassen, Pakete oder Module zur Datenkapselung verwendet werden. [Sharan und Davis, 2022, S. 23] Eng verknüpft mit der Datenkapselung ist das Verbergen von Information (*Information Hiding*). Durch spezielle Mechanismen wie – z. B. Zugriffsmodifikatoren oder Schnittstellen – kann innerhalb der Kapsel die Sichtbarkeit der Inhalte oder Methoden nach außen individuell festgelegt werden. (vgl. [Goll, 2014, S. 6])

Vererbung Die Vererbung wird in der OOP zur Ableitung einer neuen Klasse aus einer bestehenden verwendet und ermöglicht auf diese Weise die Herstellung einer hierarchischen Beziehung zwischen der Basisklasse (Superklasse) und der erbenden Klasse (Subklasse) [Sharan und Davis, 2022, S. 13]. Durch die Vererbung erhält die Subklasse automatisch alle Attribute und Methoden der Superklasse, d. h. sie erbt Struktur und Verhalten ihrer Basisklasse. [Goll, 2014, S. XX] In der Vererbungshierarchie stellen ein Supertyp und sein Subtyp eine „ist-ein“-Beziehung dar [Sharan und Davis, 2022, S. 25]. Die Vererbung ist durch die mit ihr einhergehende Reduktion von Duplikation und Vermeidung von Inkonsistenzen einer der wichtigsten Gründe, warum objektorientierter Code leicht erweiterbar ist [Kak, 2003, S. 31].

Polymorphismus Das Wort Polymorphismus stammt ursprünglich aus dem Griechischen, wobei „poly“ für viel und „morphos“ für Gestalt steht [Sharan und Davis, 2022, S. 26]. Im Deutschen wird häufig auch der Begriff der Vielgestaltigkeit als Synonym verwendet. In der OOP ermöglicht Polymorphismus es, dass eine Entität in einer Vererbungshierarchie von verschiedenem Typ sein und dadurch je nach Kontext unterschiedliche Bedeutungen annehmen kann [Sharan und Davis, 2022, S. 13; Goll, 2014, S. 12]. Polymorphismus ermöglicht unterschiedliche Definitionen für gleichnamige Funktionen in verschiedenen Klassen und den automatischen Aufruf von objektspezifischen Definitionen solcher Funktionen zur Laufzeit [Kak, 2003, S. 32].

Entwurfsmuster

Entwurfsmuster sind bewährte, generische Lösungen für wiederkehrende Entwurfsprobleme, indem Klassen oder Objekte in definierten Rollen zusammenarbeiten. Sie fassen auf diese Weise Design- und Architekturwissen zusammen und sollen Entwickler unterstützen, ihre Software flexibler, verständlicher, performanter und leichter wiederverwendbar zu implementieren (vgl. [Goll, 2014, S. XII, 64; Eilebrecht und Starke, 2019, S. V])

Entwurfsmuster wurden ursprünglich von ALEXANDER [Alexander et al., 1977] aus der Architektur heraus entwickelt und dann im Rahmen der Entwicklung der Programmiersprache Smalltalk von BECK und CUNNINGHAM und später vollumfänglich von GAMMA in seiner PhD-Thesis [Gamma, 1992] auf die Softwaretechnik übertragen. Im Jahr 1995 veröffentlichte GAMMA schließlich zusammen mit HELM, JOHNS und VLISSIDES („Gang of Four“) das Standardwerk „Design patterns“ [Gamma et al., 1995] zum Thema. (vgl. [Goll, 2014, S. 64, 69])

In [Gamma et al., 1995] werden insgesamt 23 objektorientierte Entwurfsmuster aus den drei verschiedenen Kategorien Erzeugungs-, Verhaltens- und Strukturmuster vorgestellt. Strukturmuster, wie beispielsweise das Adapter-, Dekorierer- oder Fassaden-Muster, befassen sich mit der Komposition und der Granularität von Klassen und Objekten [Goll, 2014, S. 75, 77]. Verhaltensmuster, wie z. B. Beobachter-, Strategie- oder Besucher-Muster, behandeln Zuständigkeiten und Zusammenarbeit zwischen Klassen bzw. Objekten und beschreiben deren Interaktionen [Goll, 2014, S. 75, 76, 77]. Erzeugungsmuster thematisieren die Unabhängigkeit eines Systems von der Art der Objekterzeugung. Bekannte Vertreter der Erzeugungsmuster sind das Fabrikmethode-, das Abstrakte Fabrik- oder das Singleton-Muster (vgl. [Goll, 2014, S. 76, 78]).

Im Rahmen der vorliegenden Arbeit sollen einige Prinzipien der Entwurfsmuster auf das Engineering übertragen werden (siehe Kapitel 3). Für eine tiefergehende Darstellung der Entwurfsmuster in der Informatik seien [Gamma et al., 1995] und [Goll, 2014] angeführt.

2.1.2 Modellgetriebene Softwareentwicklung

„*Modellgetriebene Softwareentwicklung ist ein Oberbegriff für Techniken, die aus formalen Modellen automatisiert lauffähige Software erzeugen*“ [T. Stahl und Bettin, 2007, S. 11]. An dieser Stelle soll, stellvertretend für die zahlreichen in der Literatur beschriebenen Ansätze, die Model-driven Architecture herausgegriffen werden, da sie die Hauptinspirationsquelle aus der Informatik für die Übertragung auf das Ingenieurwesen im Rahmen der vorliegenden Arbeit darstellt.

Model-driven Architecture

Die Model-driven Architecture (MDA) ist ein Framework für die modellgetriebene Entwicklung komplexer Softwaresysteme, der, wie auch die Unified Modeling Language (UML), von der Object Management Group (OMG) spezifiziert wird [MDA, 2014]. Das Hauptziel der MDA besteht darin „*Struktur, Semantik und Notationen von Modellen*“ zu definieren und dadurch die Komplexität zu verringern und „*Wert aus den Modellen schöpfen zu können*“ [MDA, 2014, S. 1].

MDA-basierte Anwendungen können grundsätzlich in drei elementare Modellschichten unterteilt werden [MDA, 2014, S. 8]. Auf der höchsten Abstraktionsebene sind die Geschäfts- oder Domänenmodelle, die historisch auch als Computation Independent Model (CIM) bezeichnet wurden [MDA, 2003, S. 19], angesiedelt [MDA, 2014, S. 8]. Diese Modelle konzentrieren sich auf geschäftliche Konzepte und enthalten die Anforderungen (Requirements) an das zu entwickelnde System, wobei alle Abhängigkeiten zu möglichen Realisierungen zunächst ignoriert werden [Cephas, 2006, S. 5]. Die Instanzen eines CIM sind reale Objekte [MDA, 2014, S. 8] und korrespondieren mit dem Vokabular des Domänenexperten ohne Programmierbezug [Cephas, 2006, S. 9]. Eine Ebene darunter befinden sich die logischen Systemmodelle [MDA, 2014, S. 8]. Diese werden oft auch als Platform Independent Models (PIM) bezeichnet [MDA, 2003, S. 19; Cephas, 2006, S. 9] und bilden die Interaktion von Systemkomponenten untereinander, sowie strukturelle und operative Aspekte eines Systems ab [MDA, 2014]. Ein PIM ist so konzipiert, dass es plattformunabhängig ist, und basiert im Allgemeinen auf der Annahme einer technologieneutralen virtuellen Maschine mit abstrakten Systemservices [MDA, 2003, S. 16]. Auf der untersten Abstraktionsebene sind schließlich die Implementierungs- oder Platform Specific Models (PSM) definiert. Ein PSM ist bereits in einer speziellen Programmiersprache implementiert und somit relativ hardwarenah an eine Plattform gebunden[1]. Das PSM kombiniert somit die Spezifikationen aus dem PIM mit dem Umstand, wie das betrachtete System technisch auf einer bestimmten Art von Plattform realisiert werden kann [MDA, 2003, S. 16].

In der Philosophie der MDA ist der Prozess von einem CIM über ein PIM hin zu einem PSM vollständig oder mindestens partiell automatisiert [MDA, 2014, S. 2]. Die ausführbaren Dateien (Executables) können ebenfalls automatisch aus dem zugehörigen PSM abgeleitet werden. Der allgemeine Arbeitsablauf ist in Abbildung 2.1 noch einmal zusammengefasst.

Abbildung 2.1: Allgemeines Transformationsschema der MDA [Cephas, 2006, S. 7]

Die Transformation eines Quellmodells (z. B. ein PIM) in ein Zielmodell (z. B. ein PSM) wird als Modell-zu-Modell (M2M) Transformation bezeichnet [Brambilla et al., 2017, S. 123]. Die Transformation eines Quellmodells (z. B. PSM) in eine konkrete Implementierung wird als

[1] Diese Aussage ist nicht auf die Vorstellung einer klassisch kompilierten Programmiersprache wie z. B. C++ beschränkt. Bei Mischformen aus kompilierten und laufzeitinterpretierten Sprachen wie z. B. Java ist die Plattformabhängigkeit durch die Bindung des kompilierten Bytecodes an die virtuelle Maschine gegeben, auch wenn diese zunächst als unabhängig von der tatsächlichen Hardware betrachtet wird.

Modell-zu-Text (M2T) Transformation bezeichnet [Brambilla et al., 2017, S. 141]. Diese beiden Transformationsformen sind in Abbildung 2.2 a) und b) grafisch aufbereitet.

(a) Modell-zu-Modell-Transformation (M2M) (b) Modell-zu-Text-Transformation (M2T)

Abbildung 2.2: Transformationsformen in der modellgetriebenen Softwareentwicklung

Für tiefergehende Informationen zu von der OMG spezifizierten M2M-Transformationen sei auf [QVT, 2016] verwiesen, für M2T-Transformationen auf [MOFM2T, 2008]. In [Schopper und Rudolph, 2018] wird eine zusätzliche Text-zu-Modell (T2M)-Transformation im Kontext des Ingenieurentwurfs eingeführt. Mehr zu diesem Thema ist in Kapitel Abschnitt 3.1.2 zu lesen.

2.1.3 Unified Modeling Language

Die UML ist ein von der OMG spezifizierter grafisch notierter Standard „*mit dessen Hilfe sowohl die statischen als auch die dynamischen Aspekte beliebiger Anwendungsgebiete modelliert werden können*“ [Kecher et al., 2021, S. 23]. „*Die UML zielt darauf ab, Systemarchitekten, Softwareingenieuren und Softwareentwicklern ein Werkzeug zur Analyse, zum Entwurf und zur Implementierung von softwarebasierten Systemen zur Verfügung zu stellen. Sie entstand als Zusammenführung dreier führender objektorientierter Methoden (Booch, OMT und OOSE) und vereint eine Vielzahl bewährter Methoden aus Modellierungssprachen, objektorientierter Programmierung und Architekturbeschreibungssprachen*“ [UML, 2017, S. 1]. In [Reichwein, 2011] wurde gezeigt, dass die UML geeignet ist, als Modellierungsstandard für Entwurfssprachen angewendet zu werden. Die hierfür verwendeten Modellierungselemente werden im Folgenden kurz vorgestellt. Für eine ausführliche Beschreibung aller Elemente des Standards sei auf [UML, 2017] verwiesen.

Klasse und Klassendiagramm

Klassen dienen in der objektorientierten Modellierung dazu, Attribute und Methoden zu gruppieren und zu kapseln [Rumpe, 2011, S. 16–17]. Ein Klassendiagramm stellt eine Sammlung zusammenhängender Klassen für eine bestimmte Modellierungsaufgabe dar. Die Zusammenhänge können impliziter Natur sein oder explizit über Klassenverbindungselemente modelliert sein. Im Kontext von Entwurfssprachen handelt es sich bei den Verbindungselementen der Klassen um *Vererbungen* („ist-ein“-Beziehungen) [Rumpe, 2011, S. 23] und *Assoziationen* („hat-ein“ oder „verwendet“ -Beziehungen) [Rumpe, 2011, S. 25]. Außerdem können über sie Eigenschaften bzw. Datenfelder unterschiedlicher Klassen in *Constraints* (Algebraische Gleichungen) verknüpft werden. Die modellierten Klassen bilden die Schablonen für die Objekte, die beim Verarbeiten der Regeln durch den Entwurfscompiler erzeugt werden und in ihrer Gesamtheit den Entwurfsgraphen bilden. Das Instanziieren geschieht in den Objektdiagrammen, die in den Regeln eingebettet sind und im folgenden Abschnitt erläutert werden.

Objektdiagramm

In der UML dient ein Objektdiagramm zur Definition einer Modellinstanz einer in einem Klassendiagramm allgemein definierten Klasse, sowie des Setzens von Attributen dieser Objekte [Rumpe, 2011, S. 126–127]. Die im Klassendiagramm modellierten Assoziationen können in einem Objektdiagramm ebenfalls in Form sogenannter *Links* instanziiert werden [Rumpe, 2011, S. 118–119]. In Entwurfssprachen wird dadurch die konkrete Beziehung d er Objekte untereinander zur Laufzeit des Compilers modelliert. Das Objektdiagramm ist eingebettet in den Regeln der Entwurfssprache. Die Reihenfolge der Regeln wird durch ein sogenanntes *Aktivitätsdiagramm* definiert, das im nachfolgenden Abschnitt beschrieben wird.

Aktivitätsdiagramm

In der UML stellt ein Aktivitätsdiagramm eine spezielle Ausprägung eines Verhaltensdiagramm dar [UML, 2017, S. 685] und dient der Beschreibung und grafischen Visualisierung des Kontroll- und Datenflusses des modellierten Systems [UML, 2017, S. 373]. Die Beschreibung kann eine Unteraktivität des Systems oder die ganze Systemdynamik einschließen. Im Design Cockpit 43® (DC43) wird von den im UML-Standard definierten Aktivitäten nur ein sehr kleiner Teil zur Definition der Anwendung und der Abarbeitungsreihenfolge der Regeln für den Entwurfscompiler spezifiziert. Die verwendeten grafischen Elemente sind der Start- und Endknoten [UML, 2017, S. 391], sowie Kontrollflüsse [UML, 2017, S. 381].

2.1.4 Round-Trip-Engineering in der Softwareentwicklung

Der Begriff des Round-Trip-Engineering (RTE_S) wird in der Softwareentwicklung häufig verwendet, allerdings existiert bisher noch keine eineindeutige, allgemein anerkannte Definition des Begriffs. Da der mit RTE_S verbundene Vorgang in der vorliegenden Arbeit eine zentrale Rolle einnimmt, wird zunächst eine Begriffsfindung aus dem Bereich der Softwaretechnik angestrengt, bevor das RTE_S dann später auf das Ingenieurwesen übertragen werden kann.

In [Seifert, 2011] wird RTE_S als „*Zweig der Softwareentwicklung, der die Entwicklung und Anwendung von Techniken und Werkzeugen zur Automatisierung des Prozesses der globalen Konsistenz über zusammenhängende Software-Artefakte hinweg umfasst*“ beschrieben. In [Eden et al., 2018] wird RTE_S definiert als „*die Aufgabe, die Konsistenz zwischen Entwurf und Implementierung während des gesamten Software-Lebenszyklus zu wahren*“. Hierbei wird RTE_S als ein Framework zum Abgleich zwischen Softwarespezifikation und Softwareimplementierung gesehen. Die Autoren argumentieren, dass diese vor allem in modernen Softwareentwicklungsprozessen, wie beispielsweise der agilen Softwareentwicklung, immer weiter auseinanderdriften und in der Folge schwerer lesbare und qualitativ schlechtere Software entstehe [Eden et al., 2018, S. 19].

In [Sendall und Küster, 2004] wird RTE_S beschrieben als „*Fähigkeit in Software-Entwicklungsumgebungen bzw. -Werkzeugen, die Konsistenz mehrerer Software-Artefakte automatisch aufrechtzuerhalten*“ [Sendall und Küster, 2004, S. 1]. Hierbei sei besonders hervorgehoben, dass die Autoren die Automatisierung als essenziellen Bestandteil der Definition von RTE_S ansehen.

Allen drei bisher angeführten Definitionen ist gemein, dass der Konsistenthaltung verschiede-

ner Modelle oder sonstiger Artefakte beim RTE_S eine wichtige Rolle zukommt. In [Chikofsky und Cross, 1990] wird RTE_S im Grundsatz beschrieben, ohne allerdings dafür explizit das Wort RTE_S zu verwenden. Die Autoren beschreiben anstatt dessen drei sich ergänzende Vorgänge, die in einem Softwareentwicklungsprozess durchlaufen werden: das Forward-Engineering (FE), das Reverse-Engineering (RE) und das Re-Engineering (REE).

Unter dem FE verstehen die Autoren den traditionellen, vorwärts gerichteten Softwareentwicklungsprozess von der abstrakten Beschreibung bis hin zur Implementierung [Chikofsky und Cross, 1990, S. 14–15]. In diesem Zusammenhang kann die MDA (vgl. Abschnitt 2.1.2) insgesamt auch als eine systematische Methode für das FE gedeutet werden.

Das RE umschreiben die Autoren als einen Analyseprozess, bei dem die Bestandteile und deren Zusammenhänge eines Systems identifiziert und eine neue Repräsentation des Systems auf einem höheren Abstraktionsniveau angestrebt wird [Chikofsky und Cross, 1990, S. 15].

Den Vorgang des REE fassen die Autoren zusammen, als Untersuchung und Veränderung eines Systems, in der Absicht, es in neuer Form wiederherzustellen und in dieser zu implementieren [Chikofsky und Cross, 1990, S. 15]. In Abbildung 2.3 ist der Zusammenhang wischen FE, RE und REE grafisch dargestellt. Vor dem Hintergrund der bereits vorgestellten RTE_S-Definitionen kann das Zusammenspiel als Beschreibung eines RTE_S-Prozesses verstanden werden.

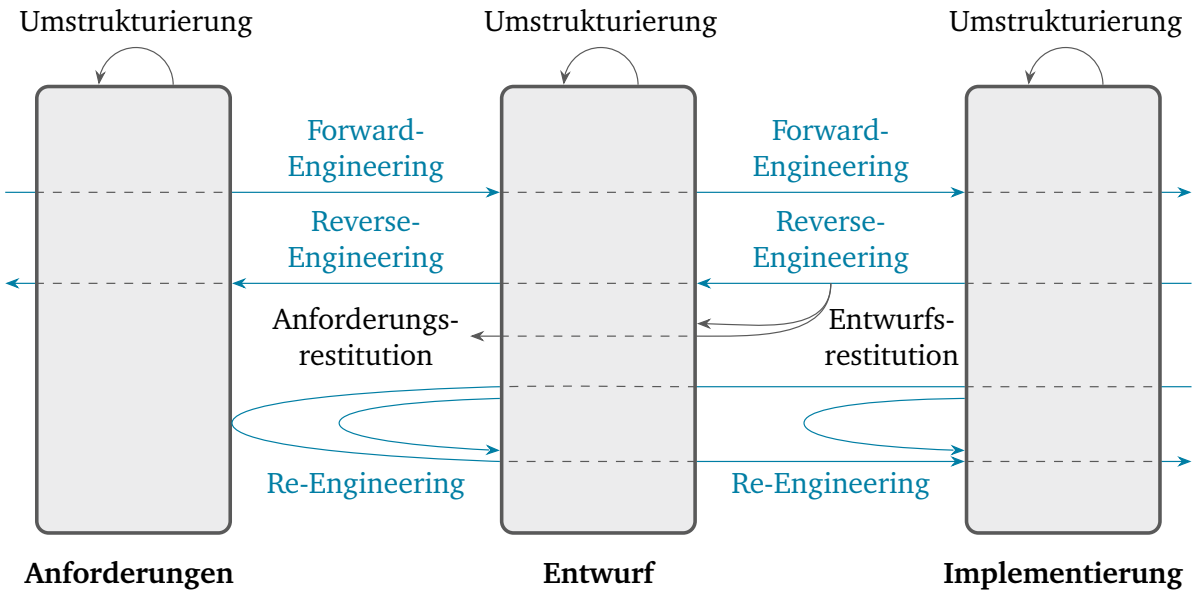


Abbildung 2.3: Forward, Reverse- and Re-Engineering [Chikofsky und Cross, 1990, S. 14]

Der in der obigen Abbildung dargestellte Prozess beginnt mit der Spezifikation von Anforderungen in einem Modell. Den ersten FE-Prozess stellt die Ableitung des Entwurfs aus den Anforderungen dar. In einem zweiten FE-Prozess kann dann aus dem Entwurf die Implementierung deriviert werden. Das RE- wird umgekehrt zum FE von der fertigen Implementierung her definiert. Von dort aus wird mit einem als Entwurfsrestitution genannten Verfahren das Entwurfsmodell abgeleitet und anschließend in einem zweiten RE-Prozess weiter zurück auf die Ursprungsanforderungen abstrahiert. Das REE tritt als Kombination eines initialen RE und

einem nachfolgenden FE-Verfahren zwischen verschiedenen Abstraktionsniveaus des Systems auf. Zusammenfassend lässt folgende Beschreibung der drei Vorgänge festhalten:

- Forward-Engineering (FE): Erstellen von Software aus Spezifikationen
- Reverse-Engineering (RE): Erstellen von Spezifikationen aus vorhandener Software
- Re-Engineering (REE): Verstehen vorhandener Software und Modifizieren derselben

Die Autoren in [Henriksson und Larsson, 2003] weisen ebenfalls auf den Zusammenhang zwischen RTE_S und RE hin und geben eine abstrakte Definition von RTE_S, die auf mathematisch formulierten Transformationssequenzen beruht. „*Sei D der Entwurfsvorgang für das Produkt P, sei r ein Reverse-Engineering-Verfahren, so dass $r(P) = D$ und sei g ein Produktgenerierungsverfahren, so dass $g(D) = P$. Wenn $g(r(P)) \equiv P$ (und somit $r(g(D)) \equiv D$), dann ist $\langle g, r \rangle$ ein Round-Trip-Engineering-System.*“ [Henriksson und Larsson, 2003, S. 2]. Die Deutung des RTE_S als Abfolge von Transformationen im mathematischen Sinne ist für das spätere Verständnis des auf das Ingenieurwesen übertragenen Begriffs ebenfalls von großer Bedeutung.

Auf Grundlage der vorgestellten Literaturanalyse zum Thema soll nun eine eigene Definition für RTE_S abgeleitet werden, die die für die vorliegende Arbeit wichtigsten Eigenschaften vereint.

Definition 2.1.1: **Round-Trip-Engineering in der Softwaretechnik**

Die durch Modelltransformationen automatische Aufrechterhaltung der globalen Konsistenz aller Modelle und Artefakte über alle Hierarchiestufen und den gesamten Produktlebenszyklus eines Systems hinweg wird als Round-Trip-Engineering in der Softwaretechnik bezeichnet. Die wichtigsten Modelltransformationen dabei werden als Forward-Engineering (FE)-, Reverse-Engineering (RE) und Re-Engineering (REE) bezeichnet.

2.2 Begriffe, Theorien und Modelle im ingenieurwissenschaftlichen Entwurf

In der vorliegenden Arbeit wird ein fortschrittliches Verständnis des ingenieurwissenschaftlichen Produktentwurfs auf Basis moderner Techniken aus der Informatik abgeleitet, das in ein zunächst rein theoretisch motiviertes und später prototypisch implementiertes Entwurfsrahmenwerk überführt wird. Um die Gemeinsamkeiten, vor allem aber auch die Unterschiede zu den gängigen Theorien und Modellen der Ingenieurwissenschaften herausarbeiten zu können, wird im Folgenden ein Überblick über den aktuellen Stand des Wissens und der Technik auf diesem Gebiet gegeben. Es sei darauf hingewiesen, dass die Selektion der Beiträge und gewählten Themenbereiche aus Sicht des Autors der vorliegenden Arbeit repräsentativ ist, aber aufgrund der Diversifikation des Wissensgebietes und der schier unendlichen Menge an Literatur zu diesem Thema kein Anspruch auf eine allumfassende Darstellung erhoben werden kann.

In [Hubka, 1984, S. 37–52] ist die geschichtliche Entwicklung der Konstruktionswissenschaften in verschiedenen Ländern zwischen 1940 und 1980 dargelegt. Diese Zeitspanne ist auch heute noch relevant, da seinerzeit die theoretischen Grundlagen der modernen Ingenieurwissenschaften gelegt wurden. Darüber hinaus bietet [Chakrabarti und Blessing, 2015] einen guten Überblick über die aus den Ingenieurwissenschaften und anderen Bereichen hervorgegangenen Theorien und Modelle im Produktentwurf ab Mitte des 20. Jahrhunderts.

In [Eigner et al., 2014, S. 15–48] sind einige Entwurfsmethodiken nach der Domänenherkunft – Mechanik, Elektrik, Software oder Mechatronik – unterteilt.

RUDOLPH fasst die Situation der Ingenieurwissenschaft folgendermaßen zusammen: „*Leider ist es bisher [. . .] nicht gelungen, in der Erstellung einer allgemeinen Entwurfstheorie eine umfassende, beweisbare und allgemein anerkannte konzeptionelle Modellvorstellung samt Weltmodell des Entwurfsprozesses im Ingenieurwesen zu entwickeln [. . .]*“ [Rudolph, 2002, S. 101]. Obwohl diese Aussage mehr als zwei Dekaden in die Vergangenheit zurückreicht, hat sich an dieser Tatsache bis heute nichts geändert. Er führt dazu weiter aus, dass die Ursache in der „*teilweisen Unverträglichkeit und Einteilung der im Entwurfsprozeß gleichzeitig anzutreffenden Modellbeschreibungen in topologisch und parametrisch getrennte Lösungsklassen und deren unterschiedlichen verbalen, symbolischen, analytischen und numerischen Teildarstellungen*“ [Rudolph, 2002, S. 130] liege. Er misst einem sprachbasierten Ansatz, wie er auch in den graphenbasierten Entwurfssprachen (siehe Abschnitt 2.2.4) verfolgt wird, die größte Aussicht auf Erfolg zu.

Der in dieser Arbeit beschriebene Mechanismus des Round-Trip-Engineering setzt auf diesem Verständnis auf und versucht einen Weg aufzuzeigen, wie eine vollständige Automatisierung für einen klar begrenzten Anwendungs- bzw. Aufgabenbereich erreicht werden kann. Im Folgenden sollen zunächst einige grundlegende Begriffe bestimmt und anschließend ausgehend von einem allgemeinen Problemlösungsprozess und der Modellbildung die wichtigsten Vorstellungen und Techniken im Status Quo des modellbasierten Ingenieurentwurfs zusammengefasst werden.

2.2.1 Von den Anforderungen zum fertigen Produkt

In den vom Verein Deutscher Ingenieuere (VDI) herausgegeben Richtlinien 2221 - 2223 [VDI 2221; VDI 2222; VDI 2223] werden einige grundlegende begriffliche Definitionen im Kontext des ingenieurwissenschaftlichen Produktentwurfs vorgenommen. Sie sollen im Folgenden als Grundlage für die Darstellung der für das Verständnis der vorliegenden Arbeit essenziellen Begrifflichkeiten dienen. Ergänzend werden weitere, begriffsspezifische Quellen hinzugezogen und in eine für den weiteren Verlauf gültige, vereinheitlichte Definition überführt.

Allgemeiner Problemlösungsprozess

Einer der Grundpfeiler der Ingenieurwissenschaft ist die Annahme, dass der Konstruktionsprozess durch Modelle zumindest teilweise abbildbar und dadurch systematisierbar ist [Hubka, 1984, S. 21]. Ein sehr generischer Problemlösungsalgorithmus wird in der VDI-Norm 2221 beschrieben und ist inhaltlich vergleichbar einer Vorgehensweise, die dem berühmten Physiker FEYNMAN zugesprochen wird. Beide Herangehensweisen sind in Abbildung 2.4 aufgeführt.

1. Write down the problem.
2. Think very hard.
3. Write down the answer.

(a) „FEYNMAN-Algorithmus“

1. Zielsuche
2. Lösungssuche
3. Lösungsauswahl

(b) [VDI 2221, S. 18]

Abbildung 2.4: Allgemeiner Problemlösungsprozess

Auch wenn es sich bei FEYNMAN vielleicht noch mehr um ein Bonmot handelt, sind die meisten sogenannter allgemeiner Problemlösungsalgorithmen sehr abstrakt gehalten und dadurch auch wenig bis gar nicht praxisrelevant. ASHBY schreibt in diesem Zusammenhang „*[...] es ist immer möglich, es linear und logisch aussehen zu lassen (und viele Bücher tun dies), aber die Realität ist nicht so*“ [Ashby, 2005, S. 16]. Aus diesem Grund sind verfeinerte Modelle und Vorgehensweisen entwickelt worden, die eine erhöhte Relevanz für die Praxis aufweisen. Dies setzt die Existenz einer allgemeinen Modelltheorie voraus, die im Folgenden beschrieben wird.

Allgemeine Modelltheorie

Die allgemeine Modelltheorie geht zurück auf STACHOWIAK [Stachowiak, 1973]. Er definiert ein Modell anhand dreier charakteristischer Merkmale:

1. Das *Abbildungsmerkmal*: Ein Modell ist Repräsentation eines natürlichen oder künstlichen Originals, das auch selbst ein Modell sein kann. [Stachowiak, 1973, S. 131]

2. Das *Verkürzungsmerkmal*: Nicht alle Eigenschaften des Originals werden durch das Modell berücksichtigt, sondern nur solche von Relevanz für den Anwendungsfall. [Stachowiak, 1973, S. 132]

3. Das *pragmatische Merkmal*: Die Zuordnung eines Originals zu einem Modell ist nicht bijektiv. Die Repräsentationsfunktion eines Modells gilt nur unter speziellen Einschränkungen für wen, wann und wozu das Modell konzipiert wurde. [Stachowiak, 1973, S. 132, 133]

Auf dieser Basis wurde in [Schopper und Rudolph, 2018] folgende prägnante Definition für ein Modell in Form einer Black Box abgeleitet, die auch in der vorliegenden Arbeit fortan analog verwendet werden soll:

Definition 2.2.1: **Modell**

Ein Modell ist die formalisierte, minimale Antwort auf eine spezielle Frage [Schopper und Rudolph, 2018, S. 2].

Diese Auffassung eines Modells ist in Abbildung 2.5 illustriert.

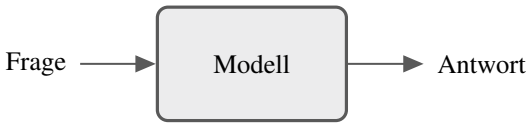


Abbildung 2.5: Modell als Black Box zwischen Frage und Antwort [Schopper und Rudolph, 2018, S. 2]

Da beim ingenieurwissenschaftlichen Produktentwurf viele Fragen gleichzeitig beantwortet werden müssen, lässt sich im Umkehrschluss aus der obigen Aussage die notwendige Existenz einer Vielzahl beschreibender Modelle ableiten. [Schopper und Rudolph, 2018, S. 2]

Produkt

Nach der Norm 24748 der Internationalen Organisation für Normung (ISO) gibt es vier verschiedene Kategorien von Produkten: Hardware (z. B. mechanisches Motorteil), Software (z. B. Computerprogramm), Dienstleistungen (z. B. Transport) und verarbeitete Materialien (z. B. Schmiermittel). Dabei handelt es sich bei Hardware und verarbeiteten Materialien im Allgemeinen um materielle Produkte, während Software oder Dienstleistungen im Allgemeinen immateriell sind. [ISO 24748, S. 6]. In dieser Arbeit wird unter dem Begriff Produkt nur ein Begriff aus der Kategorie Hardware und Software verstanden. Die VDI-Norm 2223 verknüpft das Produkt mit dem zugehörigen Entstehungsprozess und definiert es als „*Erzeugnis, das als Ergebnis des Entwickelns und Konstruierens hergestellt oder angewendet wird.*“ [VDI 2223, S. 8]. Daraus lässt sich folgende in dieser Arbeit gültige Definition für ein Produkt ableiten:

Definition 2.2.2: **Produkt**

Ein Produkt ist ein aus einem Entwicklungs- und Konstruktionsprozess hervorgegangenes Erzeugnis, das materieller oder virtueller Natur sein kann.

Produktentstehungsprozess

Der Produktentstehungsprozess (PEP) ist derjenige Teil des Produktlebenszyklus, der von der Planung, über die Entwicklung bis zur Gestaltung und Herstellung eines Produktes reicht, dessen Nutzung und Deproduktion jedoch ausschließt [Pahl und Beitz, 2013, S. 23; VDI 2221, S. 25]. In der klassischen Darstellung nach PAHL und BEITZ werden dabei die Schritte

- Planen und Klären der Aufgaben,
- Konzipieren,
- Entwerfen sowie
- Ausarbeiten

durchlaufen, wobei währenddessen zunächst die Optimierung des Prinzips und später die Optimierung der Gestalt verfolgt wird. [Pahl und Beitz, 2013, S. 17] Teilweise synonym, teilweise

leicht abweichend, aber stark verwandt im Begriffsinhalt werden in der Literatur für den PEP auch die Begriffe Entwicklungs- und Konstruktionsprozess (EKP), Produktentwicklungsprozess, Entwurfsprozess und einige weitere kombinatorische Varianten verwendet. Im Rahmen dieser Arbeit wird im Zusammenhang mit dem PEP meist vom Entwurfsprozess gesprochen, wobei hierdurch der Fokus auf die Phasen vom Beginn der Planung bis zum Ende der Gestaltung gesetzt wird. In Abbildung 2.6 sind die einzelnen Phasen des PEP gezeigt.

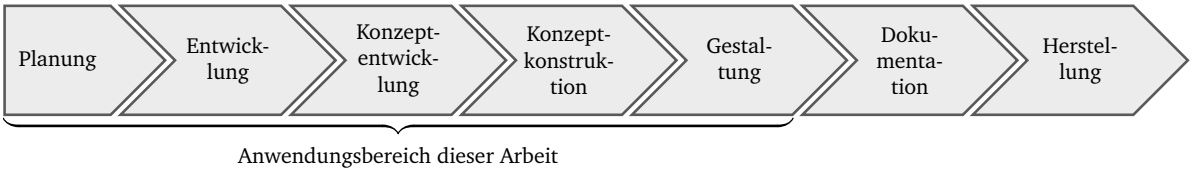


Abbildung 2.6: Produktentstehungsprozess [Pahl und Beitz, 2013, S. 23]

Der PEP wurde in der oben gezeigten Form von den Autoren in [Pahl und Beitz, 2013] aus einer Reihe sich ähnelnder Darstellungen in der Literatur allgemeingültig generalisiert, jedoch haben sich aus den verschiedenen, entwickelnden und produzierenden Branchen heraus spezielle Repräsentationsformen etabliert, die disziplinspezifische Blickrichtungen herausstellen.

In der Mechatronik hat sich die Darstellung als „V-Modell“ etabliert (siehe Abbildung 2.7).

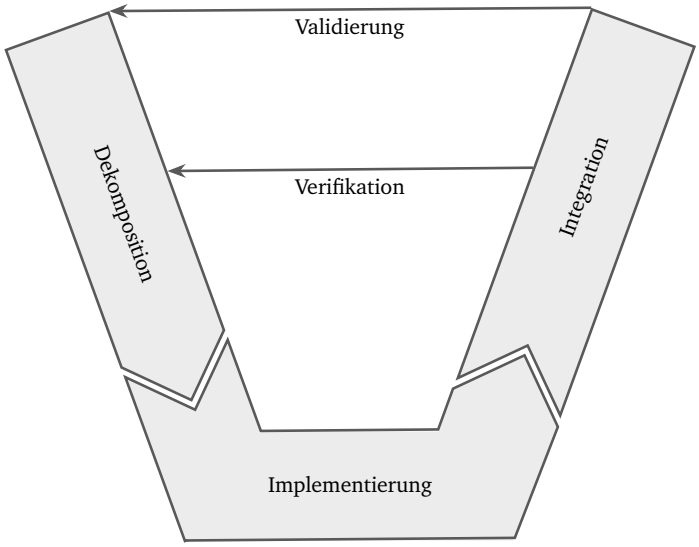


Abbildung 2.7: Produktentstehungsprozess als V-Modell [VDI 2206, S. 20]

Dabei werden die Hauptprozesse hier in chronologischer Reihenfolge als Dekomposition, Implementierung und Integration bezeichnet. Durch die V-Form ist die Möglichkeit einer Verifizierung des Entwurfs durch Abgleich der integrierten Teilsysteme mit der im Verlauf der Dekomposition getroffenen Aufspaltung in Subsysteme und die Validierung des fertigen Produkts gegenüber den ursprünglichen Anforderungen besonders einfach motivierbar. [VDI 2206, S. 22] Diesem Testmechanismus kommt im Umfeld der Mechatronik eine besondere Bedeutung zu [Walter, 2021]. In der Raumfahrt wird der PEP in von A bis F durchgezählte Phasen[5] unterteilt, die

[5]Der Phase A ist noch eine Phase 0 vorgelagert.

partiell mit generischen Meilensteinen – sog. Reviews – verknüpft werden [ECSS-M-ST-10C]. In der anglo-amerikanischen Literatur werden in Bezug auf den PEP oft [Ulrich et al., 2020] zitiert, wobei sich deren Darstellung weitgehend mit [Pahl und Beitz, 2013] deckt. Diese Deckung gilt analog für die weithin anerkannte VDI-Norm 2221 [VDI 2221].

Wie in Abbildung 2.6 durch die geschweifte Klammer kenntlich gemacht wurde, umfasst der Gültigkeitsbereich des im Rahmen der vorliegenden Arbeit entwickelten Entwurfsrahmenwerk einen Großteil des PEP. Ein Hauptaugenmerk liegt dabei auf der Automatisierbarkeit bzw. Automatisierung des Entwurfsprozesses an sich. Ohne ein grundlegendes Verständnis der Vorgänge beim Entwerfen kann eine solche Automatisierung nicht gelingen. Aus diesem Grund sollen aus den zahllosen Beschreibungen in der Literatur im Folgenden einige aus der Sicht des Autors der vorliegenden Arbeit besonders wichtige Erkenntnisse herausgegriffen werden. Darauf basierend wird eine für diese Arbeit gültige Definition für den Entwurfsprozess abgeleitet.

In [Hubka, 1984, S. 16–18] sind einige Definitionen des Entwurfsvorgangs diverser internationaler Autoren zusammengestellt. In [Naefe, 2018, S. 32] wird der Konstruktionsvorgang als Informationsumsatz beschrieben, bei dem die „*Eingangsgröße ‚Aufgabenstellung, Planung o.ä.'*“ in einer „*durch die Umgebung (Markt, Vertrieb, etc.)*“ beaufschlagten Black Box in „*die Ausgangsgröße ‚technische Unterlagen (Fertigungszeichnung, Stückliste, Bedienungsanleitung)'*“ überführt wird, die zur „*Fertigung eines ‚technischen Systems (Produkt)'*“ führt.

In [Pahl und Beitz, 2013, S. 14] wird der Konstruktionsprozess als Sequenz von Modelltransformationen beschrieben, wobei „*die Überführung des Modells (M_n) der Phase n in das Modell (M_{n+1}) der Phase $n+1$ unter Anwendung der Methoden (AS_n) und der Produktdaten (P_n)*“ erfolgt. Und weiter „*Dabei werden die Produktdaten (P_{n+1}) erzeugt*“. In [Ehrlenspiel und Meerkamm, 2017, S. 265] wird das Entwerfen als *[. . .] die Phase, in der das qualitativ funktionierende Produkt aus der Form eines Konzepts in ein quantitativ funktionsfähiges und fertigbares, körperlich gestaltetes Gebilde überführt wird*. In [Maier und Störrle, 2011, S. 2–3] wird der Konstruktionsprozess mit den Eigenschaften unklar definiert und strukturiert, komplex und iterativ in Verbindung gebracht. Aus den vorgestellten Definitionen und Beschreibungen lässt sich folgende für diese Arbeit gültige Definition des Entwurfsprozesses ableiten:

Definition 2.2.3: **Entwurfsprozess**

Der Entwurfsprozess ist ein komplexer und iterativer Prozess, bei dem Anforderungen in einer Sequenz von Modelltransformationen in ein Produkt überführt werden.

Entwurfsarten

In der klassischen Entwurfsphilosophie nach PAHL und BEITZ [Pahl und Beitz, 2013] – und darauf aufbauend auch in vielen weiteren Werken analog oder in ähnlicher Form übernommen – wird eine Konstruktion in die drei unterschiedliche Spielarten Neu-, Anpassungs- und Variantenkonstruktion untergliedert. Hierdurch soll den unterschiedlichen Herausforderungen und Anforderungen an den/die Konstrukteur/in je nach Aufgabenart Rechnung getragen werden.

Neukonstruktion Bei der Neukonstruktion werden neue Aufgaben- und Problemstellungen gelöst, indem Funktionen, Wirk- oder Lösungsprinzipien neu erarbeitet oder neu kombiniert werden. Dies betrifft alle Phasen des Entwurfs vom Konzipieren über das Entwerfen bis hin zum Ausarbeiten in gleicher Weise. [VDI 2223, S. 88; Ehrlenspiel und Meerkamm, 2013, S. 270; Pahl und Beitz, 2013, S. 293] In [Ehrlenspiel und Meerkamm, 2013] wird auf eine große Bandbreite in der Kategorisierung der Neukonstruktion hingewiesen, die von simplen Produkten bis hin zu innovativen Neukonstruktionen, wie der Entwicklung des ersten Düsentriebwerks, reicht und zitieren in diesem Zusammenhang den Begriff der *Fortschrittskonstrukion*.

Anpassungskonstruktion Bei der Anpassungskonstruktion ist das Lösungsprinzip fest vorgegeben, sodass im weiteren Konstruktionsprozess lediglich Gestalt, Werkstoff und Abmessungen an geänderte Randbedingungen angepasst werden müssen. Teilweise involviert die Anpassungskonstruktion eine Neukonstruktion von Einzelteilen oder Baugruppen. [Pahl und Beitz, 2013, S. 293; Ehrlenspiel und Meerkamm, 2013, S. 271; VDI 2223, S. 86]

Variantenkonstruktion Im Unterschied zur Anpassungskonstruktion sind bei der Variantenkonstruktion Gestalt und der Werkstoff ebenfalls festgelegt und lediglich die Größe, Position oder Anordnung von Teilen und Baugruppen variiert. Die Variantenkonstruktion basiert häufig auf der Erstellung von Baukästen oder der Ableitung von Baureihen. [Pahl und Beitz, 2013, S. 293; Ehrlenspiel und Meerkamm, 2013, S. 272; VDI 2223, S. 89] Nach [Ehrlenspiel und Meerkamm, 2013, S. 272] eignet sie sich aufgrund des stark eingeschränkten Lösungsraums ausgezeichnet für eine Automatisierung.

Der im Rahmen dieser Arbeit entwickelte Framework bildet alle drei Kategorien ab, es wird allerdings Wert darauf gelegt, dass im Verständnis des Entwurfs als „Round-Trip-Aufgabe“ die Untergruppierungen keine alleinstehenden Konstruktionshergang darstellen, sondern zu unterschiedlichen Zeitpunkten in ein und demselben Entwurf ablaufen können. Dies wird auch an der Tatsache deutlich, dass die verschiedenen Konstruktionsarten sich definitorisch nicht scharf voneinander trennen lassen [Weinbrenner, 1993, S. 28]. In Kapitel 3 wird dieser Sachverhalt umfassend dargestellt und diskutiert.

Variante

Da der Begriff Variante insbesondere bei der Entwurfsraumexploration (siehe Abschnitt 2.2.2) eine besondere Rolle spielt, soll er möglichst exakt definiert werden. In der Deutsche Industrienorm (DIN)-Norm 199-1 wird eine Variante definiert als ein „*Teil ähnlicher Form und/oder Funktion mit einem in der Regel hohen Anteil identischer Baugruppen oder Teile*“ [DIN 199-1, S. 10]. Im Verständnis des Autors der vorliegenden Arbeit ist die Ähnlichkeit einer Ausprägung gegenüber einer anderen überhaupt keine ausschlaggebende Kategorie für die Definition einer Variante. Dies wird in Kapitel 3 noch mit dem zugrunde liegenden, graphenbasierten Datenmodell in Verbindung gebracht. In [Heina, 2013, S. 5] wird eine Variante losgelöst von einem Ähnlichkeitsbegriff charakterisiert durch die Abweichung in mindestens einer Merkmalsausprä-

gung von einer Grundversion. In [Franke, 2002, S. 6] wird zusätzlich der gemeinsame Zweck aller Varianten eines technischen Systems hervorgehoben. Vor diesem Hintergrund lässt sich für diese Arbeit folgende Definition einer Variante formulieren:

Definition 2.2.4: **Variante**

Eine Variante erfüllt dieselben Anforderungen in einer mindestens in einem Punkt abweichenden Merkmalsausprägung.

Anforderung

Im Rahmen der vorliegenden Arbeit wird ein Framework erarbeitet, der initiale Anforderungen automatisiert in eine Produktbeschreibung überführt, wobei sich aus Zwängen und Interdependenzen neue Anforderungen ergeben können. Aus diesem Grund muss zunächst überhaupt definiert werden, was unter einer Anforderung im Rahmen dieser Arbeit zu verstehen ist.

In der DIN-Norm 69901-5 wird eine Anforderung als „*Beschaffenheit, Fähigkeit oder Leistung, die ein Produkt, Prozess oder die am Prozess beteiligte Person erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen*“ [DIN 699501-5, S. 6] beschrieben. In [Eigner et al., 2014] wird eine Anforderung aufgefasst als „*eine Aussage über die geforderte oder gewünschte Eigenschaft. Dies kann die Funktionalität betreffen, die das Produkt oder das System zu leisten hat (funktionale Anforderung) oder ein Qualitätsmerkmal (nicht-funktionale Anforderung), eine Bedingung oder Fähigkeit, die von einer Person zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.*“ [Eigner et al., 2014, S. 57] In [IEEE, 1990, S. 62] wird des Weiteren zwischen Entwurfsanforderungen, funktionalen Anforderungen, Implementierungsanforderungen, Schnittstellenanforderungen, Leistungsanforderungen und physikalischen Anforderungen unterschieden. Im Rahmen dieser Arbeit gilt auf dieser Basis folgende Definition einer Anforderung:

Definition 2.2.5: **Anforderung**

Eine Anforderung ist eine von außen gegebene oder durch innere Zwänge abgeleitete, geforderte Eigenschaft eines technischen Systems, die in Bezug auf die Funktionalität, Implementierung, physikalische Ausprägung oder weiterer Qualitätsmerkmale zur Erreichung eines Ziels formalisiert werden kann.

Funktion

In der VDI-Norm 2221 wird die Funktion als „*allgemeiner und gewollter Zusammenhang zwischen Eingang und Ausgang eines Systems mit dem Ziel, eine Aufgabe zu erfüllen*“ [VDI 2221, S. 6] umschrieben. In [Pahl und Beitz, 2013] wird präzisiert, dass es sich sowohl bei den Eingangs-

als auch bei den Ausgangsgrößen um die Kategorien Energie, Stoff und Signal handelt [Pahl und Beitz, 2013, S. 240]. In [Haberfellner et al., 2019] wird die Funktion bezogen auf das, was ein System tut, und damit auf den Nutzen oder den Wert, den eine Lösung oder ein Lösungsansatz bringen soll [Haberfellner et al., 2019, S. 159]. GERO verknüpft Form, Verhalten und Struktur zu einem Entwurfsartefakt, das er als Design Prototype bezeichnet [Gero, 1990]. Dies führt auf folgende, im weiteren Verlauf dieser Arbeit gültige Definition einer Funktion:

Definition 2.2.6: **Funktion**

Eine Funktion ist ein abstrakter Transformationsvorgang von Energie, Stoff oder Signal durch einen Funktionsträger zur Umsetzung eines gewünschten Verhaltens.

Nach PAHL und BEITZ kann die Funktion auf unterschiedlichen Abstraktionsniveaus definiert werden, und eine Unterteilung einer Gesamtfunktion in Teilfunktionen in Form einer Funktionsstruktur vorgenommen werden [Pahl und Beitz, 2013, S. 244, 345].

Danach können die geforderten Funktionen in einen Wirkzusammenhang mit Funktionsträgern, die auf völlig unterschiedlichen Prinziplösungen beruhen können, gebracht werden [Pahl und Beitz, 2013, S. 251]. Die Umsetzung der Funktionsträger erfolgt durch Komponenten, die modular gekapselt oder untereinander vernetzt eingesetzt werden können. Komponenten können zu Baugruppen und diese wiederum zum Gesamtprodukt assembliert werden. [Pahl und Beitz, 2013, S. 258–260] Dieser Vorgang ist in Abbildung 2.8 grafisch aufbereitet.

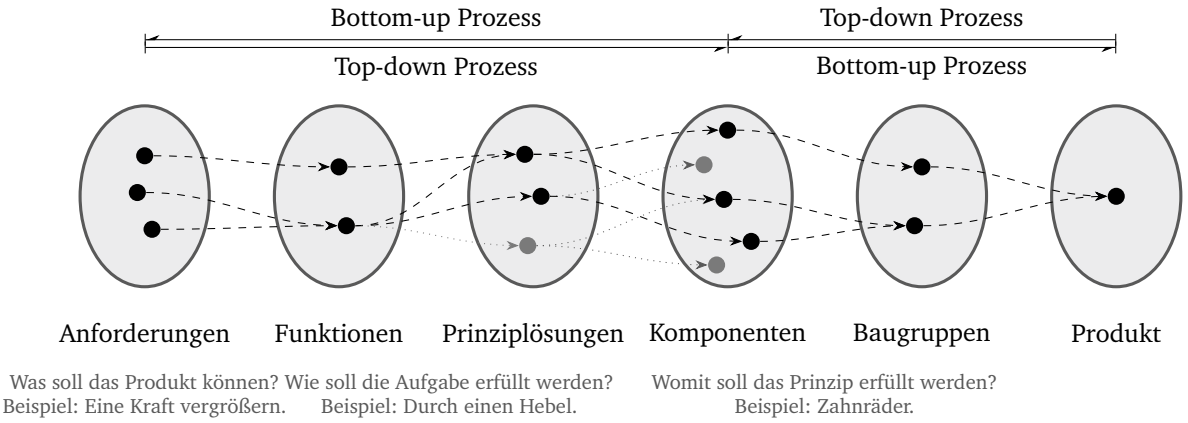


Abbildung 2.8: Modellierungskategorien und Transformationen nach PAHL und BEITZ, Darstellung basierend auf [Pahl und Beitz, 2013, S. 464] und [Groß, 2014, S. 20]

Das dargestellte Schema verfolgt einen Top-down Prozess von den Anforderungen über die Funktionen zu den Prinziplösungen und Komponenten, der dem Dekompositionsast des V-Modells (siehe Abbildung 2.7) entspricht. Anschließend werden in einem Bottom-up Prozess die Komponenten zu Baugruppen und diese schließlich zum Gesamtprodukt aggregiert (Integrationsast des V-Modells). Da es sich beim Entwerfen um einen iterativen Prozess handelt, können die Schritte auch in nicht-chronologischer und entgegengesetzter Reihenfolge durchlaufen

werden. Dieser Sachverhalt wird im Zusammenhang mit der Einführung des Reverse- und Re-Engineering in Kapitel 3 im Detail diskutiert.

Im unteren Teil von Abbildung 2.8 sind die Fragestellungen zu den einzelnen Schritten aufgeführt. Die Anforderungen beantworten die Frage nach dem „Was“, die Funktionen und Lösungsprinzipien nach dem „Wie“ und die Komponenten und Baugruppen nach dem „Womit“.

Die Struktur der Zusammenhänge, die bei der Abfolge der Schritte entsteht, wird als Produktarchitektur bezeichnet [Pahl und Beitz, 2013, S. 257]. Diese beinhaltet sowohl die Funktionsstruktur, die Produktstruktur sowie alle Entscheidungen und Festlegungen der Produktentstehung. In der Literatur wird die Summe dieser Informationen mitunter auch als Produktlogik oder Produkthergang bezeichnet [Weinbrenner, 1993; Malmqvist, 1997].

In der obigen Transformationssequenz wird die Überführung der Prinziplösungen in Komponenten als Gestaltungsphase bezeichnet. Diese hat in der Konstruktionsmethodik eine besondere Bedeutung, da hier für ein materielles Produkt aus einem Konzept ein geometrisch festgelegter Körper entsteht. Die Methodik des schrittweisen Gestaltens ist ein weit verbreiteter Ansatz zur Formalisierung dieses Prozesses [Pahl und Beitz, 2013, S. 479–491]. Dabei wird das Wirkkonzept (Prinziplösung) mit dem Gestaltungskonzept zu einem Wirkflächenpaar zusammengefasst [Pahl und Beitz, 2013, S. 480]. Für eine genauere Darstellung sei auf die Arbeiten von [Matthiesen, 2002] und [Lemburg, 2009] verwiesen.

In [Vogel, 2016] im Kapitel über „Ordnungsmechanismen im wissensbasierten Entwurf“ wird zum ersten Mal der Gedanke eingebracht, den Entwurf als Analogon eines *nicht linearen Systems* aufzufassen. VOGEL kommt zu dem Schluss, dass die Reihenfolge der Entwurfsschritte allein durch die Dimension des zugehörigen Entwurfsraums determiniert ist und diese anhand mathematischer Kriterien ableitbar sei. Aus Sicht des Autors der vorliegenden Arbeit wird diese Auffassung prinzipiell geteilt, der Fokus aber verstärkt auf die explizit formulierten, bekannten und unbekannten Randbedingungen und deren Einfluss auf die Entwurfsabfolge gelegt. Dies wird in Kapitel 3 detailliert beschrieben. Wichtig ist in diesem Zusammenhang die Bewusstmachung, dass das Wort Funktion (siehe Definition 2.2.6) aus der Mathematik entliehen ist, um einen Transformationsvorgang im Ingenieurwesen zu beschreiben.

2.2.2 Entwurfsraum und Entwurfsraumexploration

Eine sehr allgemeine Definition des Entwurfsraums ist, dass es sich um „*ein Produkt möglicher Entwurfsentscheidungen*“ [Saxena und Karsai, 2010, S. 1] handelt. Einige Autoren reduzieren den Begriffsinhalt auf den Raum formal gültiger Entwürfe[10] [Antonsson und Cagan, 2001; Haibing Li, 2020; Gembarski und Lachmayer, 2018], welcher aus Sicht des Autors der vorliegenden Arbeit allerdings trefflicher mit dem Wort Lösungsraum beschrieben wird.

In diesem Sinne wird der universelle Entwurfsraum zunächst initial durch konkrete Anforderungen und Zwangsbedingungen auf den Primärlösungsraum eingeschränkt und anschließend durch weitere Festlegungen während des Konstruktionsprozesses immer weiter verkleinert, bis am Ende nur noch ein gültiger Entwurf übrig bleibt. Dies deckt sich auch grundsätzlich mit den Auffassungen der Autoren des „Münchner Konretisierungsmodells“ [Ponn und Lindemann,

[10] In [Vogel, 2016] wird für formale Sprachen die syntaktische Korrektheit der Entwürfe vorausgesetzt.

2011, S. 27] und den evolutionären Arbeiten [Eigner et al., 2017, S. 75], [Pfenning, 2017, S. 142] und [Gembarski, 2018, S. 72], wobei unter anderem noch eine weitere Unterteilung des Lösungsraums in funktionale, logische und physikalische Ebenen unterteilt wird.

Der Begriff des Entwurfsraums wird im Kontext der Entwurfsformalisierung und -automatisierung oft verwendet, um die Komplexität einer Entwurfsaufgabe zu quantifizieren. Die Entwurfsraumgröße allein ist jedoch kein hinreichendes Merkmal für den Komplexitätsgrad, da beispielsweise auch einfache parametrische Modelle auf einen mit der Anzahl der Parameter über alle Grenzen hinaus wachsenden Entwurfsraum führen können: „*Geht man beispielsweise davon aus, dass ein zu konstruierendes Aggregat aus n verschiedenen Einzelteilen besteht und jedes dieser Einzelteile wiederum in m Varianten ausgeführt werden kann, ergeben sich daraus $z = m^n$ verschiedene Kombinations- bzw. Lösungsmöglichkeiten.*“[Wittel et al., 2019].

Diese parametrische Komplexität tritt auch bei der „Mass Customization“ zutage, bei der aus einer vordefinierten Menge eine individuelle Lösung auswählt wird. Der Lösungsraum ist dabei potenziell groß, aber, da alle Varianten a priori bekannt sind, stabil [Salvador et al., 2009].

Die wahre Entwurfsraumkomplexität rührt aus der Mächtigkeit, alle vorstellbaren qualitativ und quantitativ[11] unterschiedlichen Entwürfe zu beinhalten. Über die rein theoretischen Betrachtungen hinaus hängt nach [Alber und Rudolph, 2002] die Mächtigkeit des Entwurfsraums von der Repräsentationsmethode ab, denn nur wenn diese in der Lage ist, alle möglichen parametrischen, topologischen und funktionalen Einflüsse gleichzeitig zu berücksichtigen, kann ein wirklich komplexer Entwurfsraum aufgebaut und untersucht werden.

Das systematische Absuchen des aufgespannten Entwurfsraums nach Entwürfen, welche die gestellten Anforderungen und Zielkriterien optimal erfüllen, wird in der Literatur als Entwurfsraumexploration bezeichnet [Kang et al., 2011, S. 34]. Eine frühe Repräsentationsform des topologischen Entwurfsraums ist der morphologische Kasten [Zwicky, 1971], bei dem durch Permutation die gesamte kombinatorische Vielfalt unterschiedlicher Entwurfsvarianten ermittelt werden kann. GERO beschreibt in [Gero, 1990] verschiedene ineinander gebettete Lösungsräume und stellt eine Verbindung dieser Räume zu Standard-, innovativen und kreativen Entwürfen her. Dies ist in Abbildung 2.9 dargestellt.

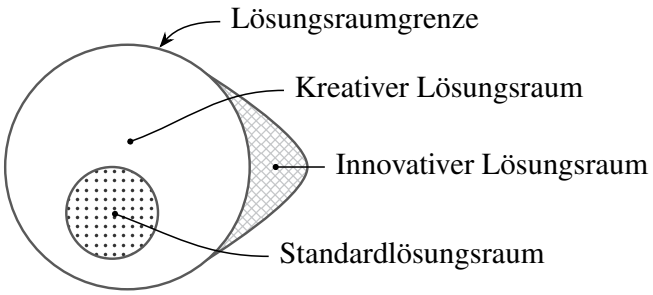


Abbildung 2.9: Lösungsräume für verschiedene Entwurfsarten und Lösungsraumgrenze nach [Gero, 1990, S. 34–35]

Die Lösungsraumgrenze ist die von der Entwicklungsumgebung umfasste Modellierungsmächtigkeit. Wichtig ist, dass eine innovative Lösung die ursprüngliche Lösungsraumgrenze vergrößert

[11] Darunter fallen alle topologisch und parametrisch unterschiedlichen Varianten

und somit in einen ursprünglich nicht vom Lösungsraum abgedeckten Bereich vorstößt. Dieser Vorgang ist in der realen Umsetzung wie bereits erwähnt von der Repräsentationsmethode abhängig, die eine solch flexible Grenzverschiebung konzeptionell vorsehen muss.

RUDOLPH charakterisiert drei ineinander gebettete Begriffsgebiete, die beim ingenieurwissenschaftlichen Entwurf überstrichen werden: Glauben, Können und Wissen [Rudolph, 2002, S. 98]. Diese sind in Abbildung 2.10 grafisch dargestellt.

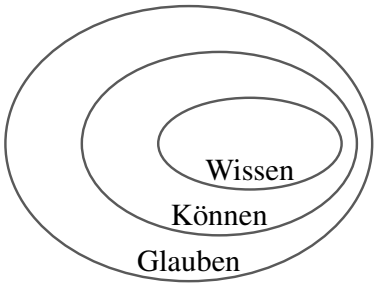


Abbildung 2.10: Einbettung von Glauben, Können und Wissen [Rudolph, 2002, S. 98]

Wissen Zur Kategorie des Wissens zählen alle Begründungsweisen, deren „*Ergebnisse auf Theorien und Gesetzmäßigkeiten beruhen, die im naturwissenschaftlichen Sinne konsistent, beweisbar, oder deren Experimente messbar und reproduzierbar sind*“ [Rudolph, 2002, S. 99]. Diese Eigenschaften gelten, in den gut erforschten und weitestgehend formalisierten Bereichen der Wissenschaft wie der Physik und des Ingenieurwesens. [Rudolph, 2002, S. 99]

Können Unter dem Begriff des Könnens werden alle Begründungsweisen kategorisiert, die sich einer formalen Beweisbarkeit entziehen, aber trotzdem ihre Nützlichkeit bereits in der Anwendung unter Beweis gestellt haben. Hierunter zählen auch diverse Konstruktionsrichtlinien, -vorschriften und -handlungsanweisungen. [Rudolph, 2002, S. 99]

Glauben Der Bereich des Glaubens als Begründungsweise stützt sich auf freiwillig auferlegte, abstrakte, philosophische und kulturelle Grundideen und Ideale zur Orientierung, Werterhaltung und Wertbestimmung. Im Kontext des Entwurfs können darunter die Einfachheit, Mächtigkeit, Angemessenheit, Eleganz und Schönheit verstanden werden und als Bewertungskategorie zu Rate gezogen werden. [Rudolph, 2002, S. 99]

Die Idee eines Produktes entstammt demnach der Kategorie des Glaubens und wird über einen Lösungsfindungs- und Syntheseprozess in die Bereiche des Könnens und auch des Wissens überführt. Im Verlauf des Entwurfsprozesses kann es durch die bereits angesprochenen Ambiguitäten und weitere Effekte zu einem mehrfachen Wechsel zwischen den Begründungsgebieten kommen. SUH spricht in einem ähnlichen Zusammenhang vom „Zick-Zack“-Vorgehen [Suh, 1990]. Die drei genannten, dem Entwurf inhärenten Begriffsgebiete werden in der Philosophie des Round-Trip-Engineering (siehe Kapitel 3) ebenfalls reflektiert.

2.2.3 Wissensbasierter Ingenieurentwurf

Da sich die vorliegende Arbeit im Kontext des modellbasierten Ingenieurentwurfs bewegt, soll zunächst die Bedeutung des Begriffs Wissen dargelegt und eine Abgrenzung gegenüber den verwandten Begriffen Daten und Information vorgenommen werden. Anschließend soll der aktuelle Stand der Technik in Bezug auf die Formalisierung, Repräsentation und Anwendung von Wissen in Entwurfssystemen umrissen werden.

Daten, Information und Wissen

Daten sind zunächst reine Symbole, die ohne Kontext noch nicht interpretierbar sind. Dabei kann es sich um beliebige Zeichen oder Zeichenabfolgen handeln. [North, 2011, S. 36; VDI 5610-1, S. 4] Informationen unterscheiden sich von Daten durch eine Struktur und einen zugehörigen Bedeutungskontext. Sie können in Kategorien unterteilt und Grundlage für Berechnungen sein. [North, 2011, S. 36; VDI 5610-1, S. 4] Wissen ist die einem Zweck dienliche Zusammenführung von Information und kapselt dadurch spezielle Kenntnisse und Fähigkeiten zur Problemlösung. Auf dieser Basis können Vergleiche initiiert, Verknüpfungen hergestellt und Entscheidungen abgeleitet werden. Darüber hinaus wird zwischen an Personen geknüpftem *impliziten* und formalisierbarem *expliziten Wissen* unterschieden. [North, 2011, S. 36; VDI 5610-1, S. 4]

In Abbildung 2.11 ist die Beziehung der drei Begrifflichkeiten noch einmal dargestellt.

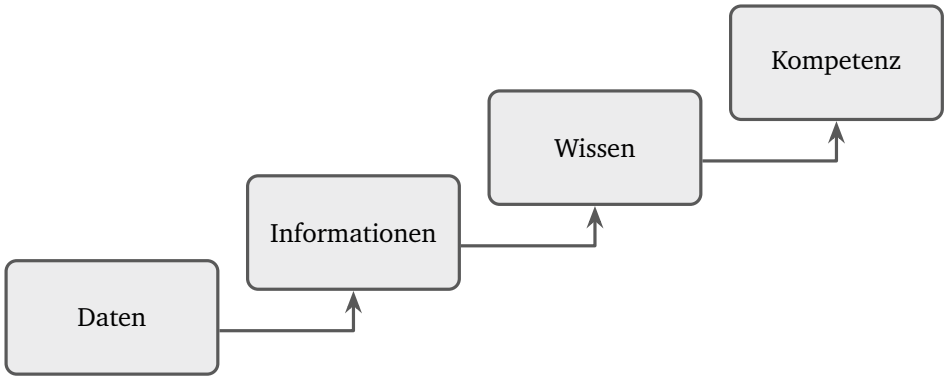


Abbildung 2.11: Wissenstreppe: Zusammenhang zwischen Daten, Information, Wissen und Kompetenz, vereinfachte Darstellung angelehnt an [North, 2011, S. 40]

Zusätzlich ist in der obigen Abbildung oberhalb des Wissens der Begriff Kompetenz aufgeführt, wodurch betont werden soll, dass nur auf Grundlage einer fundierten Wissensbasis kompetente, d. h. pragmatische, maßgeschneiderte und unter definierten Bewertungsgrößen optimale, Lösungen für eine bestimmte Problemstellung gefunden werden können [North, 2011, S. 40].

Wissensrepräsentation

Unter dem Begriff Wissensrepräsentation versteht man syntaktisch und semantisch unterlegte Strukturen zur Beschreibung von Objekten und Vorgängen [VDI 5610-2, S. 9]. Grundsätzlich kann man zwischen zwei Formen der Wissensrepräsentation unterscheiden: Die *deklarative*

Repräsentation bezeichnet das Abbilden von Faktenwissen, die *prozedurale* Repräsentation ein Modell von Handlungswissen [Styczynski et al., 2017, S. 48]. Es existiert eine Vielzahl an Unterkategorien dieser beiden Typen, die in der praktischen Anwendung gängigsten davon sind

- Regeln oder Produktionsregeln (WENN-DANN-Regeln),
- Constraints (Zwangsbedingungen),
- Data-Mining basierte Vorhersagemodelle, Regressionsgleichung und künstliche neuronale Netze,
- semantische Netze und
- Frames (Klassen oder Klassenmodelle)

[VDI 5610-2, S. 9–11; Styczynski et al., 2017, S. 48]. Im nächsten Absatz wird dargelegt, wie für eine Anwendung benötigtes Domänenwissen abgeleitet und repräsentiert werden kann.

Ableitung von Domänenwissen Ein Grundpfeiler der Wissenschaft ist die Definition von Begriffen und Bezeichnungen zur genauen Charakterisierung von Objekten und Vorgängen [Schopper und Rudolph, 2018, S. 4]. In diesem Zusammenhang wird ein Begriff als „*Denkeinheit, die aus einer Menge von Gegenständen unter Ermittlung der diesen Gegenständen gemeinsamen Eigenschaften mittels Abstraktion gebildet wird*“ [DIN 2342, S. 5] beschrieben.

Demgegenüber wird die Bezeichnung, als „*Repräsentation eines Begriffs mit sprachlichen oder anderen Mitteln*“ [DIN 2342, S. 10] verstanden. Die ingenieurwissenschaftliche Durchdringung einer Domäne kann in diesem Sinne mit der Vielfältigkeit des zugehörigen Begriffsfelds und der Quantität unterschiedlicher Bezeichnungen in Zusammenhang gebracht werden [Schopper und Rudolph, 2018, S. 5]. Zur Veranschaulichung dieser Aussage aus einem technischen Anwendungsfall heraus kann die Diskretisierung einer finiten Elemente Simulation herangezogen werden: je mehr Elemente berücksichtigt werden, desto genauer die Simulation.

Ontologien sind eine weitere, mit den semantischen Netzen und Frames verwandte Möglichkeit der Wissensrepräsentation. Ein substanzieller Teil des Domänenwissens kann daher in Form von Ontologien, die Begriffe und Bezeichnungen ordnen, formalisiert werden [Schopper und Rudolph, 2018, S. 5]. Nach GRUBER ist eine Ontologie „*eine explizite Spezifikation einer Konzeptualisierung*“ [Gruber, 1993, S. 199]. Konkret bedeutet dies, dass eine Ontologie eine formale, explizite Beschreibung einer Menge zusammenhängender Repräsentationsobjekte durch Klassen darstellt. Deren Eigenschaften und Beziehungen untereinander, wie beispielsweise die Vererbung, kodieren zusätzliches Wissen [Noy und McGuinness, 2001, S. 3]. Ontologien enthalten formale Inferenz- und Integritätsregeln, die die Interpretation steuern und die Wohlgeformtheit garantieren [Gruber, 1993, S. 199]. Daraus lässt sich folgende Definition formulieren, die fortan bei Verwendung des Begriffs Ontologie in der vorliegenden Arbeit gültig sein soll:

Definition 2.2.7: **Ontologie**

Eine Ontologie ist eine formale Beschreibung einer Menge zusammenhängender Repräsentationsobjekte unter Berücksichtigung von Inferenz- und Integritätsregeln.

Ontologien können in verschiedene Sprachen und Formaten wie z. B. dem Knowledge Interchange Format (KIF)[13], der Web Ontology Language (OWL)[14] oder dem Resource Description Framework (RDF)[15] implementiert werden [Kalibatiene und Vasilecas, 2011, S. 5].

In der vorliegenden Arbeit werden Ontologien in Form von Klassendiagrammen zur Modellierung des für den Framework benötigten Domänenwissens eingesetzt (siehe Kapitel 4). Eine Möglichkeit eine Ontologie abzuleiten, ist die formale Begriffsanalyse.

Formale Begriffsanalyse Die Formal Concept Analysis (FCA), zu deutsch formale Begriffsanalyse, ist eine mathematische Theorie, die Ende der 1970er Jahre im Fachbereich Mathematik der Technischen Hochschule Darmstadt entwickelt wurde [Ganter und Wille, 1996, S. 58]. Sie zielt darauf ab, inhaltliche und strukturelle Abhängigkeiten beliebiger Daten zu analysieren und in eine hierarchische Struktur zu überführen [Schopper und Rudolph, 2018, S. 3].

Ein Begriff wird dabei als gedankliche Einheit des Begriffsinhalts (Intension) und des Begriffsumfangs (Extension) aufgefasst, wobei die Intension der Menge aller unter dem Begriff zusammengefassten formalen Merkmale entspricht und die Extension der Menge aller Objekte, die unter den Begriff fallen [Ganter et al., 2000, S. 126, 245].

Die systematische Anwendung der FCA zur Ordnung von Begriffen eines Anwendungsbereichs führt auf eine Taxonomie. Aufgrund der mathematischen Fundierung der FCA kann für jede beliebige Menge eine Halbordnung erzeugt und dadurch strukturelles Wissen auf formale Weise abgeleitet werden [Ganter und Wille, 1996, S. 2; Schopper und Rudolph, 2018, S. 3].

Der Formalismus der logischen Schlussfolgerung auf mathematisch formulierten Begriffsmengen kann anhand des folgenden Beispiels veranschaulicht werden: Wird festgestellt, dass die Intension eines Begriffs A – z. B. Lebewesen – in der Intension eins Begriffs B – z. B. Mensch – vollständig enthalten ist, so bedeutet dies, dass der Mensch alle Eigenschaften eines Lebewesens besitzt – z. B. atmet und ist sterblich –, nicht jedoch vice versa. Daraus kann geschlossen werden, dass die Extension des Menschen – z. B. Max und Lisa – in der Extension von Lebewesen enthalten ist. In anderen Worten muss der Begriff Mensch ein Unterbegriff von Lebewesen und Lebewesen somit auch ein Oberbegriff von Mensch sein.

In [Schopper und Rudolph, 2018] wurde gezeigt, wie das mittels FCA abgeleitete Domänenwissen über im Rahmen von Entwurfssprachen deklarierte Klassendiagramme maschinenlesbar gemacht werden kann. Dazu werden Begriffe und Objekte durch Klassen modelliert, die Attribute zur Abbildung der zugehörigen Merkmale besitzen. Darüber hinaus gibt es über Vererbungen oder Assoziationen Verbindungen zwischen den Klassen. Diese Datenstruktur ermöglicht es, von einem abstrakten Begriff über die bekannten „Erweiterungen“ zu konkreten Objekten, die sich als Blätter der Ontologie darstellen, zu navigieren und auf konkrete Eigenschaften zu schließen. Insgesamt bilden die Klassen und Beziehungen ein Klassendiagramm, das den formal modellierten Teil der Domänensemantik abbildet. [Schopper und Rudolph, 2018, S. 3] Eine mittels FCA abgeleitete Ontologie wird fortan als *objektorientierte Ontologie* bezeichnet.

[13] `http://logic.stanford.edu/kif/`
[14] `https://www.w3.org/OWL/`
[15] `https://www.w3.org/RDF/`

Wissensbasierte Konstruktionssysteme

Formalisierte Entwurfssysteme zur Unterstützung des modellbasierten Ingenieurentwurfs – auf Englisch Knowledge-based Engineering (KBE)-Systeme – stellen hilfreiche Methoden und Werkzeuge zur Erfassung und Wiederverwendung von Produkt- und Prozesswissen auf integrierte Weise bereit [Stokes, 2001; VDI 5610-2, S. 2].

LaRocca präzisiert, dass durch Automatisierung sich wiederholender und nicht kreativer Entwurfsaufgaben und Unterstützung der multidisziplinären Entwurfsoptimierung in allen Phasen des Entwurfsprozesses Zeit und Kosten in der Produktentwicklung reduziert werden können [LaRocca, 2011, S. 161]. Die Wissensbasis eines KBE-Systems entspricht einem expliziten und wartbaren Modell menschlicher Fachkenntnisse [Knublauch, 2002, S. 12].

Die wesentlichen Ziele bei der Anwendung von KBE-Systemen sind

- Standardisierung
- Automatisierung
- Qualitätssteigerung und -sicherung
- Schaffung von Transparenz
- Erhöhung der Kundenzufriedenheit
- Erhöhung der Mitarbeiterzufriedenheit

[VDI 5610-2, S. 4]. Im Kontext ingenieurwissenschaftlicher Anwendungen werden KBE-Konstruktionssysteme oftmals als Expertensysteme bezeichnet [Styczynski et al., 2017, S. 11].

In Abbildung 2.12 ist eine typische Architektur eines Expertensystems dargestellt.

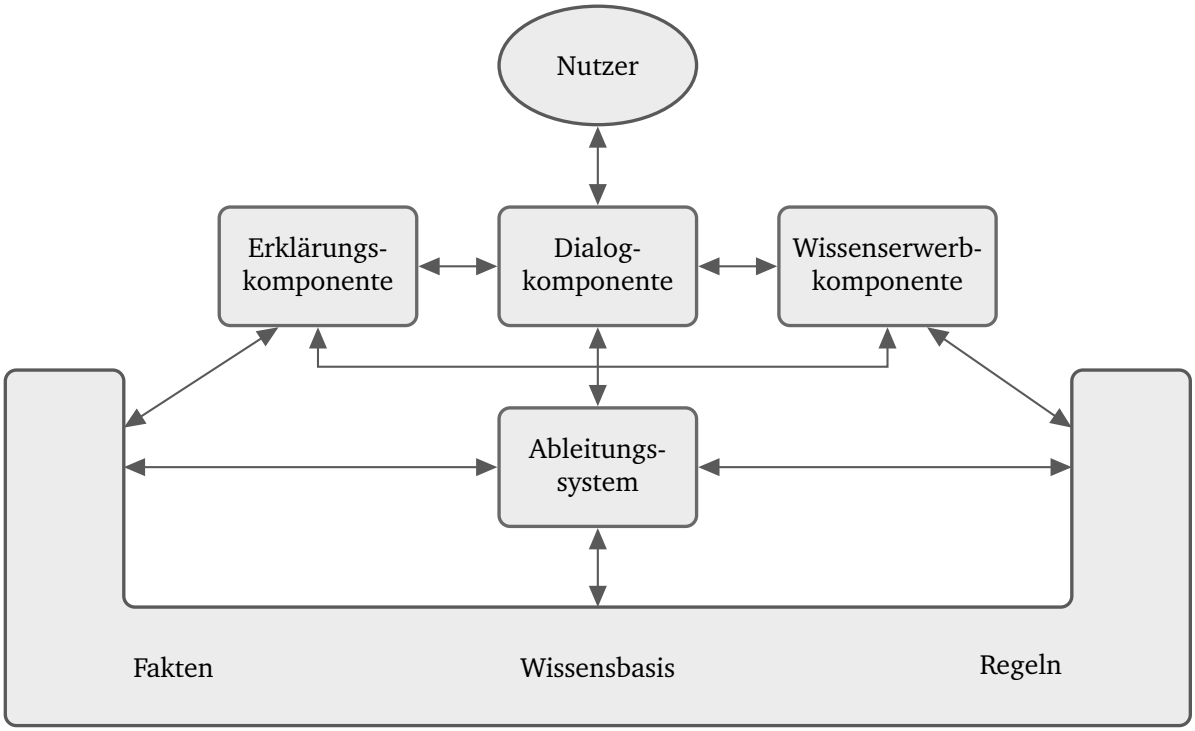


Abbildung 2.12: Architektur eines modellbasierten Expertensystems [Mainzer, 2019, S. 44]

Das Expertensystem besteht aus den fünf wesentlichen Bausteinen Wissensbasis, Problemlösungskomponente (Ableitungssystem), Erklärungskomponente, Wissenserwerb und Dialogkomponente. Die Erklärungskomponente dient der Aufklärung des Nutzers über die angewandten Untersuchungsschritte des Systems, die Dialogkomponente zur Kommunikation des Systems mit dem Nutzer. [Mainzer, 2019, S. 44]

Eine aus philosophischer Sicht äußerst wichtige Tatsache ist, dass die Wissensbasis eines Expertensystems eine spezialisierte Informationsbasis ohne allgemeines und strukturelles Wissen über die Welt darstellt [Mainzer, 2019, S. 43].

2.2.4 Systems Engineering

Im Kontext des Entwurfs komplexer Produkte wie Flugzeugen, Automobilen, Schiffen oder Raketen hat sich der Ausdruck Systems Engineering (SE) etabliert, wobei das zu entwickelnde Produkt selbst aufgrund seiner Komplexität im Zusammenspiel diverser Komponenten als System of Systems (SoS) aufgefasst wird [Haberfellner et al., 2019, S. 3, 7].

In der ISO-Norm 24748 wird SE als „*interdisziplinärer Ansatz, der den gesamten technischen und verwaltungstechnischen Aufwand regelt, der erforderlich ist, um eine Reihe von Bedürfnissen, Erwartungen und Einschränkungen der Interessengruppen in eine Lösung umzusetzen und diese Lösung während ihrer gesamten Lebensdauer zu unterstützen*“ [ISO 24748, S. 9] definiert.

Eine Spielart des SE, die im Kontext der vorliegenden Arbeit von besonderer Bedeutung ist, ist das Model-based Systems Engineering, das im nachfolgenden Abschnitt beschrieben wird.

Model-based Systems Engineering

In den vergangenen 15 Jahren hat der modellbasierte Ingenieurentwurf, auf Englisch Model-based Systems Engineering (MBSE), unter besonderem Einfluss des International Council on Systems Engineering (INCOSE) als treibender Kraft dieser Entwicklung aus dem US-amerikanischen Raum, kontinuierlich an praktischer Relevanz gewonnen.

Immer mehr Branchen übernehmen die Grundsätze, Praktiken und Erkenntnisse in ihren Prozessen [INCOSE, 2021, S. 18]. Obwohl es viele Gemeinsamkeiten gibt, ist das MBSE weniger formal definiert wie die MDA in der Softwareentwicklung (siehe Abschnitt 2.1.2).

In der INCOSE SE Vision 2020 wird MBSE beschrieben als „*die Anwendung formaler Modelle zur Unterstützung der Systemanforderungen, des Entwurfs, der Analyse, der Verifikation und der Validierung von Aktivitäten, die in der konzeptionellen Entwurfsphase beginnen und sich über die gesamte Entwicklung und spätere Lebenszyklusphasen erstrecken*“ [INCOSE, 2007, S. 15].

In einem Report über modellbasiertes Entwerfen der US-amerikanischen National Defense Industrial Association (NDIA) wird MBSE definiert als „*ein Engineering-Ansatz, der Modelle als integralen Bestandteil der technischen Basis verwendet, die die Anforderungen, die Analyse, den Entwurf, die Implementierung und die Verifikation einer Fähigkeit, eines Systems und/oder eines Produkts während des gesamten Produktlebenszyklus umfasst*“ [NDIA, 2011, S. 9].

MBSE ist ein Weg, um von einer dokumentenzentrierten hin zu einer modellbasierten Produktentwicklung zu gelangen [Haberfellner et al., 2019, S. 132]. Der pragmatische Teil des

MBSE beginnt mit der Definition der Anforderungen und endet nach einer Modellkaskade mit der vollständigen Produktdefinition [Schopper und Rudolph, 2018, S. 2].

In Abbildung 2.13 wird gezeigt, wie die Anforderungen schrittweise über eine Abfolge von Zwischenmodellen in das Endprodukt überführt werden.

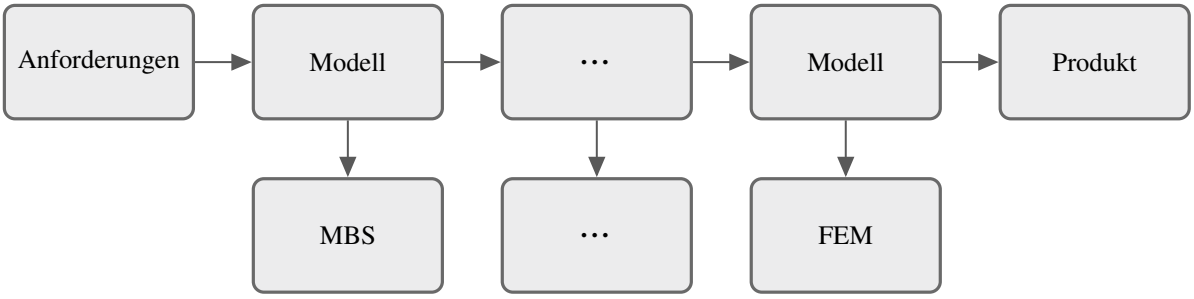


Abbildung 2.13: Modellabfolge im MBSE [Schopper und Rudolph, 2018, S. 2]

In Bezug auf die in der MDA eingeführten Transformationen (siehe Abschnitt 2.1.2), sind die horizontalen Transformationen als M2M und die vertikalen Transformationen als M2T zu interpretieren, allerdings handelt es sich bei all den zugrunde liegenden Modellen um manuell generierte, statische Artefakte [Schopper und Rudolph, 2018, S. 4]. Graphenbasierte Entwurfssprachen, die im nachfolgenden Abschnitt beschrieben werden, setzen an dieser Stelle an und erlauben eine automatische Generierung aller benötigten Modelle.

Graphenbasierte Entwurfssprachen

Graphenbasierte Entwurfssprachen kombinieren die Eigenschaften einer formalen Sprache[16] mit einem System zur Gleichungsdefinition, -reihenfolgefindung und -lösung und einer graphenbasierten Repräsentation zu einem Rahmenwerk für den automatisierten Produktentwurf.

Die erste Idee hierzu entstand Ende der 90er Jahre des letzten Jahrhunderts und geht zurück auf RUDOLPH [Rudolph, 2002]. Im Folgenden soll ein kurzer Überblick über die wichtigsten Elemente und Zusammenhänge, der für das weitere Verständnis der vorliegenden Arbeit notwendig ist, gegeben werden. Für eine detaillierte Darstellung des Themenkomplexes Entwurfssprachen sei auf [Rudolph, 2002, S. 98 – 131] und [Rudolph und Kröplin, 2005] verwiesen.

Die Modellbildung des Ansatzes beginnt mit einer Analyse des zu lösenden Entwurfsproblems, die nach einer geistigen Dekomposition – beispielsweise funktional, geometrisch oder modular – in einer Sammlung unterschiedlicher Grundentitäten (Bausteine), sowie deren Zusammenhänge und Eigenschaften mündet. Auf diese Weise können in einer Entwurfssprache Elemente formuliert werden, „*die Kenntnis ihrer eigenen Geometrie und Physik sowie weiterer relevanter Wissensbausteine besitzen.*“ [Rudolph, 2002, S. 102]. Anschließend werden die Bausteine aus dieser Wissensbasis in einem generisch formulierten, regelbasierten Rekombinationsprozess durch einen Entwurfscompiler zum gewünschten Produkt assembliert. In Abbildung 2.14 sind die wichtigsten Elemente einer Entwurfssprache und der Informationsfluss in vereinfachter Form grafisch aufbereitet.

[16] Von besonderer Bedeutung ist die maschinelle Übersetzbarkeit und Ausführbarkeit.

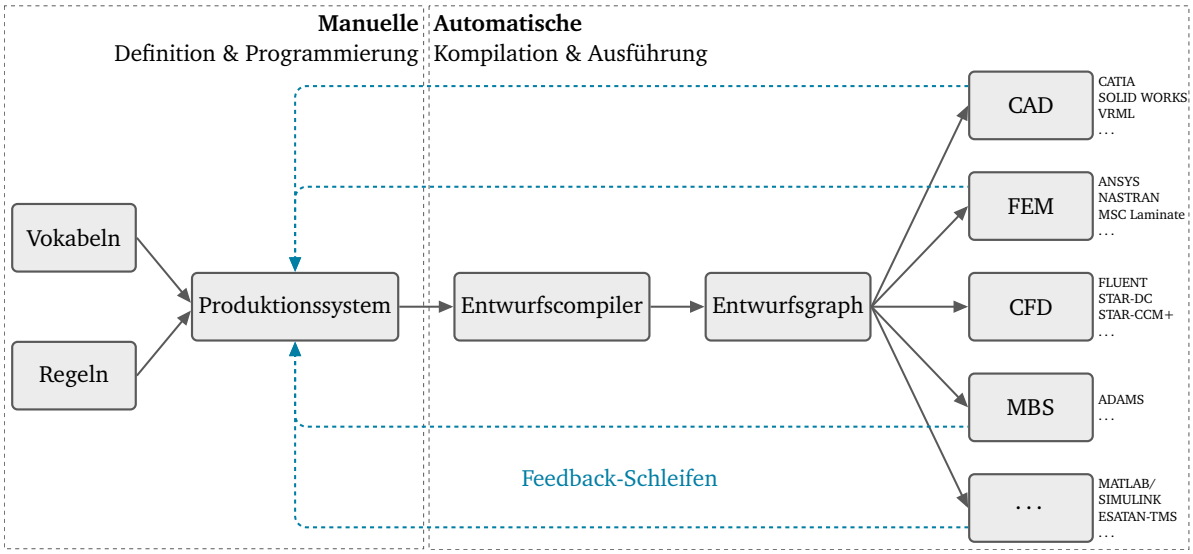


Abbildung 2.14: Informationsarchitektur graphenbasierter Entwurfssprachen, angelehnt an [Rudolph, 2002, S. 102]

Die Bausteine werden in diesem Kontext als *Vokabeln* bezeichnet, das Rekombinieren geschieht durch einen problemspezifisch formulierten Satz an *Regeln*. Durch die Kombination aus Wissensbasis und Regelwerk entsteht eine Entwurfsgrammatik, die den Entwurfsprozess insgesamt abbildet. Zusätzlich können in den Eigenschaften der Vokabeln analytische Gleichungen – sogenannte *Constraints* – hinterlegt sein. Ein im Rahmenwerk integrierter *Lösungspfadgenerator (LPG)* [Rudolph, 2002, S. 120–129] führt im Zusammenspiel mit einem Computeralgebrasystem (CAS) im konkreten Entwurfskontext auf die aus den Gleichungen berechenbaren, zunächst unbestimmten Entwurfsparameter. Beim Ausführen aller zu berücksichtigenden Regeln (*Produktionssystem*) durch den Entwurfscompiler wird das abstrakte Entwurfsmodell (*Entwurfsgraph*) mit allen Informationen und Parametern automatisch generiert. Aus dem abstrakten Modell können über Schnittstellen (*Plugins*) domänenspezifische Modelle (siehe rechte Seite in Abbildung 2.14) – wie beispielsweise ein Computer Aided Design (CAD)-, Finite Elemente Methode (FEM)-, oder Computational Fluid Dynamics (CFD)-Modell – generiert werden.

Jede Regel wird sukzessive, in der im Produktionssystem festgelegten Reihenfolge durch den Entwurfscompiler zur Laufzeit auf den aktuellen Entwurfsgraphen angewendet. Auf diese Weise können je nach Regelformulierung Elemente hinzugenommen, gelöscht oder verändert[17] werden. Dieser Vorgang kann als die in Abbildung 2.2 eingeführte, M2M-Transformation aufgefasst werden, da der Graph zu jedem Zustand einem entsprechenden Modell(-stand) zugeordnet werden kann. Im Gegensatz dazu handelt es sich bei der Modellgenerierung durch Plugins um eine M2T-Transformation, bei der die Information aus dem abstrakten Entwurfsgraph in eine domänenspezifische Sprache (Englisch Domain Specific Language (DSL)) des Zielprogramms übersetzt wird. Durch die Einführung einer Optimierungsschleife (siehe blau gestrichelte Pfeile in Abbildung 2.14) können die Ergebnisse aus den verschiedenen Domänenprogrammen zurück in das Rahmenwerk gespielt und somit Einfluss auf den Entwurf genommen werden. Der

[17] Hierbei können Eigenschaften hinzugefügt, geändert oder gelöscht werden oder der Typ des Elements auf einen im Sinne der Wissensbasis verträglichen Typ geändert werden.

kreative Aspekt des Entwerfens wird durch den Einsatz einer Entwurfssprache nicht beschränkt. In einer Entwurfssprache können analog zur natürlichen Sprache sowohl „harte“ physikalische Gesetzmäßigkeiten kodiert als auch „willkürliche“ Zusammenhänge abgebildet werden. Aufgrund der Gerichtetheit des Informationsflusses und der Ableitung aller domänenspezifischen Sichten aus einem zentralen Modell, ist die Konsistenz dieser Teilsichten zu jedem Zeitpunkt gewährleistet [Rudolph und Kröplin, 2005, S. 41]. Ein weiteres Feature bildet die sogenannte *Graphenvariation*, die durch die Einführung von *Variationsblöcken* die parallele Ausführung unterschiedlicher Regeln auf denselben Ausgangsentwurfsgraphen ermöglicht. Da der resultierende Graph nach jeder parallel anzuwendenden Regel potenziell verschieden sein kann, wird vor der Regelexekution jeweils eine Kopie des Ausgangsgraphen erstellt, die dann individuell transformiert wird und auf eine Graphenschar als Ergebnis führt. Dieser Mechanismus erlaubt eine gezielte Produktvariantengenerierung und stellt somit ein Werkzeug zur systematischen Exploration des zugehörigen Entwurfsraums zur Verfügung. Ein Begriff, der in diesem Zusammenhang für die Vorgehensweise geprägt wurde, lautet an die Methode des Vorwärtsschreitens [Pahl und Beitz, 2013, S. 288] angelehnt *systematische Vorwärtsvariation (SVV)*.

Ein wesentlicher Vorteil graphenbasierter Entwurfssprachen gegenüber anderen Formen der Entwurfsrepräsentation ist die ausreichende Mächtigkeit zur abstrakten Abbildung des gesamten Entwurfswissen und der gleichzeitigen maschinellen Verarbeitbarkeit durch die Definition als formale Sprache. Die Granularität der Auflösung des Entwurfsproblems bei der Dekomposition ist abhängig von der Modellierung der Anwenderin oder des Anwenders und kann „*[. . .] ganz entsprechend dem Vorbild natürlicher Sprachen, [. . .] auf nahezu beliebig verschiedenen Abstraktionsniveaus und Detaillierungsgraden erfolgen*“ [Rudolph und Kröplin, 2005, S. 43].

Entwurfssprachen wurden bereits in zahlreichen Projekten und Arbeiten erfolgreich zum Einsatz gebracht. Um dem Leser einen Überblick zu verschaffen, seien hier einige Arbeiten stellvertretend herausgegriffen. Eine erste Implementierung eines Entwurfscompilers wurde in [Alber und Rudolph, 2003] gezeigt. Die Entwurfsraumexploration wurde zum ersten Mal in [Rudolph und Alber, 2002] gezeigt. In [Kormeier, 2010] wird die Topologiemodifikation eines Bauteils in Faserverbundbauweise beschrieben. Die Auslegung eines Satelliten mit Entwurfssprachen wird in [Groß und Rudolph, 2015] gezeigt. Die zu einem Produkt passende automatische Auslegung einer zugehörigen Fabrikzelle wird in [Arnold und Rudolph, 2012] und später auch in [Schopper et al., 2021] vorgestellt. Die Beschreibung und Behandlung von Geometrie in Entwurfssprachen wird zum ersten Mal in [J. Schmidt und Rudolph, 2016] dargelegt. Daneben wurden Entwurfssprachen eingesetzt für den Entwurf von Flugzeugkabinen [Rudolph et al., 2013], zur Auslegung von Kaffeemaschinen [Tonhäuser und Rudolph, 2017], zur automatisierten Überführung und Auswertung von Requirements [Walter et al., 2018], zur Auslegung von Klemm- und Spannkonzepten [Zech et al., 2021], für die Fehlerbaumanalyse [Riestenpatt gen. Richter, 2021] sowie zur Gebäudeplanung [Voss et al., 2021].

Design Cockpit 43®

Das Design Cockpit 43® (DC43) ist eine Implementierung des im vorangegangen Abschnitt abstrakt beschriebenen Entwurfscompilers zur maschinellen Ausführung graphenbasierter Ent-

wurfssprachen. Entwickelt und vertrieben wird er von der Ingenieurgesellschaft für intelligente Lösungen und Systeme mbH (IILS mbH), die aus einer Gründung durch Mitarbeiter und Angehörige des Institut für Statik und Dyanmik der Luft- und Raumfahrtkonstruktionen (ISD) an der Universität Stuttgart vor mehr als 20 Jahren hervorgegangen ist. DC43 bildet die Basis für den im Rahmen des Dissertationsvorhabens beschriebenen Framework und liegt zum Zeitpunkt der Entstehung in Version drei (Kurzname „DC43V3“) vor. In DC43 sind Werkzeuge zur Definition und Programmierung, Verwaltung und Auswertung graphenbasierter Entwurfssprachen in einer Entwicklungsumgebung gebündelt. Informationstechnisch liegt die standardisierte, grafische Modellierungssprache UML als neutraler Datenstandard zugrunde. Von dieser verwendet DC43 das *Klassendiagramm*, das *Objektdiagramm* und das *Aktivitätsdiagramm* [Reichwein, 2011].

2.3 Literaturübersicht im Umfeld der vorliegenden Arbeit

In diesem Abschnitt soll zusätzlich zu den bereits in den Grundlagen zitierten Quellen eine kompakte Übersicht über die Literatur zu den wichtigsten im Rahmen der vorliegenden Arbeit überstrichenen Themenkomplexen (TK) präsentiert werden. Ob der vielfältigen Wissensgebiete, auf der die vorliegende Arbeit fußt, ist ein klassischer Ansatz mittels Defizitanalyse des Stands der Technik sehr schwer darzustellen und würde den Umfang einer Dissertationsschrift übersteigen. Aus diesem Grund sollen die wesentlichen Themengebiete kategorisiert und die aus Sicht des Autors der vorliegenden Arbeit wichtigsten Beiträge zugeordnet werden. Am Ende jedes Abschnitts sollen die Literaturbeiträge bezüglich der in Tabelle 1.1 bereits aufgestellten fünf zentralen Forschungsfragen in einer Übersichtstabelle (vgl. Tabellen 2.2 - 2.7) zugeordnet werden. Es liegt in der Natur der Sache, dass einige der aufgeführten Beiträge in mehreren TK Beiträge liefern. Sie werden in diesem Fall aber trotzdem nur in einem TK genannt.

Tabelle 2.1: In der vorliegenden Arbeit überstrichene Themenkomplexe (TK)

TK 1	Ingenieurwissenschaftlicher Entwurf
TK 2	Auswahl und Zusammenspiel von Material, Geometrie und Verbindungstechnik
TK 3	Wissensbasierte Konstruktionssysteme
TK 4	Entwurfsraum und Entwurfsraumexploration
TK 5	Forward-, Reverse- und Re-Engineering in der Informatik und im Ingenieurentwurf
TK 6	Generativer Entwurf und Entwurfsautomatisierung

TK 1 Ingenieurwissenschaftlicher Entwurf

In diesem Abschnitt soll ein Überblick über die aus Sicht des Autors der vorliegenden Arbeit relevanteste Literatur im Themenkomplex des allgemeinen Wissens über den ingenieurwissenschaftlichen Entwurfsprozess ohne Anspruch auf Vollständigkeit gegeben werden.

In der Literatur oft zitierte Standardwerke vorwiegend aus dem deutschsprachigen Raum stellen [Hubka, 1984], [Koller, 1998], [Eversheim und Schuh, 2005], [Pahl und Beitz, 2007], [Ponn und Lindemann, 2011], [Dym und Brown, 2012], [Lindemann, 2016], [Le Masson et al.,

2017],[Ehrlenspiel und Meerkamm, 2017], [Eigner et al., 2017], [Naefe, 2018], [Wittel et al., 2019], [Isaksson et al., 2019], [Ulrich et al., 2020] und die VDI-Norm [VDI 2221] dar.

Darüber hinaus sind in [Antonsson und Cagan, 2001], [Haberfellner et al., 2019], [Knöös Franzén et al., 2019] und [Bajzek et al., 2021] wichtige Beiträge zum Thema Systementwurf, System of Systems und zur automatisierten Entwurfssynthese zu finden.

Tabelle 2.2: Bewertung der untersuchten Literatur im TK 1 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[Hubka, 1984]	○	◑	○	○	○
[Koller, 1998]	○	●	○	○	○
[Eversheim und Schuh, 2005]	◑	◑	○	○	○
[Pahl und Beitz, 2007]	○	○	○	○	○
[Ponn und Lindemann, 2011]	◑	○	○	○	○
[Dym und Brown, 2012]	○	○	○	○	○
[Lindemann, 2016]	◑	○	○	○	○
[Le Masson et al., 2017]	○	◑	○	○	○
[Ehrlenspiel und Meerkamm, 2017]	◑	◑	○	○	◔
[Eigner et al., 2017]	◕	◑	○	○	○
[Naefe, 2018]	○	◔	○	○	○
[Wittel et al., 2019]	○	○	○	○	◔
[Isaksson et al., 2019]	◑	◑	○	○	○
[VDI 2221]	◔	◑	○	○	○
[Ulrich et al., 2020]	◔	◑	○	○	○
[Antonsson und Cagan, 2001]	◕	◕	◕	◑	◑
[Haberfellner et al., 2019]	◑	◑	○	○	○
[Knöös Franzén et al., 2019]	◑	◑	○	○	○
[Bajzek et al., 2021]	◑	◑	○	○	○

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

TK 2 Auswahl und Zusammenspiel von Material, Geometrie und Verbindungstechnik

In [Houldcroft, 1990] wird die Auswahl eines zur Problemstellung passenden Schweißprozesses in einem dreistufigen, formalisierten Verfahren vorgestellt. Die Basis bilden drei verschiedene Listen, die die Auswahl eines Prozesses durch den/die Anwender/-in steuern. In der ersten Liste sind 44 abstrakte Schweißstöße verzeichnet, aus der der zur Problemstellung passendste Fall ausgesucht werden muss. Bei der zweiten Liste handelt es sich um eine Matrix, die die Verträg-

lichkeit der Stöße aus der ersten Liste mit 28 verschiedenen Schweißverfahren angibt. Um aus den verbleibenden Verfahren das geeignetste auswählen zu können, enthält die letzte Liste eine detaillierte Beschreibung inklusive der Vor- und Nachteile der verschiedenen Verfahren. Die Idee des Vorgehens spiegelt sich in vielen computerbasierten Auswahlsystemen, die eine Blüte in der Zeit um die Jahrtausendwende hatten, wider. Als Vertreter für diese computergestützten Prozessselektoren können exemplarisch [Lovatt und Shercliff, 1998a], [Lovatt und Shercliff, 1998b], [L'Eglise et al., 2001], [Brechet et al., 2001], [Shercliff und Lovatt, 2001], [LeBacq et al., 2002] und [Esawi und Ashby, 2004] genannt werden. Auch in neuerer Zeit werden noch verwandte Ansätze vorgestellt, wie z. B. in [Zindani et al., 2020].

In [Roth, 1996] werden sogenannte Konstruktionskataloge zur systematischen Konstruktion vorgestellt. Dahinter verbirgt sich eine umfangreich aufbereitete Sammlung von Prinziplösungen für unterschiedliche Konstruktionsbereiche wie z. B. der Verbindungstechnik, die dem/der Anwender/in nach dem Baukastenprinzip Basiselemente zur Verfügung stellt, auf deren Basis die Konstruktion ausgeführt werden kann. Des Weiteren werden auch zahlreiche methodische Hilfestellungen bei der Konstruktionsarbeit beschrieben.

Eine detaillierte Methodik zur Materialauswahl ist in [Ashby, 2005] dargestellt, die sich zusätzlich auch dem Zusammenspiel von Material, Herstellungs- und Fügeprozessen widmet. Wie bereits aus den Prozessselektoren bekannt, wird bei ASHBY prinzipiell die Menge der infrage kommenden Entitäten von allen Möglichkeiten ausgehend immer weiter eingeschränkt. Eine fundamentale Rolle dabei spielen aus Materialkennwerten (z. B. E-Modul, Spannung, Biege- oder Zugfestigkeit) abgeleitete Schaubilder – in der Literatur auch unter dem Stichwort „ASHBY-Chart" bekannt –, in denen ähnlich zu Technologiewechselkurven Materialien bezüglich dieser Kennwerte aufgetragen sind und sich Inseln der Verwendbarkeit ausbilden. Eine ähnliche Vorgehensweise findet sich bei [Swift und Booker, 1997] und vielen weiteren Autoren.

Basierend auf den in den vorgenannten Büchern geschaffenen Grundlagen bauen Arbeiten für spezielle Problemstellungen innerhalb des Themenkomplexes auf. In [Pasini, 2002] wird die gekoppelte Auswahl von Material und Form durch Transformationsgruppen beschrieben, die auf den Kennwerten von ASHBY aufbauen.

In [Beilstein, 2011] werden ähnlichkeitsmechanische Betrachtungen angestellt, um dimensionslose Kenngrößen zur Gewichtsabschätzung und Bewertung von Strukturverbindungen im Flugzeugentwurf abzuleiten.

Auf andere Art, aber thematisch verwandt, wird in [Mesa et al., 2018] die Strukturverbindung allgemein charakterisiert und für eine verbesserte Auswahl bewertet.

Ebenfalls auf dem Gebiet der Verbindungstechnik, aber beschränkt auf die Menge der lösbaren Verbindungen, wird in [Rusitschka, 2017] eine Repräsentationsform und Auswahlmethodik beschrieben.

In [Swain et al., 2014] und [Das und Swain, 2016] wird die Auslegung von Fügebereichen durch die Einführung sogenannter Liaisons formalisiert.

Weitere wissenschaftliche Abhandlungen auf dem Gebiet der Fügetechnologien sind in [J. Stahl, 2008], [Michalos et al., 2010], [Choudry, 2019] und [Omar und Soltan, 2020] beschrieben.

Eine letzte Rubrik in diesem Themenkomplex ist die integrierte Definition von Produkt und zugehörigem Produktionsprozess. In [Zafirov, 2017], [Stoffels, 2017] und [Kaspar und Vielhaber, 2018] sind entsprechende Ansätze dazu zu finden. Der Autor der vorliegenden Arbeit hat sich in Kooperation mit Forschungspartnern intensiv mit diesem Thema auseinandergesetzt. Für eine tiefergehende Darstellung sei daher an dieser Stelle auf [Schopper et al., 2021] verwiesen.

Tabelle 2.3: Bewertung der untersuchten Literatur im TK 2 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[Houldcroft, 1990]	○	◔	○	○	◑
[Roth, 1996]	○	◕	◔	◔	◔
[Lovatt und Shercliff, 1998a]	◑	◑	◔	◔	○
[Lovatt und Shercliff, 1998b]	◑	◑	◔	◔	○
[L'Eglise et al., 2001]	◑	◑	◔	◔	○
[Brechet et al., 2001]	◑	◑	◔	◔	○
[Shercliff und Lovatt, 2001]	◑	◑	◔	◔	○
[LeBacq et al., 2002]	◑	◑	◔	◔	○
[Esawi und Ashby, 2004]	◑	◑	◔	◔	◔
[Zindani et al., 2020]	◑	◑	○	◑	○
[Ashby, 2005]	◔	◔	◔	○	○
[Swift und Booker, 1997]	◔	◔	◔	○	○
[Pasini, 2002]	○	○	◔	◑	◑
[Beilstein, 2011]	◔	◑	◑	◑	◑
[Mesa et al., 2018]	◔	○	○	◑	◔
[Rusitschka, 2017]	◔	○	○	◑	◔
[Swain et al., 2014]	◑	◑	◔	◑	◕
[Das und Swain, 2016]	◑	◑	◔	◑	◕
[J. Stahl, 2008]	◔	○	○	◔	○
[Michalos et al., 2010]	○	○	○	◑	○
[Choudry, 2019]	◔	◔	○	◔	◔
[Omar und Soltan, 2020]	◔	◑	◑	◔	○
[Zafirov, 2017]	◕	◑	◔	◔	○
[Stoffels, 2017]	◑	◔	◔	◑	◔
[Kaspar und Vielhaber, 2018]	◑	◑	◑	◑	○
[Schopper et al., 2021]	◕	●	●	◑	○

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

TK 3 Wissensbasierte Konstruktionssysteme

Eine kurze Einführung in die wissensbasierten Konstruktionssysteme wurde bereits in Abschnitt 2.2.3 gegeben. Der Umfang an Literatur im Themenkomplex der wissensbasierten Konstruktionssysteme ist überwältigend und eine umfassende Übersicht zu geben kaum möglich. In diesem Abschnitt sollen daher nur diejenigen Beiträge genannt werden, die eine herausragende Stellung innerhalb des Wissensgebiets innehaben oder einen Einfluss auf die Entwicklung der im Rahmen der vorliegenden Arbeit vorgestellten Methodik haben.

In [Alfaris, 2009] wird ein Framework namens „Multi-Disciplinary Design System (MDDS)“ zur Unterstützung des Entwurfs komplexer Systeme in einer Vielzahl von Bereichen beschrieben. Dabei liegt ein Verständnis des Entwurfs analog dem mechatronischen V-Modell (siehe Abbildung 2.7) und der Entwurfsraumexploration (siehe Abschnitt 2.2.2) zugrunde, indem zunächst eine funktionale und geometrische Dekomposition erfolgt, auf deren Grundlage die Produktarchitektur aufgebaut wird. Anschließenden wird eine modularisierte Modellierung der Entwurfsaktivitäten, Synthese, Analyse und Bewertung ausgeführt. Diese Module werden schließlich in einem Datenflussnetzwerk integriert und zur Exploration des Entwurfsraums genutzt. Daraus entstehen ganzheitlich optimierte Lösungen. Aus Sicht des Autors der vorliegenden Arbeit handelt es sich um ein umfassendes Framework, das in vielen Bereichen als Inspiration zur Modellierung der komplexen Zusammenhänge des Ingenieurentwurfs dienen kann. Obwohl die Automatisierung einiger Teilprozesse diskutiert wird, fehlt eine konkrete Vorstellung der vollständigen Automatisierung des Gesamtprozesses.

In [Sedchaicharn, 2010] wird die Festlegung der Produktarchitektur mit einer kombinierten Vorgehensweise bestehend aus systematischer Beschreibung, matrixbasierten und algorithmischen Methoden beschrieben.

In [Helms, 2013] wird auf der Basis objektorientierter Graphgrammatiken ein Konstruktionsframework namens *booggie* vorgestellt, das die automatisierte Generierung von Modellen ermöglicht. Der Ansatz ist stark von graphenbasierten Entwurfssprachen beeinflusst. Als Anwendungsbeispiel wird die Sitzverteilung in einem Verkehrsflugzeug modelliert.

In [Kratzer, 2015] wird eine proaktives Konstruktionssystem auf Basis von Softwareagenten (ProKon) vorgestellt. Das ProKon-System ist ein Assistenzsystem für den/die Ingenieur/in und ermöglicht eine Konsistenzprüfung des Produktmodells gegenüber den Anforderungen, der Funktion und den Design for X-Richtlinien [Kratzer, 2015, S. VII]. Wie bereits beim MDDS steht auch bei ProKon eine vollständige Automatisierung nicht im Fokus des Entwicklers.

In [Gembarski, 2018] wird eine Vorgehensweise zum Komplexitätsmanagement mittels wissensbasiertem CAD vorgestellt. Im Fokus steht die Modellierung konstruktiver Lösungsräume, aus denen individuelle Varianten von Produkten abgeleitet werden können.

Die am Massachusetts Institute of Technology (MIT) entwickelte Systematik CommonKADS hat sich zu einer weitverbreiteten Anwendung im Bereich der wissensbasierten Konstruktionssysteme entwickelt. In [Schreiber et al., 1994], [Valente et al., 1998] und [Schreiber et al., 2001] werden die Konzepte beschrieben. Neben CommonKADS sind MIKE [Angele et al., 1992], DEKLARE [Forster et al., 1996], MOKA [Stokes, 2001] und KNOMAD [Curran et al., 2010] wichtige Vertreter wissensbasierter Konstruktionssysteme [Haibing Li, 2020, S. 32].

In [Sobieszczanski-Sobieski et al., 2015] findet sich eine umfassende Darstellung und Diskussion der multidisziplinären Optimierung mit KBE-Ansätzen als Buch.

Tabelle 2.4: Bewertung der untersuchten Literatur im TK 3 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[Alfaris, 2009]	◕	●	●	◑	◑
[Sedchaicharn, 2010]	◕	◑	◑	◑	◔
[Helms, 2013]	◕	◑	◕	◑	◔
[Kratzer, 2015]	◕	◔	◔	◔	○
[Gembarski, 2018]	◑	◑	◑	◔	○
[Schreiber et al., 1994]	◔	◔	◔	◔	○
[Valente et al., 1998]	◔	◔	◔	◔	○
[Schreiber et al., 2001]	◔	◔	◔	◔	○
[Angele et al., 1992]	◔	◔	◔	◔	○
[Forster et al., 1996]	◔	◔	◔	◔	○
[Stokes, 2001]	◔	◔	◔	◔	○
[Sobieszczanski-Sobieski et al., 2015]	◕	◑	◕	◑	○

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

TK 4 Entwurfsraum und Entwurfsraumexploration

Der Entwurfsraum an sich und die Möglichkeit der Entwurfsraumexploration wurde in Abschnitt 2.2.2 bereits eingeführt. Der Themenkomplex stellt einen elementar wichtigen Pfeiler der vorliegenden Arbeit dar und aus diesem Grund ist die Literaturübersicht besonders ausführlich.

Die ersten Grundlagen in Bezug auf die Entwurfsraumexploration wird in [Zwicky, 1971] mit der Einführung des morphologischen Kastens gelegt, bei dem Basiselemente permutiert und somit eine Methode zur umfassenden Variantenuntersuchung in allen kombinatorischen Möglichkeiten beschrieben wird.

In [Forster et al., 1997] wird eine auf DEKLARE basierende Methode für Variantendesign unter Erfüllung definierter Randbedingungen beschrieben.

Stellvertretend für die große Vielfalt der Beiträge auf dem Gebiet des parametrischen CAD sei [Shah, 2001] angeführt. Die Beschreibung geometrischer Formen über Parameter ermöglicht die Generierung einer Vielzahl an Varianten, die allerdings topologisch identisch sind.

In [Fitch und Cooper, 2005] werden probabilistische Methoden eingesetzt, um die weitere Ausgestaltung einer durch initiale Randbedingungen beschriebenen Konstruktion auf Basis von Schätzwerten zu ermöglichen.

In [Suh, 2005] wird eine auf Transformationen zwischen Funktionen und Parametern be-

ruhende Konstruktionssystematik beschrieben, die in der Lage ist, vollautomatisch ausgeführt zu werden. Das Wort axiomatisch bezieht sich auf die beiden Grundannahmen, das Axiom der Unabhängigkeit und das Axiom des Informationsgehalts. Nach SUH ist ein Design auf ein Minimum an Funktionen und Informationsgehalt zu optimieren.

In [Kingston, 2007] wird eine auf CommonKADS basierende Methodik für Knowledge Engineering und die Nutzung von Ontologien vorgestellt.

In [Yang et al., 2008] wird ein auf Ontologien basierendes Produktkonfigurationstool vorgestellt. Das Wissen wird über die OWL repräsentiert und aufgezeigt, wie weitere Constraints berücksichtigt werden können.

In [Aldanondo und Vareilles, 2008] wird für den Anwendungsfall der Mass Customization dargelegt, wie Anforderungen und Prozesse gleichzeitig bei der Produktkonfiguration mitberücksichtigt werden können.

In [Vajna et al., 2010] wird eine Lösungsraumexplorationsstrategie mit der Autogenetic Design Theory (ADT) beschrieben. Dabei werden Anforderungen, Constraints und Randbedingungen berücksichtigt, um nach einem natürlich Evolutionsprinzip die beste Alternative auszuwählen. Eine besondere Neuerung stellt dabei die Einführung sogenannter Tabuzonen dar, die den Lösungsraum einschränken sollen.

In [Kang et al., 2011] wird eine Vorgehensweise zur kosteneffektiven Suche nach optimalen Lösungen im Entwurfsraum vorgestellt. Der Grundgedanke ist dabei, dass es im Entwurfsraum viele äquivalente Entwürfe gibt, die nicht alle einzeln untersucht werden müssen. Der präsentierte Ansatz soll sicherstellen, dass nur nicht äquivalente Varianten untersucht werden.

In [Saxena und Karsai, 2010] wird der Fokus auf eine Verallgemeinerung der Entwurfsraumexploration gelegt, da diese in anderen Ansätzen oft auf problem- oder domänenspezifischen Annahmen basiert. Dies wird im Wesentlichen durch die Entkopplung des Entwurfsraums vom Suchalgorithmus erreicht.

In [Lo et al., 2010] wird aufgezeigt, wie ein Variantendesign mittels morphologischem Kasten realisiert werden kann.

In [Krish, 2011] wird eine CAD-basierte, generative Entwurfsmethodik vorgestellt, die eine Entwurfsraumexploration ermöglicht.

In [Kesper, 2012] wird ein Variantenentwurf mit matrixbasierten Methoden beschrieben. Dabei werden Eigenschaften- und Komponentenkombinationen von Produktvarianten ein eine Matrixnotation überführt und mittels einer Software namens LOOMEO anschließend analysiert und grafisch aufbereitet. Dadurch können verschieden Aspekte der Variantenvielfalt einfacher und effizienter untersucht und in ihren Auswirkungen abgeschätzt werden.

In [J. Schmidt und Rudolph, 2014] werden graphenbasierte Entwurfssprachen am Beispiel von Satellitenantriebssystemen eingesetzt, um eine automatisierte Entwurfsraumexploration zu ermöglichen. Die Untersuchung topologisch unterschiedlicher Varianten ermöglicht das Auftragen sogenannter Technologiewechselkurven, die für ein Menge ausgewählter Parameter als Koordinatenachsen – in diesem Fall der Systemmasse über der Geschwindigkeitsdifferenz Δv – eine optimale Systemauslegung indizieren.

In [R. Wang et al., 2018] wird eine systematische Exploration des Entwurfsraums mithilfe

einer auf Vorlagen basierenden ontologischen Methode vorgestellt. Insbesondere die Verknüpfung von Ontologien mit der Entwurfsraumexploration ist bei diesem Beitrag hervorzuheben. In diesem Zusammenhang kann auch [Das und Swain, 2020] genannt werden, wo die Autoren ebenfalls ein auf Ontologien basierenden Ansatz zur Entwurfsraumexploration beschreiben.

In [McMahon, 2021] werden philosophische Überlegungen zu Mustern im Entwurf und zur Entwurfsexploration angestellt.

Weitere Literatur findet sich in [Hoschek, 1994], [Lenders, 2009], [Zimmermann und von Hoessle, 2013], [Lüdtke, 2016], [Gembarski et al., 2017], [Müller, 2018] und [Müller, 2020].

Tabelle 2.5: Bewertung der untersuchten Literatur im TK 4 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[Zwicky, 1971]	○	○	◔	○	○
[Forster et al., 1997]	◔	◔	◔	○	○
[Shah, 2001]	◔	◔	○	○	◑
[Fitch und Cooper, 2005]	◔	◔	◑	◕	○
[Suh, 2005]	◑	◑	●	◕	◔
[Kingston, 2007]	◔	◔	◑	◔	○
[Yang et al., 2008]	◑	◔	○	◑	○
[Aldanondo und Vareilles, 2008]	◑	◔	◑	◕	○
[Vajna et al., 2010]	◑	◑	◕	◕	○
[Kang et al., 2011]	◔	◔	◑	◑	○
[Saxena und Karsai, 2010]	◑	◔	◔	◕	○
[Lo et al., 2010]	◔	○	◔	◔	○
[Krish, 2011]	◔	◑	◔	◑	◑
[Kesper, 2012]	◔	◔	◑	◕	○
[J. Schmidt und Rudolph, 2014]	◕	◑	●	◑	◑
[R. Wang et al., 2018]	◑	◔	◑	◑	○
[Das und Swain, 2020]	◑	◔	◑	◑	○
[McMahon, 2021]	◔	◔	◑	○	○
[Hoschek, 1994]	○	○	◑	◑	◔
[Lenders, 2009]	◔	◔	◔	◑	○
[Zimmermann und von Hoessle, 2013]	◕	◑	◕	◑	○
[Lüdtke, 2016]	◔	◔	◔	◔	○
[Gembarski et al., 2017]	◔	◔	◑	◔	◑
[Müller, 2018]	◔	◑	◑	◔	○
[Müller, 2020]	◑	◑	◕	◑	◔

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

TK 5 Forward-, Reverse- und Re-Engineering in der Informatik und im Ingenieurentwurf

In diesem Themenkomplex soll die Literatur aus Informatik und Ingenieurwissenschaft gebündelt dargestellt werden, da die Übergänge der beiden fließend sind.

Wie bereits in Abschnitt 2.1.4 aufgezeigt, geht die Grundidee des Round-Trip-Engineering in der Softwareentwicklung zurück auf [Chikofsky und Cross, 1990].

In [Knublauch und Rose, 2000] wird ein auf UML-Ontologien und Java basierender Ansatz zum automatisierten Round-Trip-Engineering in der Softwareentwicklung vorgestellt.

In [Aßmann, 2003] wird eine Methodik für das automatisierte Round-Trip-Engineering im Softwareentwurf vorgestellt.

In [Welch und Dixon, 1990] wird ein Ansatz zum iterativen Re-Engineering am Beispiel des Entwurfs von Blechhalterungen aufgezeigt.

In [Arana et al., 2001] wird auf Basis von DEKLARE eine Methodik zur Auslegung von Varianten im Rahmen eines Re-Engineerings aufgezeigt.

In [Baxter et al., 2008] wird ein Framework zur integrativen Wiederverwendung von Entwurfswissen in Verbindung mit dem Anforderungsmanagement präsentiert und anhand des Beispiels einer Vakuumpumpe demonstriert.

In [Mesa et al., 2015] wird ein Ansatz für eine modulare Produktarchitektur unter besonderer Berücksichtigung der Rekonfigurationsfähigkeit bei wechselnden Anforderungen demonstriert.

In [Camba et al., 2016] wird anhand eines parametrischen CAD-Modells die prinzipielle Wiederverwendungsmöglichkeiten von Domänenmodellen vorgestellt.

In [Anwer und Mathieu, 2016] wird ein Reverse-Engineering-Ansatz im Bereich der Geometriedomäne vorgestellt, mit dessen Hilfe aus Domänenartefakten ein geometrisches Modell rekonstruiert werden kann.

In [Gori et al., 2017] wird mit *FlowRep* eine wiederholt in der Geometriedomäne angesiedelte Methodik vorgestellt, die die Ableitung von Kurvennetzwerken aus Freiformflächen ermöglicht. Überdies haben die Autoren eine Implementierung des entwickelten Algorithmus veröffentlicht.

In [Du et al., 2018] wird unter dem Namen *InverseCSG* ein Ansatz zur automatischen Konvertierung von 3D-Modellen zu Constructive Solid Geometry (CSG)-Bäumen vorgestellt. Eine solche Vorgehensweise wird insbesondere beim Reverse-Engineering von Geometrieartefakten im Rahmen eines automatisierten Entwurfs benötigt.

In [Anacker et al., 2020] stellen die Autoren einen Ansatz zur Formulierung von Lösungsmustern vor, die zur effizienten Nutzung von vorhandenem Entwurfswissen führen sollen.

Tabelle 2.6: Bewertung der untersuchten Literatur im TK 5 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[Chikofsky und Cross, 1990]	○	○	○	●	○
[Knublauch und Rose, 2000]	◔	◔	◔	◕	○
[Aßmann, 2003]	○	○	○	●	○

[Welch und Dixon, 1990]	◑	◔	◑	◔	◔
[Arana et al., 2001]	◑	◑	◑	◔	○
[Baxter et al., 2008]	◔	◔	◑	◑	◔
[Mesa et al., 2015]	◑	◕	◔	●	○
[Camba et al., 2016]	◔	◑	◔	◔	◕
[Anwer und Mathieu, 2016]	◔	○	○	◑	◕
[Gori et al., 2017]	○	○	○	◑	◕
[Du et al., 2018]	○	○	○	◔	◕
[Anacker et al., 2020]	◑	◑	◑	◔	○

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

TK 6 Generativer Entwurf und Entwurfsautomatisierung

In [L. C. Schmidt und Cagan, 1997] wird eine auf Graphgrammatiken basierende Systematik namens GGREADA beschrieben, mit der ein automatisierter Entwurf durchgeführt werden kann. Die Autoren verwenden das Demonstrationsbeispiel, den Bausatz eines Modellfahrzeugs.

In [Soddu, 2006] findet sich eine allgemeine Beschreibung des generativen Entwurfs und verwandter Themenfelder sowie einer Darstellung der Grenzen der Vorgehensweise.

In [Kurtoglu et al., 2010] wird eine funktionsbasierte Entwurfsautomatisierung mittels Graphenalgorithmen gezeigt.

In [Tarkian, 2012] wird eine Vorgehensweise zur Entwurfsautomatisierung auf Basis parametrischer Geometriebasiskörper beschrieben.

In [Groß, 2014] wird auf Basis graphenbasierter Entwurfssprachen die automatisierte Auslegung eines Satelliten mitsamt aller Subsysteme vorgestellt. Dabei werden die Bedeutung der Rekombinationsprinzipien Bottom-Up und Top-Down im automatisierten Entwurfsprozess beleuchtet und der Entwurf als solcher als die Abfolge eines Beschreibungsproblems, Enumerationsproblems, Konfigurationsproblems und Integrationsproblems gedeutet.

In [Hooshmand, 2014] wird die Kombination von generativen Graphgrammatiken mit Simulationen vorgeschlagen, um ingenieurwissenschaftliche Entwurfsprobleme automatisiert zu lösen. Als Anwendungsbeispiele werden ein Strömungskanal und ein Layoutproblem besprochen.

In [Münzer, 2015] wird eine auf einem graphenbasierten Ansatz aufbauende, objektorientierte Methodik für die automatisierte Entwurfssynthese präsentiert. Ein Algorithmus löst das Constraint-Satisfaction-Problem, das sich durch das Zusammenspiel unterschiedlicher Randbedingungen ergibt. Eine automatisierte Untersuchung des Entwurfsraums wird ermöglicht. Die Anwendung wird anhand diverser Beispiele validiert.

In [Sauthoff, 2017] wird eine detaillierter Vorgehensweise zur Entwurfsautomatisierung von Strukturkomponenten aufgezeigt. Der Autor definiert eine sogenannte technische Vererbung, um geänderte Randbedingungen automatisiert berücksichtigen zu können.

In [Poorkiany, 2017] werden Methoden und Werkzeuge für die Unterstützung der Modellie-

rung von Entwurfswissen und die Entwurfsautomatisierung vorgestellt.

In [Gericke und Eisenbart, 2017] stellen die Autoren eine Vorgehensweise zur funktionsgetriebenen Konstruktion vor.

In [Rigger und Vosgien, 2018] fassen die Autoren den Stand der Forschung und Entwicklung im Bereich der Entwurfsautomatisierung (2018) zusammen.

In [Johansson, 2019] wird eine KBE-Anwendung zur automatisierten Auslegung von Haltern im Automobilbereich gezeigt.

In [T. Li et al., 2020] präsentieren die Autoren verschiedene Anforderungs-, funktionale, logische und physikalische Modelle für den Entwurf im Flugzeugbereich.

In [Haibing Li, 2020] wird ein generativer Ansatz für die Auslegung von Strukturkomponenten unter Unsicherheiten vorgestellt. Der Autor demonstriert anhand eines Rohbaukarosserieteils die automatische Ableitung der benötigten Modellartefakte mit dem sogenannten Generative Design Approach (GDA).

In [Albers et al., 2021] wird die Einführung eines Variationsoperators im Rahmen des Product Generation Engineering (PGE) nach ALBERS für den generativen Entwurf diskutiert.

Tabelle 2.7: Bewertung der untersuchten Literatur im TK 6 anhand der fünf Forschungsfragen aus Tabelle 1.1.

Referenzliteratur	**FF 1**	**FF 2**	**FF 3**	**FF 4**	**FF 5**
[L. C. Schmidt und Cagan, 1997]	◔	◑	◕	◕	◔
[Soddu, 2006]	◑	◑	◔	◔	○
[Kurtoglu et al., 2010]	◔	◔	◑	◑	○
[Tarkian, 2012]	◑	◔	◑	◔	◑
[Groß, 2014]	◕	◑	●	◑	◑
[Hooshmand, 2014]	◑	◑	◑	◑	◔
[Münzer, 2015]	◔	◑	◑	●	◔
[Sauthoff, 2017]	◕	◕	●	◕	◕
[Poorkiany, 2017]	◕	◑	◑	◑	◑
[Gericke und Eisenbart, 2017]	◔	◑	◑	◑	○
[Rigger und Vosgien, 2018]	◑	◑	◑	◔	◔
[Johansson, 2019]	◔	◑	◑	◔	◑
[T. Li et al., 2020]	◕	◑	◕	◑	◔
[Haibing Li, 2020]	◕	◕	◑	◑	◕
[Albers et al., 2021]	◕	◑	◕	◔	◔

Die Forschungsfrage wird
○ nicht ◔ wenig ◑ teilweise ◕ umfangreich ● vollständig adressiert.

3 Round-Trip-Engineering im modellbasierten Ingenieurentwurf

> *Wohin noch mag mein Weg mich führen? Närrisch ist er, dieser Weg, er geht in Schleifen, er geht vielleicht im Kreise. Mag er gehen, wie er will, ich will ihn gehen.“*
>
> — **Hermann Hesse (aus dem Buch Siddhartha [Hesse, 1975, S. 80])**
> (deutsch-schweizerischer Schriftsteller, Dichter und Maler)

In diesem Kapitel wird im ersten Abschnitt die wissenschaftliche Grundlage und die philosophische Begründung für die Übertragung des ursprünglich aus der Softwareentwicklung stammenden Konzepts des Round-Trip-Engineerings auf das Ingenieurwesen dargestellt. Aufgrund der fortschreitenden Digitalisierung im Ingenieurwesen nähern sich Informatik und Ingenieurwissenschaft in vielerlei Hinsicht immer weiter an. Es existieren aber auch fundamentale Unterschiede, die im weiteren Verlauf dieses Kapitels herausgearbeitet werden. Insbesondere das Teilgebiet des Reverse-Engineering stellt in der Ingenieurwissenschaft eine große Herausforderung dar. Dies liegt in der Tatsache begründet, dass hierbei ein Mustererkennungsproblem vor dem Hintergrund eines vielfältigen Lösungsraums gelöst werden muss.

Im zweiten Teil dieses Kapitels wird aufbauend auf der gelegten theoretischen Grundlage ein allgemeines Entwurfsrahmenwerk beschrieben, das ein solches Round-Trip-Engineering im modellbasierten Ingenieurentwurf ermöglicht und unabhängig von einer Implementierung aufgezeigt, wie die relevanten Domänen modelliert und verknüpft werden können.

Kapitelübersicht

3.1 Die Philosophie des Round-Trip-Engineering

In diesem Abschnitt wird eine Definition des im Ingenieurwesen neuartigen Begriffs Round-Trip-Engineering (RTE) vorgestellt und aufgezeigt, welche neuen Einsichten in den Entwurfsprozess einerseits und Möglichkeiten für ein zukünftiges Engineering andererseits damit einhergehen.

Im Zentrum der bisherigen Forschung und Entwicklung im Bereich des modellbasierten Ingenieurentwurfs (vgl. Abschnitt 2.2.3) stand und steht nach wie vor in erster Linie die Unterstützung des kreativen Ingenieurs und dessen Entlastung von Routinetätigkeiten wie der Erstellung von Simulationsmodellen. Ebenfalls werden viele Konstruktionssystematiken maßgeblich vor dem Hintergrund eines konstruierenden Menschen abgeleitet (siehe hierzu z. B. [Swift und Booker, 1997], [Ullman, 2003], [Ashby, 2005], [Ehrlenspiel und Meerkamm, 2013], [Pahl und Beitz, 2013] u. v. m.). Die Vorgehens- und Denkweise des Menschen beim Konstruieren beeinflusst damit indirekt auch seine angewandten Methoden und Werkzeuge.

Diese Konzentration auf den menschlichen Faktor bringt aber gerade in Hinsicht auf die Entwurfsautomatisierung und Lösungsraumuntersuchung viele Defizite mit sich. Der größte Nachteil ist in jedem Fall, dass der Mensch nicht mehr in der Lage ist, die beim heutigen Entwurf komplexer Systeme wie z. B. Flugzeugen, Schiffen oder Automobilen anfallenden enormen Datenmengen und sonstigen Entwurfsartefakte vollständig zu überblicken und adäquat zu evaluieren[1]. Das Wissen, die Komplexität und auch Anzahl an unterschiedlichen Modellen wird in Zukunft stetig weiter zunehmen [Fritzsche und Oks, 2018]. Diese nur möglichst gut für den Menschen aufzubereiten, ist somit keine nachhaltige und zufriedenstellende Lösung.

Auch die Automatisierung einzelner Phasen des Entwurfs oder die Generierung einzelner Domänenmodelle, welche in vielen aktuellen Arbeiten verfolgt werden (siehe Abschnitt 2.3), kann die prinzipiellen Nachteile nur abmildern. Nach Meinung des Autors der vorliegenden Arbeit ist die einzige Möglichkeit, den Herausforderungen des zukünftigen Ingenieurentwurfs angemessen begegnen zu können, die lückenlos durchgängige Automatisierung des Entwurfsprozesses unter weitestgehender Exklusion des menschlichen Eingriffs aus der Entwurfsschleife. Dabei handelt es sich um einen zwangsläufigen Prozess, der auch in anderen Bereichen zu beobachten ist. Beispielsweise war der Pilot eines Flugzeuges in den 1960er Jahren noch für alle Steuerkommandos selbst verantwortlich. Ein modernes Überschallflugzeug kann heutzutage durch einen Piloten allein, ohne Sensorik- und Computerunterstützung, gar nicht mehr beherrscht werden. In Zukunft wird es mit hoher Wahrscheinlichkeit sicherer und auch effizienter sein, den Piloten ganz durch einen Autopiloten zu ersetzen. Beim Schach, dessen Schwierigkeit aus der riesigen Anzahl potenzieller Spielstellungen – in etwa 10^{43} [Bonsdorf et al., 1978, S. 9] – resultiert, besiegen Computer, auf denen spezielle Algorithmen laufen, schon seit Ende des letzten Jahrtausends mühelos jeden Großmeister im Schach. Auch beim Brettspiel Go mit einer unvorstellbar großen Spielstellungsvarianz von circa $2 \cdot 10^{170}$ [Tromp, 2016] ist die Maschine mittlerweile nicht mehr von einem menschlichen Spieler besiegbar. In diesem Aspekt wird auch der Unterschied des in der vorliegenden Arbeit vorgestellten RTE im Vergleich zu einem wis-

[1] Komplexe Systeme werden niemals von einer einzigen Person entworfen, trotzdem gelten die Aussagen völlig analog auch für eine Gruppe entwerfender Menschen, da hier die Zunahme der Verarbeitungskapazität nur über eine Zunahme der Interface- und Abstimmungsaufwände erreicht werden kann.

sensbasierten Expertensystem offenbar, da letzteres als Kernbestandteile eine Erklärungs- und Dialogkomponente für die Interaktion mit dem Menschen beinhaltet (siehe Abschnitt 2.2.3).

Als Konsequenz dieser Beobachtung ergibt sich, dass sämtliche Aufgabenstellungen des Entwerfens – Neukonstruktion, Variantenkonstruktion und Anpassungskonstruktion (siehe Abschnitt 2.2.1) – und die zugehörigen Modelle vollautomatisch innerhalb des Frameworks generiert werden müssen. Gleichzeitig muss weiterhin sichergestellt sein, dass der/die Ingenieur/-in die volle Kontrolle über den Entwurf an sich behält. Das Haupttätigkeitsfeld wird sich daher zukünftig auf das präzise Formulieren der Anforderungen an das zu entwerfende Produkt konzentrieren. Die VDI-Norm 5610 spricht diesbezüglich auch von einem/einer „Wissensingenieur/-in“ [VDI 5610-2, S. 15]. Den eigentlichen Vorgang des Entwerfens übernimmt das RTE-Entwurfsrahmenwerk, das automatisiert eine Auswahl an gut an die spezifizierten Randbedingungen angepassten Produkten entwirft[2]. Aus dieser Auswahl ist es dann die Aufgabe des Ingenieurs, anhand bestimmter Kennwerte eine Variante auszuwählen.

Wichtig bei diesem Ansatz ist zu verstehen, dass der/die Ingenieur/-in nur zu Beginn und ganz am Ende in den Prozess involviert ist. Durch die Definition der Randbedingungen an den Entwurf kann einerseits indirekt Einfluss auf das Produkt genommen werden und andererseits auf geänderte Anforderungen reagiert werden. Die Festlegung auf eine Variante geschieht erst, wenn wirklich alle Informationen aus allen beteiligten Domänen in einem Produkt zusammengeführt wurden. Das ist der entscheidende Unterschied zu den etablierten Vorgehensmodellen, in denen die Entscheidung während des Entwurfsvorgangs und somit auf einer unvollständigen Daten-, Informations- und Modellgrundlage getroffen werden muss [Ullman, 2003].

Das Konzept des in dieser Arbeit vorgestellten RTE zielt genau auf den beschriebenen voll maschinellen Entwurf ohne menschlichen Eingriff ab. Da aber Randbedingungen niemals in ausreichender Präzision und Vollständigkeit definiert werden können, um daraus ein eineindeutiges Produkt entwerfen zu können, kann der automatisierte Entwurf niemals geradlinig – im Sinne eines First-best-Entwurfs – vonstattengehen. Auch der Umgang mit Unsicherheiten in den Anforderungen ist ein wichtiges Thema in diesem Zusammenhang [Haibing Li, 2020] da vielen Entwurfsvariablen a priori kein eindeutiger Wert zugewiesen werden kann. Wie auch in den klassischen Entwurfsmethodiken muss der Algorithmus sich in Schleifen nach vorne arbeiten, die abgeleiteten Modelle immer wieder neu validieren und gegebenenfalls zum Ausgangspunkt zurückkehren. Diese Aussagen gelten somit analog zur menschlichen Vorgehensweise, wie sie beispielsweise in [J. Wang, 2002] beschrieben wird. Bei diesem Vor- und Zurückgehen treten immer wieder ähnliche Muster auf, die algorithmisch abgebildet werden können.

Ein besonders wichtiger Aspekt dabei ist, dass die Domänenmodelle immer valide gegenüber den Anforderungen und somit konsistent zueinander gehalten werden müssen. Auf dem Gebiet der Softwaretechnik haben sich in einem ähnlichen Problemkontext dafür Techniken etabliert, die unter dem Begriff Round-Trip-Engineering (RTE) zusammengefasst werden (siehe hierzu Abschnitt 2.1.4). Aus diesem Grund soll im Rahmen der vorliegenden Arbeit der Begriff RTE und die zugehörigen Modellierungs-, Generierungs- und Analysetechniken auf

[2] Auch bei vielen Optimierungsproblemen steckt die Schwierigkeit heutzutage in der Problemspezifikation und Lösungsraumdefinition („closed world assumption“ aus der KI) und das eigentliche Rechnen bzw. Lösungsfinden übernimmt der Computer bzw. Algorithmus

das Ingenieurwesen übertragen, angepasst und erweitert werden. Für die Aufrechterhaltung der globalen Systemkonsistenz wird die Automatisierung dreier Entwurfsvorgänge benötigt, die gleichermaßen in der modellbasierten Softwareentwicklung identifiziert worden sind: Das *Forward-*, *Reverse-* und *Re*-Engineering. Während das Forward-Engineering bei synthetisierenden Vorgängen während der Konkretisierung des Entwurfs auftritt, wird beim Reverse-Engineering von einer fertigen Entwurfslösung ausgegangen und versucht über eine Abstraktion zurück zu einer generischen Beschreibung bzw. einem Systemmodell zu gelangen. Auf dieser Basis kann eine Überarbeitung[3] – das Re-Engineering – initiiert werden. Die einzelnen Teilprozesse werden in den folgenden Abschnitten detailliert beschrieben und in für die vorliegende Arbeit gültige Definitionen verdichtet. Zuletzt wird eine Definition für das gesamte RTE abgeleitet.

3.1.1 Forward-Engineering

Beim Forward-Engineering steht die Generierung einer Initiallösung aus abstrakten Anforderungen im Vordergrund. Alle dem Autor bekannten bisherigen Anwendungen im Bereich des Forward-Engineering haben gemeinsam, dass die Reihenfolge, in welcher die verschiedenen Domänenmodelle generiert werden, fest definiert ist. In einer starren Reihenfolge wird aber implizit Einfluss auf die Lösung genommen, da nachgelagerte Entscheidungen auf vorher getroffenen Entwurfsentscheidungen aufbauen. Die Entwurfsreihenfolge hängt im Wesentlichen von den initial an den Entwurf gestellten Randbedingungen ab, die ihrerseits in voller Allgemeinheit variabel sind. Für eine Teilmenge an Aufgabenstellungen kann davon ausgegangen werden, dass die initialen Randbedingungen konstant sind und daher eine konstante Entwurfsreihenfolge gerechtfertigt ist. Ein vollautomatisiertes RTE-Framework, wie es in dieser Arbeit vorgestellt wird, darf aber nicht auf einer im Vorhinein festgelegte Entwurfsabfolge aufbauen.

Wie bereits in [Vogel und Rudolph, 2015] angedeutet (siehe auch Abschnitt 2.2.1), wird der Ingenieurentwurf allgemein im weiteren Verlauf als Analogon zu einem nicht linearen System in der Mathematik aufgefasst, dessen Lösungssequenz im Allgemeinen von den Randbedingungen abhängig ist. Wie auch in der Mathematik sind Randbedingungen für eine konkrete Aufgabenstellung spezielle Variablen, denen ein konstanter Wert zugewiesen wird. Unter einer anderen Aufgabenstellung kann diese konstante Zuweisung aufgehoben und dadurch aus der Randbedingung wieder eine freie Variable werden[4]. Auf den ingenieurwissenschaftlichen Entwurf übertragen, kann man sich das anhand eines einfachen Beispiels veranschaulichen. Wenn das Material, aus dem ein Produkt oder Teilprodukt gefertigt werden soll, von vornherein feststehen soll – beispielsweise da kein alternatives Material zur Verfügung steht –, dann wäre das Material eine Randbedingung an den Entwurf. Wenn für das Material aber unterschiedliche Kategorien zugelassen sein sollen, dann wäre das Material als freie Variable aufzufassen, die gegebenenfalls auf einen Wertebereich eingeschränkt werden könnte. Die Analogie des

[3] Hierbei kann es sich um eine Anpassung an geänderte Randbedingungen, eine Erweiterung des Funktionsumfangs oder auch eine neuartige Implementierung handeln.

[4] Das Wechselspiel von fester Vorgabe zu freier Variable und wieder zurück ist typisch für häufige Zwischenanalysen im Entwurf und wird auch als „Was-wäre-wenn-Frage“ bezeichnet. Für deklarative Informationsverarbeitungsmechanismen wie den LPG (siehe Abschnitt 2.2.4) oder den im Rahmen dieser Arbeit vorgestellten Mechanismen stellt dieser typische Wechsel von Randbedingungen kein Verarbeitungsproblem dar.

Wertebereichs im Kontext der Materialdefinition wäre die Festlegung auf eine oder mehrere Materialkategorie(n), wie beispielsweise nur metallische oder keramische Werkstoffe.

Das Material als Entwurfsvariable hat seinerseits diverse Rückkopplungen auf weitere Entwurfsdomänen. Die anwendbaren Fertigungsverfahren können hier als naheliegendes Beispiel angeführt werden. Beispielsweise lassen sich metallische Werkstoffe in der Regel schweißen, keramische Werkstoffe im Allgemeinen nicht. Potenziell kann eine einzige Randbedingung sogar Einfluss auf alle während des Entwurfs überstrichenen Domänen haben. Es ergibt sich somit eine spezielle Lösungssequenz, die auf eine Teilmenge des möglichen Entwurfsraums führt.

Die Kopplung von Randbedingung auf freie Domänenvariable im Syntheseprozess des Entwurfs ist somit anhand dieses einfachen Beispiels motiviert. Im Entwurf gibt es eine Vielzahl solcher Kopplungen, die ein Mensch unmöglich alle in voller Tiefe und Konsequenz überblicken und abschätzen kann. Wichtig ist an dieser Stelle festzuhalten, dass jeder Versuch des Entwerfens nach einem festen Schema „*x follows y*“[5] usw. nur unter einem festen Satz an Randbedingungen zu rechtfertigen ist. Die vorigen Überlegungen führen zu folgender, im weiteren Verlauf der vorliegenden Arbeit gültigen Definition für das Forward-Engineering:

Definition 3.1.1: **Forward-Engineering in der Ingenieurwissenschaft**

Die Fähigkeit im Entwurf komplexer Systeme automatisiert auf Basis einer domänenübergreifenden Wissensbasis („Weltmodell“) aus Anforderungen konkrete Modelle auf allen Abstraktionsebenen synthetisieren zu können. Die zugehörige Modelltransformationssequenz wird individuell und automatisiert aus dem Problemkontext abgeleitet.

3.1.2 Reverse-Engineering

Das Reverse-Engineering widmet sich dem umgekehrten Weg des Forward-Engineering – von einer fertigen Konstruktion zurück zum Modell. Die zentrale Forschungsfrage dabei ist, wie der Prozess der vorwärts gerichteten Entwurfssynthese invertiert werden kann. Konkret bedeutet dies die Rückentwicklung der Auskonstruktion in die abstrakte Beschreibung und damit auch eine Umkehrung des klassischen PAHL/BEITZ-Schemas (vergleiche Abbildung 2.8).

Wie in [Schopper und Rudolph, 2018] gezeigt, handelt es sich bei diesem Vorgang ganz allgemein um die Lösung eines *Mustererkennungsproblems*. Hierbei wird eine fertige Konstruktion analysiert, abstrahiert und in einen breiteren Zusammenhang gebracht, in dem anschließend wieder alle mit den Randbedingungen kompatiblen Prinziplösungen erlaubt sind. Theoretisch ist es möglich, über das Lösungsprinzip hinweg auf die abstrakten Produktfunktionen und sogar auf die Anforderungen zurückzuschließen. Allerdings bleibt dieser Vorgang selbst mit einem sehr effektiven Mustererkennungsmechanismus von Ebene zu Ebene uneindeutig, da der vorausgegangene Syntheseprozess eine nicht-bijektive Abbildung von der höheren auf die niedrigere Ebene darstellt, deren Umkehrung mehrdeutig ist [Schopper und Rudolph, 2018].

[5] Motiviert von dem berühmten Zitat „*Form follows function*“ von Louis Sullivan [Sullivan, 1896, S. 408], der das Satzfragment zwar nicht erfunden, aber entscheidend geprägt hat.

In der praktischen Umsetzung reicht das Niveau der Lösungsprinzipien meist aus, um eine Variantenbildung – im weiteren Verlauf Re-Engineering bezeichnet – zu initiieren.

Dieses Vorgehen ist nur auf der Grundlage einer abstrakt vorgehaltenen Sammlung generischer Prinziplösungen denkbar, die im Rahmen des RTE auch für das Forward-Engineering benötigt wird und idealerweise identisch mit der vorgenannten ist. Die Inspiration hierfür liefern die in Abschnitt 2.1.1 vorgestellten, aus der objektorientierten Softwareentwicklung bekannten Entwurfsmuster, die an dieser Stelle in Form *digitaler Prinziplösungsmuster* auf das Ingenieurwesen übertragen werden. Im folgenden Abschnitt wird die Bedeutung der Mustererkennung für das Reverse-Engineering herausgearbeitet. Die folgenden beiden Abschnitte basieren auf in [Schopper und Rudolph, 2018] bereits veröffentlichten Ergebnissen.

Warum benötigt das Reverse-Engineering eine Mustererkennung?

Jede technische Aufgabe beginnt mit einer Modellbildung. In Abschnitt 2.2.1 wurde bereits dargelegt, dass im Ingenieurentwurf viele unterschiedliche Modelle benötigt werden. Als gängige Beispiele können hier ein Geometrie (CAD)-, ein Finite Elemente Methode (FEM)- oder ein numerisches Strömungsmechanik (CFD)-Modell genannt werden, da diese branchenübergreifend relevant sind. Als zusätzlicher Komplexitätsfaktor ist bei der Modellbetrachtung zu berücksichtigen, dass die diversen Modelle in unterschiedlichen offenen oder proprietären Computerprogrammen repräsentiert und gespeichert werden. In der Regel sind diese Modelle keine alleinstehenden, sondern gegenseitig gekoppelte Modelle, wodurch sich in der industriellen Praxis eine Reihenfolge in der Modellerstellung etablieren lässt. Der Entwurf als Ganzes kann somit als eine Sequenz verschiedener Modelle, die durch Modelltransformationen ineinander überführt werden, angesehen werden (vgl. Abbildung 2.13). Beispielsweise wird konkret oftmals zuerst ein CAD-Modell erstellt, bevor darauf aufbauend ein FEM- oder CFD-Modell abgeleitet wird, das wiederum für unterschiedliche Simulationen genutzt werden kann.

Bei der allermeisten dieser Modelltransformationen handelt es sich mathematisch gesehen um nicht-bijektive Abbildungen – sogenannte Projektionen – bei denen einige Informationen des ursprünglichen Modells während der Abbildung verloren gehen. In der Mathematik ist die Umkehrung einer Projektion nicht eindeutig und führt – falls überhaupt berechnenbar – auf eine mehrdeutige Lösung bzw. Lösungsschar. Dies lässt sich bereits am sehr einfachen Beispiel der Differentiation und anschließenden Integration einer ganzrationalen Funktion f nachvollziehen. Die Ableitung f' ist eindeutig bestimmbar, während die erneute Integration $\int f'$ das Auftreten einer Integrationskonstanten c mit sich bringt. Diese kann nachfolgend nur durch eine bekannte (oder geforderte) Randbedingung bestimmt werden.

Wie in Abschnitt 2.1.2 bereits dargestellt, treten in der modellbasierten Softwareentwicklung wesentlich ähnliche Modelltransformationen auf, die im Rahmen der Model-driven Architecture (MDA) als Modell-zu-Modell (M2M)- und Modell-zu-Text (M2T)-Transformation bezeichnet werden (siehe Abbildung 2.2). Für die Invertierung einer M2T-Transformation[7], muss eine zusätzliche und in der Definition der MDA fehlende Definition einer *Text-zu-Modell*

[7]Die Notwendigkeit der T2M wird an dieser Stelle zunächst ohne weitere Angabe von Gründen postuliert. Im weiteren Verlauf dieses Kapitels wird die Erforderlichkeit aber hinreichend motiviert und abgeleitet.

(T2M)-Transformation eingeführt werden. Auch diese Transformation ist analog zum bereits erwähnten mathematischen Invertierungsproblem mehrdeutig und kann nur durch den Einsatz entsprechender Randbedingungen wieder auf eine einzige Lösung eingeschränkt werden. Das Anwenden der korrekten Randbedingungen muss aus dem lokalen Entwurfskontext geschlossen werden. Bei diesem Vorgang handelt es sich somit ganz allgemein um ein *Mustererkennungsproblem*[8]. Dieses tritt in jeder der beteiligten Domänen in unterschiedlicher Form auf und muss entsprechend individuell gelöst werden. Aus den dargelegten Gedanken kann im Rahmen der vorliegenden Arbeit folgende Definition für das Reverse-Engineering formuliert werden:

Definition 3.1.2: **Reverse-Engineering in der Ingenieurwissenschaft**

Die Fähigkeit im Entwurf komplexer Systeme automatisiert auf Basis eines Weltmodells aus konkreten Modellen die zugrundeliegenden abstrakten Modelle ableiten zu können. Bei der Umsetzung muss ein domänenspezifisches Mustererkennungsproblem vor einem vielfältigen Lösungsraum gelöst werden.

Im nachfolgenden Abschnitt wird gezeigt, wie eine automatisierte Mustererkennung für die im Ingenieurentwurf elementar wichtige Domäne der Geometrie formuliert und umgesetzt werden kann. Die Grundidee ist die Verknüpfung formaler Ontologien mit analytisch berechneten, geometrischen Zentralmomenten, um ein geometrisches Weltmodell zu definieren.

Reverse-Engineering in der Geometriedomäne

Das Reverse-Engineering in der Geometriedomäne ist der inverse Vorgang der Geometriesynthese. Der nachfolgend vorgestellte Ansatz aus [Schopper und Rudolph, 2018] soll an einem Spezialfall aufzeigen, wie die abstrakt formulierte Forderung einer Mustererkennung in einem automatisierten Framework praktisch umgesetzt werden kann. Eingeschränkt wird das Problem durch die Annahme, dass die Geometrie des zu untersuchenden geometrischen Entwurfsartefakts ausschließlich aus Basiskörpern einer Teilebibliothek – der domänenspezifischen Ausprägung des Weltmodells aus Definition 3.1.2 – hervorgegangen ist. Als zusätzliche Bedingung müssen diese Basiskörper parametrisierbar sein, um Eingang in das Basismodell finden zu dürfen. Darunter fallen beispielsweise Konstruktionselemente wie Platten, Winkel, Gelenke, Schrauben und viele weitere, die oftmals in Normdatenbanken zu finden sind. Der für die Inverse benötigte Mustererkennungsmechanismus muss folglich nur Objekte erkennen, die bereits bekannt oder aus bekannten Basisentitäten zusammengesetzt sind.

Philosophisch gesehen wird hier davon ausgegangen, dass nur konstruiert werden kann, was in der Einhüllenden des beschreibenden Geometriesynthesemodells umfasst ist. Emergenz – die spontane Herausbildung neuer Begriffe mit neuen Eigenschaften [Helbig, 2020, S. 675] (siehe hierzu auch [Riestenpatt gen. Richter und Rudolph, 2020]) – während der Synthese wird hier

[8] Der Mensch löst ein solches Mustererkennungsproblem oft mühelos mithilfe seines allgemeinen Weltverständnisses. Diesen Vorgang maschinell abzubilden, stellt allerdings eine große Herausforderung dar.

ausgeschlossen. In voller Allgemeinheit kann emergentes Verhalten dazu führen, dass nicht zu jedem Problem eine Inverse formuliert werden kann.

Für den vorliegenden Spezialfall muss in der Geometrieontologie nun eine Formulierung gefunden werden, in der Objekte wie Platten, Schrauben etc. umfasst und durch ihre geometrischen Eigenschaften eindeutig zugeordnet werden können. Hierfür eigenen sich dreidimensionale geometrische Zentralmomente, die einen individuellen Volumenkörper durch eine Reihe numerischer Werte eindeutig repräsentieren [Melan, 2003]. Die Anwendung solcher Zentralmomente für die Objektklassifizierung ist wissenschaftlich lange erprobt[10].

In [Schopper und Rudolph, 2018] wurden die Bauteile hierfür geeignet parametrisiert und diese Parameter als Eigenschaften in der jeweiligen Objektklasse hinterlegt. Eine Platte hat beispielsweise die Eigenschaften Länge a, Breite b und Höhe c. Die Besonderheit des Ansatzes ist, die Zentralmomente zusätzlich zu den ähnlichkeitsmechanisch transformierten numerischen Werten (siehe [Melan, 2003] und [Kaiser und Rudolph, 2018]) auch *analytisch* für jedes Objekt zu bestimmen. Dies geschieht nach folgender Berechnungsvorschrift:

$$\psi_{pqr} = \iiint\limits_{B(P_i)} (x - x_{cg})^p (y - y_{cg})^q (z - z_{cg})^r \, dx \, dy \, dz \tag{3.1}$$

$B(P_i)$ ist dabei das in die Integrationsgrenzen überführte Definitionsgebiet der objektzugehörigen Indikatorfunktion f in Abhängigkeit der Parameter P_i. Da dies in Gleichung (3.1) aus praktischen Gründen der Laufzeit und Speicheradressierung nicht für unendlich viele Zentralmomente realisiert werden kann, werden nach [Kaiser und Rudolph, 2018] 19 Parameterausprägungen für p, q und r für die Mustererkennung ausgewertet und verglichen. Der auszuwertende Satz an (p, q, r)-Tripeln ist in Tabelle 3.1 dargestellt.

Tabelle 3.1: Auszuwertender Satz an (p, q, r)-Tripeln für geometrische Zentralmomente [Kaiser und Rudolph, 2018]

$(0,0,0)$	$(0,0,1)$	$(0,0,2)$	$(1,2,0)$	$(0,1,2)$	$(0,0,3)$	$(0,0,4)$
$(1,0,0)$	$(2,0,0)$	$(2,1,0)$	$(1,0,2)$	$(3,0,0)$	$(4,0,0)$	
$(0,1,0)$	$(0,2,0)$	$(2,0,1)$	$(0,2,1)$	$(0,3,0)$	$(0,4,0)$	

Bei der Klassifikation werden für ein zu erkennendes Objekt die zugehörigen, bezüglich der räumlichen Lage und Skalierung invarianten Zentralmomente Π_i (siehe [Kaiser und Rudolph, 2018]) numerisch bestimmt und ein Koeffizientenvergleich mit den entsprechenden Einträgen in den Bauteilklassen durchgeführt. Bei Übereinstimmung aller Einträge ist eine Zuordnung eines Bauteils zu einer Bauteilklasse gefunden. Für die Rekonstruktion der Modellparameter werden anschließend die *gewöhnlichen* μ-Zentralmomente des Bauteils bestimmt und mit den in der Bauteilklasse hinterlegten symbolischen ψ-Zentralmomenten verglichen. Dieser Koeffizientenvergleich führt auf ein nicht lineares Gleichungssystem, das mit einem adäquaten Verfahren gelöst und somit der gesuchte Parametersatz des Metamodells $P(i)$ aus dem vorhandenen Modell extrahiert werden kann. Sollte die Anzahl an Eigenschaftsparametern einer Bauteilklasse

[10] Hier können z. B. [Hu, 1962], [Cybenko et al., 1996] oder [Melan, 2003] angeführt werden.

19 übersteigen, können beliebig weitere (p, q, r)-Tripel ergänzt werden. Wichtig ist, dass es sich bei den zusätzlichen Gleichungen immer um von den bisherigen Zentralmomenten linear unabhängige Gleichungen handelt. Andernfalls können keine neue Information daraus gewonnen werden. In [Schopper und Rudolph, 2018] wird zur Lösung des Gleichungssystems das NEWTON-RAPHSON-Verfahren (siehe z. B.[Huckle und S. Schneider, 2006, 246 ff.]) angewendet.

Dieser Vorgang der Mustererkennung wird um ein Vielfaches vereinfacht, wenn das finale Modell bereits Resultat eines RTE-Verfahrens war. Im Allgemeinen kann der Einstiegspunkt des RTE aber auch ein Reverse-Engineering-Prozess, basierend auf einem externen Entwurfsartefakt, sein. Dieser Modellbildungsbeginn ist – mit gewissen Einschränkungen – ebenfalls möglich und wird in Abschnitt 5.2 später am Beispiel eines Satellitengehäuses noch ausführlich dargestellt.

3.1.3 Re-Engineering

Im Rahmen der vorliegenden Arbeit wird das Re-Engineering als Synonym für eine *Umkonstruktion* im weitesten Sinne verwendet. In der klassischen Literatur wie z. B. PAHL/BEITZ [Pahl und Beitz, 2013] kommt dem Re-Engineering die Varianten- oder Anpassungskonstruktion am nächsten (siehe auch Abschnitt 2.2.1). Die Umkonstruktion kann in diesem Kontext auf verschiedenen Abstraktionsebenen stattfinden. Stellt man sich eine Repräsentation des Entwurfs als Graph vor, der sämtliche Aspekte von der funktionalen über die physikalische bis zur geometrischen Domäne in Repräsentationsknoten und Kanten umfasst, so kann die Änderung einer oder mehrerer Eigenschaften eines Knotens, das Hinzu- oder Wegnehmen eines Knotens oder Links innerhalb des Graphen oder der Tausch einzelner Knotens bereits als eine Umkonstruktion aufgefasst werden. Die Gültigkeit einer solchen Graphentransformation kann vor dem Hintergrund einer Basisontologie validiert werden, muss aber anschließend im lokalen Modellkontext nochmals auf globale Konsistenz geprüft werden, da durch die Ersetzung Nebeneffekte auftreten können [Rudolph, 2006; Rudolph, 2013]. Hierfür sind weitere Zwangsbedingungen in Form von Gleichungen und Regeln zu berücksichtigen, die über die Ontologie alleine nicht abbildbar und somit nicht validierbar sind. Der Austausch eines Knotens kann mit einem Knoten auf demselben, einem höheren oder einem niedrigeren Abstraktionsniveau gegenüber der Definition in der Basisontologie geschehen. Ausschlaggebend hierfür ist deren Struktur, die allerdings der Natur der Modellbildung entsprechend nicht eindeutig ist. Wird für die Ableitung der Basisontologie die FCA (siehe Abschnitt 2.2.3) eingesetzt, wird die abgeleitete Ontologie im Kontext dieser Arbeit als *objektorientierte Universalontologie* bezeichnet.

Die Konkretisierungstätigkeit ist typisch für den allgemeinen Entwurfsprozess. In allen gängigen Beschreibungen wird eine abstrakte Beschreibung in verschieden ausgeführten Zwischenschritten in die finalen Bauteile eines Produkts überführt. PAHL und BEITZ beschreiben diesen Vorgang beispielsweise als Übertragung von Anforderungen, über abstrakte Produktfunktionen und Prinziplösungen auf Bauteile (siehe Abbildung 2.8). Funktionale Repräsentationen können gegebenenfalls noch ohne Nebeneffekte auskommen, aber spätestens bei der Übersetzung in echte Bauteile, entstehen stark kontextabhängige Lösungen, da jedes Bauteil eine individuelle Form, eine Masse und ein Material besitzt, Energie verbraucht, absorbiert oder abgibt und noch viele weitere inhärente Eigenschaften in den Entwurf einbringt [Rudolph, 2006]. Hat eine oder

mehrere dieser Größen Einfluss auf eine feste Randbedingung an den Entwurf – beispielsweise ein maximal zulässiges Gesamtgewicht – muss nach jedem Konkretisierungsschritt die Validität des aktuellen Entwurfs im Gesamtkontext geprüft werden. Diese ist einer der Hauptgründe für das Auftreten von Iterationsschleifen im Entwurfsprozess [Rudolph, 2006].

Erschwerend kommt beim Re-Engineering im RTE-Kontext hinzu, dass die beim Forward-Engineering inkrementell erfolgte Wissensanreicherung neue Randbedingungen in den Entwurf einbringen kann, die einerseits neue Möglichkeiten weiter vorne im Entwurf ermöglichen, andererseits gegebenenfalls einen Teil der bisherigen Spezifizierungen falsifizieren können. Daher muss das Validieren mit einem Reverse-Engineering gekoppelt werden, um nicht zu einem Entwurfsabbruch zu führen. Dies kann in einem reinen, vorwärts gerichteten Top-Down-Prozess nicht abgebildet werden, sondern benötigt zwingend einen Iterationsmechanismus, der fortan als *iterative Vorwärtsvariation* bezeichnet wird. Als ein Beispiel kann hier die nachgelagerte Spezifizierung des zunächst unbestimmten Materials nach der Festlegung eines Fügeverfahrens angeführt werden[11]. Zum Zeitpunkt der Konkretisierung des Fügeverfahrens als konkrete Variante handelt es sich beim Material um eine freie Randbedingung, die die Wahl des Fügeverfahrens nicht limitiert. Wird aber anschließend das Material auf eine konkrete Klasse wie z. B. Holz eingeschränkt, werden dadurch implizit Fertigungsverfahren wie beispielsweise das Schweißen ausgeschlossen, die aber im aktuellen Schleifendurchgang noch als valide Varianten vorgehalten werden. Aus diesem Grund müssen in einem iterativen Prozess noch einmal alle Domänen – und somit auch die Fügeverfahren – überstrichen werden, um die im Widerspruch zu den aktuellen Randbedingungen stehenden Varianten aussortieren zu können. Erst wenn das Gesamtdatenmodell über einen vollständigen Zyklus über alle Domänen hinweg konstant bleibt, ist das implizite Gleichungssystem bestmöglich gelöst und der Entwurf abgeschlossen. Zu diesem Zeitpunkt existieren keine widersprüchlichen Varianten mehr, es ist aber möglich, dass keine valide Variante gefunden werden konnte. Als Analogon dieser Situation kann ein bestimmtes oder überbestimmtes mathematische Gleichungssystem dienen, wobei sich einzelne Gleichungen widersprechen und somit keine Lösung errechnet werden kann ($\mathbb{L} = \{\}$).

Spezifikations- und Abstraktionscast

Das Ausspezifizieren einer abstrakteren zu einer konkreteren Ontologieklasse (vgl. [Rudolph, 2006, S. 17]) wird im weiteren Verlauf – in Anlehnung an eine bekannte Formulierung aus der Informatik – als *Spezifikationscast* bezeichnet. Diese Form der Konkretisierung ist im Kanon der klassischen objektorientierten Programmierung (OOP) nicht vorgesehen, da bei diesem Vorgang Wissen „ergänzt“ werden muss, das zur Compilezeit noch nicht vorhanden war.

Dabei spielt es keine Rolle, ob es sich um zusätzliche Instanzvariablen oder Methoden handelt, die die Basisklasse nicht umfasst hat. Durch die in der Ontologie vorgehaltene Hierarchieordnung und die vereinbarten Eigenschaften und Methoden (semantischer Kontext) sowie das im aktuellen Kontext des zur Laufzeit existierenden Entwurfsmodells durch Gleichungs- oder Constraintlösung akquirierte Wissen (pragmatischer Kontext) kann der Spezifikationscast im

[11] Die Reihenfolge der Entwurfsentscheidungen spielt durch die Iterationsschleifen prinzipiell keine Rolle. Für das Anschauungsbeispiel wurde willkürlich eine Reihenfolge festgelegt.

Rahmen des RTE trotzdem ausgeführt werden. Hierfür muss zwangsweise das alte Repräsentationsobjekt gelöscht, seine Information in die neue Instanz der ersetzenden Kindklasse übertragen und die zusätzliche Information aus dem lokalen Kontext ergänzt werden[12].

Der umgekehrte Fall, von einer spezifischeren auf eine abstraktere Klasse zu generalisieren, wird nachstehend als *Abstraktionscast* bezeichnet und ist als Analogon zum gewöhnlichen *Cast* aus der OOP zu verstehen. Die Definition von Spezifikations- bzw. Abstraktionscast sind insbesondere für die Teilaspekte des Re-, aber auch des Reverse-Engineering von großer Bedeutung. Als Abschluss dieses Abschnitts können die vorausgehend formulierten Gedanken auf folgende Definition für das Re-Engineering verdichtet werden:

Definition 3.1.3: **Re-Engineering in der Ingenieurwissenschaft**

Die Fähigkeit im Entwurf komplexer Systeme aufbauend auf einer Basiskonstruktion automatisiert Konstruktionsalternativen durch

- direkte Respezifikation (Umkonstruktion) oder
- Abstraktion und anschließende variierende Respezifikation (Variationskonstruktion)

vor dem Hintergrund eines Weltmodells generieren zu können.

Zwischenfazit Nachdem die einzelnen Entwurfsprozesse in den vorangegangen Abschnitten beschrieben wurden, lässt sich folgendes vorläufiges Fazit ziehen: Fasst man den Entwurfsprozess als iterativen Prozess mit wiederkehrenden Entwurfsmustern auf, so können die drei Bestandteile des RTE als unterschiedliche Einstiegspunkte in diesen Prozess betrachtet werden. Der Einstiegspunkt entscheidet, welche Konstruktionsaufgabe davon zunächst ausgeführt werden muss. Prinzipiell muss daher jede Form sukzessive in jede andere überführbar sein, da potenziell alle Entwurfsarten in einem gesamten Schleifendurchgang (Round-Trip) überstrichen werden. Legt man ferner die Annahme[14] zugrunde, dass es keinen Entwurf ohne Iterationsschleifen geben kann, können Forward-, Reverse- und Re-Engineering gar nicht unabhängig voneinander auftreten. Die Ausführung der einzelnen Prozesse ist nur eine Momentaufnahme eines sich wiederholenden Kreislaufs auf unterschiedlichen Ebenen der Abstraktion, in dem die Reihenfolge, in der die einzelnen Prozesse aufeinander folgen, je nach Aufgabenstellung variieren kann. Die vorausgegangenen Überlegungen führen zu folgender Definition:

Definition 3.1.4: **Round-Trip-Engineering in der Ingenieurwissenschaft**

Die Fähigkeit im Entwurf komplexer Systeme automatisiert

1. aus abstrakten alle benötigten konkreten Modelle (Forward-Engineering)
2. aus konkreten die zugrundeliegenden abstrakten Modelle (Reverse-Engineering)

[12] Für ein anschauliches Beispiel zu diesem Sachverhalt sei auf Kapitel 5 verwiesen.

[14] Bei einem mathematischen Analogon eines nicht linearen Gleichungssystems als Entwurfsdarstellung kann im Allgemeinen davon ausgegangen werden, dass das Lösungsverfahren einen iterativen Anteil hat.

3. und Modellvarianten aus einem Vorgabemodell (Re-Engineering)

ableiten zu können. Die Konsistenz aller domänenspezifischen Modellartefakte muss automatisch aufrechterhalten werden können. Kernbestandteile eines Round-Trip-Engineering Frameworks sind Synthese-, Analyse- und Interpretationsfähigkeit.

In Abbildung 3.1 ist die repetitive Abfolge der Teilprozesse des RTE als Iterationsprozess dargestellt, wobei die einzelnen Vorgänge zu einem Wechsel der Abstraktionsniveaus führen.

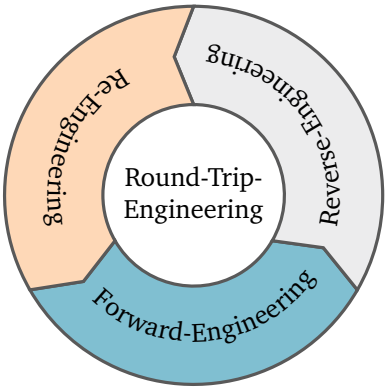


Abbildung 3.1: Round-Trip-Engineering als Iterationsprozess aus Forward-, Reverse- und Re-Engineering

3.2 Konzeption des Entwurfsrahmenwerks

Das im Folgenden vorgestellte Entwurfsrahmenwerk hätte ohne die zahlreichen Vorarbeiten anderer niemals entstehen können (vgl. u. a. Abschnitt 2.2.3). Es erhebt daher nicht den Anspruch auf eine vollständig neuartige Vorgehensweise, sondern stellt vielmehr eine intelligente Verknüpfung und Erweiterung bestehender Ansätze dar. Da den durch den/die Ingenieur/-in spezifizierten Randbedingungen eine herausragende Bedeutung im Gesamtkontext des Entwurfsrahmenwerks zukommt, erfolgt im folgenden Abschnitt zunächst eine Begriffsklärung.

3.2.1 Randbedingungen im Round-Trip-Engineering

In der Entscheidungstheorie wird eine feste Randbedingung als *Datenparameter* bezeichnet. Darunter sind „*solche exogenen Größen zu verstehen, auf die die Unternehmung keinen direkten oder indirekten Einfluss ausüben kann oder will, die sie also als Gegebenheiten hinnehmen muss bzw. hinnimmt.*“ [Kosiol, 1973]. Im Gegensatz dazu wird der Aktionsparameter definiert, der in „*einer Wirtschaftseinheit eine in ihren Wirtschaftsplan eingehende Größe [bezeichnet], welche die Wirtschaftseinheit nach eigenem Ermessen festsetzen (fixieren) kann.*“ [E. Schneider, 1934].

Im Folgenden wird für einen auf das Ingenieurwesen übertragenen Datenparameter der Begriff *feste Randbedingung*, für den übertragenen Aktionsparameter der Begriff *freie Randbedingung* verwendet. Als Randbedingung kann prinzipiell nur das spezifiziert werden, was die zugrunde liegenden Domänenontologien in ihren „Einhüllenden“ umfassen. Im Sinne der

formalen Begriffsanalyse (FBA) (siehe Abschnitt 2.2.3) können Randbedingungen auf Begriffsebene oder auf Eigenschaftsebene definiert sein. Randbedingungen, die aus der Instanziierung einer Ontologieklasse hervorgegangen sind (Begriffsebene), müssen als *diskrete* Randbedingung an den Entwurf aufgefasst werden. Daneben existieren Randbedingungen auf Eigenschaftsniveau, die auf der Zuweisung eines Wertes zu einem in der Ontologieklasse angelegten Feld basieren. Darunter fallen *stringbasierte* Randbedingungen, *symbolische* Randbedingungen in Form mathematischer Gleichungen bzw. Gleichungssysteme und *numerische* Randbedingungen, die einen Wert bzw. Wertebereich einer Instanzvariablen festlegen.

Tabelle 3.2: Verschiedene Arten von Randbedingungen

1.	instanzbasiert
	- bestimmt-diskret
	- unbestimmt-diskret
2.	feldbasiert
	- stringbasiert
	- symbolisch
	- numerisch
	- instanzverknüpft

In einer Implementierung setzen Randbedingungen auf Feldebene das Vorhandensein von Instanzvariablen mit einem passenden Datentyp innerhalb einer Ontologieklasse voraus (z. B. `String`, `int` oder `double[]`). Im Spezialfall der Vereinbarung einer anderen Ontologieklasse als Eigenschaft der betrachteten Klasse geht die feldbasierte Randbedingung wieder in eine diskrete Randbedingung über (*instanzverknüpfte Randbedingung*). Auch innerhalb der feldbasierten Randbedingungen sind die Übergänge fließend. Beispielsweise ist eine symbolische Gleichung der Form $var = const.$ gleichzeitig eine numerische Randbedingung. Eine hierarchische Struktur der zugrunde liegenden Ontologie vorausgesetzt[16], können diskrete Randbedingungen auf verschiedenen Abstraktionsniveaus angegeben werden.

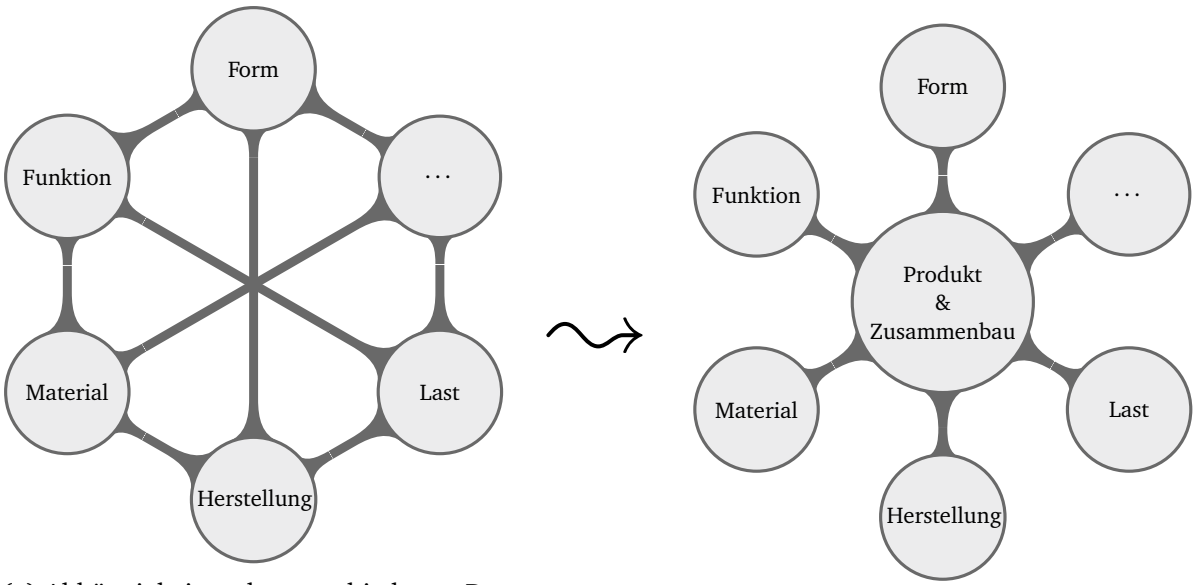
Zur Verdeutlichung dieses Sachverhalts sei wieder das Beispiel des Materials bemüht: Die Ontologieklasse `Material` ist auf einem höheren Abstraktionsniveau definiert als die spezifischere Ontologieklasse `Metall`, wobei diese wiederum abstrakter ist als die sehr spezifische Ontologieklasse der Aluminiumlegierung `Al6061TS`. Die Hierarchiebeziehungen in diesem Beispiel sind als sogenannte *ist-ein*-Beziehungen (Vererbung) modellierbar: `Al6061TS` *ist-ein* `Metall`, `Metall` *ist-ein* `Material`. Somit gilt auch unmittelbar `Al6061TS` *ist-ein* `Material`.

Die Blätter der Ontologien sowie prinzipiell alle unverknüpften Ontologieklassen, d. h. Klassen ohne Assoziationen oder Vererbungsbeziehungen, werden im weiteren Verlauf als *fest-diskrete Randbedingungen (FDR)* bezeichnet. Sie zeichnen sich insbesondere dadurch aus, dass auf Sie kein Spezifikationscast (siehe Abschnitt 3.1.3) angewendet werden kann. Alle anderen diskreten Randbedingungen sind zunächst als *variabel-diskrete-* (*VDR*) oder *kontinuierlich-diskrete Randbedingungen (KDR)* aufzufassen. Da FDR den Blättern der Ontologie und allen unverknüpften Klassen entsprechen, hängt deren Detaillierungsgrad maßgeblich davon ab, wie ausführlich die zugehörige Domänenontologie modelliert wurde. Durch die Einführung einer *Maximaltiefenvariablen* τ_{max}, wie sie in der Informatik auch beispielsweise bei der begrenzten Tiefensuche definiert wird [Korf, 1985], ist es möglich, aus VDR FDR zu machen, da der tiefer als τ_{max} liegende Teil der Ontologie in diesem Fall nicht berücksichtigt wird und so aus einer Zwischenschicht eine Blattschicht entsteht. Gemeinsam mit der Definition einer *Minimaltiefen-*

[16] Dies kann durch die Anwendung der FCA zur Ontologiedefinition sichergestellt werden.

variablen τ_{min} ermöglicht dies ein zielgerichtetes Einschränken des Entwurfsraums auf eine Teilmenge der jeweiligen Domänenontologie und wirkt einer Variantenexplosion entgegen.

Die verschiedenen Formen der Randbedingungen treten – je nach gewählter Modellierung – potenziell in allen für den Entwurf betrachteten Domänen auf. Da die Domänen nicht unabhängig voneinander sind, sondern im Extremfall alle untereinander verknüpft sind, führt dies auf eine mit den Abhängigkeiten quadratisch wachsende Anzahl an Schnittstellen. Die Modellierung der Gesamtheit der Domänenontologien in einem zentralen Basismodell bietet hier den Vorteil, dass die Anzahl an Schnittstellen bei n Domänen auf n reduziert werden kann. Dies ist auch ein Kerngedanke der Basic Formal Ontology (BFO) [Smith und Grenon, 2021], wobei die formale Basisontologie „*eine kleine Ontologie auf höherer Ebene ist, die für die Unterstützung der Informationsbeschaffung, -analyse und -integration in wissenschaftlichen und anderen Bereichen entwickelt wurde.*“ [Smith und Grenon, 2021]. Ebenfalls in diese Richtung zielt das Ontology Definition Metamodel (ODM) [ODM, 2014] der OMG. Damit soll unter anderem eine „*[. . .] standardbasierte, modelltheoretische Semantik für die unterstützten Wissensrepräsentationssprachen [definiert werden], die ausreichen, um [. . .] Ontologien zu verstehen, zu validieren und anzuwenden,*“. Auch wenn in der vorliegenden Arbeit weder die BFO noch die ODM direkt verwendet werden, so werden die Ontologien mit einer sehr ähnlichen Herangehensweise definiert. Für eigene Implementierungen oder auch eine künftige Weiterentwicklungen des hier vorgestellten Frameworks sollte untersucht werden, ob sich einer der beiden Standards dafür eignet, direkt als Modellierungsgrundlage angewendet zu werden. Die Reduzierung der Schnittstellen durch Einführung eines zentralen Modells ist in Abbildung 3.2 für die im Rahmen der vorliegenden Arbeit relevanten Domänen illustriert.



(a) Abhängigkeiten der verschiedenen Domänen im Entwurf mit $\frac{n\,(n-1)}{2}$ Schnittstellen

(b) Abhängigkeiten der verschiedenen Domänen im Entwurf mit n Schnittstellen

Abbildung 3.2: Reduzierung der Schnittstellenanzahl durch ein zentrales Modell, angelehnt an [Reichwein, 2011, S. 3]

Die Reduzierung der Schnittstellenanzahl hat auch direkte Auswirkungen auf die spätere Ableitung einer problemspezifischen Entwurfssequenz, da sich mit sinkender Schnittstellenanzahl auch die Anzahl benötigter Iterationszyklen verringert (siehe hierzu auch Abschnitt 3.2.3).

Nachdem die verschiedenen Formen der Randbedingungen in diesem Abschnitt vorgestellt wurden, soll im Folgenden deren Anwendung für die Problemspezifikation dargelegt werden. Die Erkenntnis, dass die Randbedingungen einen entscheidenden Einfluss auf die auszuführende Entwurfssequenz haben, ist dabei ein zentrales Thema.

3.2.2 Wissensformulierung und Modellierung der relevanten Domänen

In der vorliegenden Arbeit sollen die für den Entwurf eines Produkts und dessen Zusammenbaus fünf wesentlichen Domänen – Form, Funktion, Material, Herstellungsprozess und Last – herausgegriffen werden. Darüber hinaus existiert eine Vielzahl weiterer Domänen, die aber über den Rahmen dieser Arbeit hinausführen und daher nicht gesondert betrachtet werden sollen. Prinzipiell ist das Entwurfsrahmenwerk dergestalt konzipiert, dass es mit beliebig vielen weiteren Domänen erweitert werden kann. Bevor in Kapitel 4 eine prototypische Implementierung des Domänenwissens in Form von Ontologieklassendiagrammen erfolgt, sollen in diesem Kapitel Grundüberlegungen zum benötigten Domänenwissen und zum Zusammenhang der unterschiedlichen Domänen angestrengt werden. In Abbildung 3.2b sind die betrachteten Domänen bereits aufgezeigt worden. Die Form eines Produktes hat wesentlichen Einfluss auf den Entwicklungsprozess. Daneben spielen auch die durch das Produkt zu erfüllenden Funktionen eine wichtige Rolle. Das Material bzw. die Materialien, aus dem das Produkt gefertigt werden soll, schränken andere Domänen, wie beispielsweise den zum Einsatz gebrachten Fertigungsprozess ein. Der Fertigungsprozess bzw. die Fertigungsprozesse haben wiederum Rückkopplungen auf die übrigen Domänen. Die Lasten – darunter fallen alle physikalischen Lasten wie mechanische, thermische und viele weitere – haben ebenfalls Einfluss auf alle anderen Domänen. So wird hier nochmals deutlich, dass das zentrale Datenmodell von entscheidender Bedeutung ist, da hierdurch die Verknüpfung aller Domänen einfach sichergestellt werden kann. Nachfolgend sollen nun die einzelnen Domänen und Ihre Kopplungen genauer betrachtet werden.

Funktion

In Abschnitt 2.2.1 wurde der Begriff der Funktion im Kontext des Produktentstehungprozess bereits allgemein eingeführt und in Definition 2.2.6 zusammengefasst. Im Unterschied zu den anderen in dieser Arbeit als relevant betrachteten Einflussbereichen, die im weiteren Verlauf dieses Unterkapitels noch besprochen werden, ist die Funktion als vollständig abstrakte Domäne ohne physische Repräsentation am fertigen Produkt zu betrachten.

Die Funktion ist eng mit dem Zweck eines Produkts verknüpft [Pahl und Beitz, 2013, S. 244] und wird in den meisten Fällen in einem Top-down-Dekompositionsprozess aus der Problemstellung abgeleitet. Als Resultat ergibt sich eine Menge von Hauptfunktionen, die anschließend weiter untergliedert und bis auf Basisfunktionen heruntergebrochen werden können. Daraus folgt schließlich die Funktionsstruktur des zu entwerfenden Produkts (vgl. [Pahl und Beitz,

2013, S. 242–246, 344–347]). In der klassischen Denkweise ist die Funktion somit *immer* den anderen Domänen und deren verbundenen Aktivitäten und Prozessen vorgelagert.

Insbesondere durch die in dieser Arbeit vertretene Sicht auf den Entwurf als Iterationsprozess muss die Funktionsdomäne um einen neuen Blickwinkel ergänzt werden: Eine Funktion kann auch Folge einer im aktuellen Kreislauf nachgelagert vorgenommenen Konkretisierung – beispielsweise im Rahmen der Gestaltung – sein, die erst im nächsten Durchlauf in ihrer Konsequenz offenbar wird. Auch im Kontext des Reverse-Engineering (siehe Abschnitt 3.1.2) kommt ein weiterer Gesichtspunkt der Funktion hinzu: Durch das Vorhandensein bestimmter, in einem Mustererkennungsprozess erkannter Zusammenhänge, kann beispielsweise – zu einem gewissen Teil – in einem Bottom-up-Prozess auf die ursprünglich geforderte Funktion beziehungsweise Funktionsstruktur eines Produkts zurückgeschlossen werden. Folgt man der Auffassung von PAHL und BEITZ, wonach eine Funktion Energie, Stoff und Signal sowohl als Ein- als auch als Ausgangsgrößen besitzt [Pahl und Beitz, 2013, S. 240], unterstreicht dies die prinzipielle Möglichkeit der Umkehrung deren Richtungen. Es sei an dieser Stelle aber noch einmal darauf hingewiesen, dass die Dimensionalität der Räume vor und nach der Transformation durch die Funktion verschieden sein können[17], was in der Konsequenz bei der Invertierung auf das bereits mehrfach angesprochene Problem der Mehrdeutigkeit führen kann. Die impliziten Kopplungen der verschiedenen Domänen sind zum Teil auf Funktionen zurückzuführen, die durch unterschiedliche Funktionsträger disparater Domänen umgesetzt werden.

Die Modellierung der Funktionsdomäne im Entwurfsrahmenwerk, die der klassischen als auch der novellierten Sichtweise gerecht wird, muss mindestens die drei Grundklassen Energie, Stoff und Signal beinhalten. Darüber hinaus müssen die Umsätze dieser drei Klassen modelliert werden. Auf einem sehr abstrakten Niveau kann man sich hierfür an den in [Pahl und Beitz, 2007, S. 248] aufgelisteten Vorschlägen orientieren. Diese sind in Tabelle 3.3 aufgeführt.

Tabelle 3.3: Energie-, Stoff- und Signalumsatz [Pahl und Beitz, 2007, S. 248]

Energieumsatz	**Stoffumsatz**	**Signalumsatz**
Energie wandeln	Stoffumsatz wandeln	Signal wandeln
Energiekomponente ändern	Stoffabmessungen ändern	Signalgröße ändern
Energie mit Signal verknüpfen	Stoff mit Energie verknüpfen Stoff mit Signal verknüpfen Stoffe miteinander verknüpfen	Signal mit Energie verkn. Signal mit Stoff verkn. Signale verknüpfen
Energie leiten	Stoff leiten	Signal leiten
Energie speichern	Stoff speichern	Signal speichern

Zusammenfassend werden aus der obigen Tabelle für die Implementierung und weitere Ausdetaillierung des Funktionsontologie in jedem Fall die Begriffe *wandeln*, *ändern*, *verknüpfen*, *leiten*

[17] In der Vorstellung der Entwurfskonkretisierung nach PAHL und BEITZ wird prinzipiell von einer höheren auf eine niedrigere Dimensionsebene abgebildet.

und *speichern* benötigt. Aus Sicht des Autors der vorliegenden Arbeit sollte noch der Begriff *trennen* in dieser Auflistung ergänzt werden. Mit den genannten Modellierungsbausteinen sollten auf abstraktem Niveau alle relevanten Elemente der Funktionsdomäne abgebildet und somit modellierbar sein. Für ein konkretes Entwurfsproblem müssen diese dann im Entwurfsrahmenwerk um problemspezifische Eigenheiten erweitert und angepasst werden.

Form

Zu Beginn dieses Abschnitts soll darauf hingewiesen werden, dass im weiteren Verlauf bezugnehmend auf [Ashby, 2005, S. 19] bei Verwendung des Wortes Form die „*externe Makro-Form*“ eines Bauteils bzw. Gesamtprodukts verstanden werden soll. In dieser Arbeit soll die „*interne Mikro-Form*“ – wie beispielsweise Wabenstrukturen – nicht explizit betrachtet werden[18].

Eine herausragende Stellung nimmt die Form bei allen Produkten ein, die auch ästhetische Anforderungen zu erfüllen haben. Ein gängiges Vorgehen in der industriellen Praxis stellt die Vorgabe einer sogenannten „*Class-A-Oberfläche*“ [Farin, 2006] durch einen Designer dar, die anschließend als (weitgehend) feste Randbedingung für die weitere Auskonstruktion beibehalten werden muss. Als naheliegendes Beispiel einer solchen Arbeitsweise kann die stark designgetriebene Automobilindustrie angeführt werden, wobei eine Optimierung bezüglich weiterer Randbedingungen an die Grundformvorgabe des Designers stattfinden kann.

Eine gegenteilige Philosophie verfolgt der ursprünglich aus der Architektur stammende Ansatz „ *form follows function*[19]“ [Sullivan, 1896, S. 408]. Dabei nimmt die Form die Rolle einer variablen Randbedingung ein, die sich als Konsequenz der Erfüllung anderer, dominanter Anforderungen ableiten lässt. Die Luftfahrtindustrie verfolgt diesen Ansatz beispielsweise beim Tragflügelentwurf, bei dem die Erfüllung eines gewünschten Kennfelds des Auftriebsbeiwerts c_A über dem Widerstandsbeiwert c_W[20] über eine typische Mission gefordert wird. Wie das Beispiel des Flügels zeigt, kann die Form eines Bauteils prinzipiell auch direkt eine konkrete Funktion erfüllen. Als weiteres Beispiel können hier Düsen oder Diffusoren angeführt werden, deren Funktion an eine konvergierende oder divergierende Form des Bauteils geknüpft ist. Es sei aber auch angemerkt, dass eine weitere Abhängigkeit von der Machzahl – im Kontext dieser Arbeit also aus dem Bereich Last – besteht. Dies verdeutlicht eindrücklich, dass die verschiedenen Domänen alle zumindest implizit miteinander verknüpft sein können. „*Form follows daher nicht nur function, sondern sie kreiert sie auch. Systematisch betrachtet, wird daher aus dem stofflosen Raum der Geometrie die (ebenfalls noch stofflose) Form, die sich mit den physikalischen Wirkungen im Stoff zu der Gestalt verbindet.*“ [Rudolph und Kröplin, 2005, S. 35].

Der Zusammenhang zwischen der Form und den Herstellungsprozessen ist evident, da nicht jede gewünschte geometrische Form durch jeden Fertigungsprozess realisierbar ist. Beispielsweise lässt sich in einem Rapid-Prototyping-Verfahren nahezu jede Form erzeugen, aber nicht jedes Material ist für diesen Prozess geeignet [Um, 2018]. In einem Strangpressverfahren können extrudierte Formen in hohen Stückzahlen und vorwiegend aus metallischen Werkstoffen er-

[18] Prinzipiell ist auch die Mikro-Form im Geltungsbereich dieses Frameworks enthalten, wird hier aber für die grundsätzliche Beschreibung der Vorgehensweise nicht in diesem Detail aufgelöst.

[19] Zu deutsch: „*die Form folgt aus der Funktion*“

[20] In der Strömungslehre spricht man hierbei von einem Polardiagramm [Bschorer et al., 2021, S. 317].

zeugt, aber keine mehrfach gekrümmten Bauteilformen hergestellt werden[21] [Fritz, 2018]. Des Weiteren spielen die Fügeverfahren bei einem aus mehreren Komponenten zusammengesetzten Produkt eine Rolle. Eine Schraub- oder Klebeverbindung benötigt beispielsweise immer zwingend eine Überlappung an der Stoßstelle [Wittel et al., 2019], welche wiederum Auswirkungen auf die Form des Gesamtbauteils hat. Die Verknüpfung von Form und Material ist darüber hinaus auch unabhängig von den Herstellungsverfahren direkt über die Eigenschaften des zu verwendenden Materials gegeben. Im Zusammenhang mit einer geforderten Lastaufnahme (z. B. Kraft, Moment oder Temperatur) ist stets eine materialabhängige Mindestdicke eines Bauteils erforderlich, die wiederum Einfluss auf die Form haben kann.

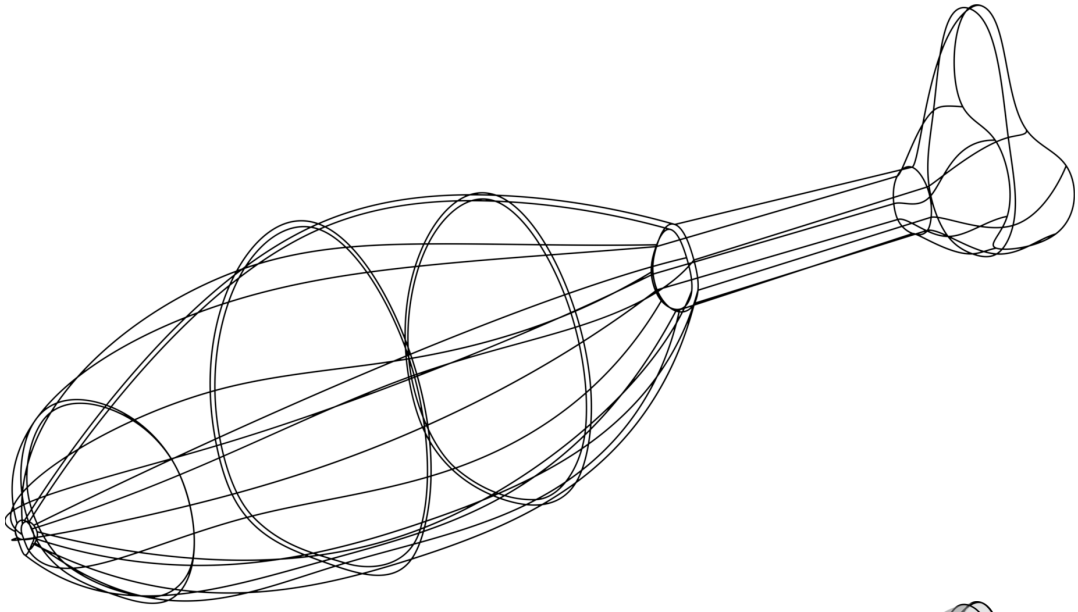
Ein Framework, wie es in dieser Arbeit beschrieben wird, muss in der Lage sein, diese im- und expliziten Kopplungen der Form mit den restlichen Domänen abzubilden. Die prinzipielle Modellierbarkeit einer beliebigen Form ist hierfür eine Grundvoraussetzung. In dieser Arbeit wird – im Falle der Form als freier Randbedingung – das Formmodell über das sogenannte Begrenzungsflächenmodell (Boundary Representation Methode (BREP)) abgebildet, die auf den Grundelementen Punkt, Linie, Fläche und Volumen aufbaut. Eine Menge an Punkten beschreibt einen Linienzug, ein geschlossener Linienzug definiert eine Fläche, eine Menge an Flächen beschreibt das eingegrenzte Volumen [Mäntylä, 1987]. Konkret wird im Entwurfsrahmenwerk eine zu modellierende Form zunächst durch ein Drahtgittermodell als Summe von Spline-Linienzügen beschrieben. Um dieses Drahtgittermodell mit einer oder mehreren Flächen zu füllen, muss ein Kurvennetzwerkinterpolationsproblem gelöst werden. Dazu werden im Entwurfsrahmenwerk globale GORDON-Flächen [Gordon, 1969] verwendet, da diese gegenüber Oberflächen aus zusammengesetzten COONS-Flächen [Farin, 1993] an den Randkurven des Drahtgitters krümmungs- und nicht nur tangentenstetige Übergänge der Oberflächen und damit insgesamt eine glattere Gesamtoberfläche erzeugen[22]. Die Implementierung des zugehörigen Algorithmus wurde stark inspiriert von [Siggel et al., 2019] und in purem Java® [Java, 2021] Code umgesetzt, um direkt mit den Objekten des im DC43 gewrappten Geometriekerns Open-CASCADE Technology® [Open Cascade, 2022] interagieren zu können.

Wenn es sich bei der Form um eine feste Randbedingung (Vorgabe) handelt und diese nicht im Rahmen eines Forward-Engineering direkt im Entwurfsrahmenwerk generiert wurde, muss zunächst über einen Reverse-Engineering-Prozess (siehe Abschnitt 3.1.2) ein zugehöriges Modell abstrahiert werden. Dieses Modell wird anschließend weiterverarbeitet und muss dabei – je nach sonstigen gesetzten Randbedingungen – gegebenenfalls lokal angepasst werden, wobei die Grundform des Bauteils stets erhalten bleiben muss. Dieser Fall tritt auf, wenn das Bauteil aufgrund weiterer zu erfüllender Randbedingungen aufgeteilt, Stützstrukturen angebracht oder sonstige Manipulationen an der äußeren Form durchgeführt werden müssen.

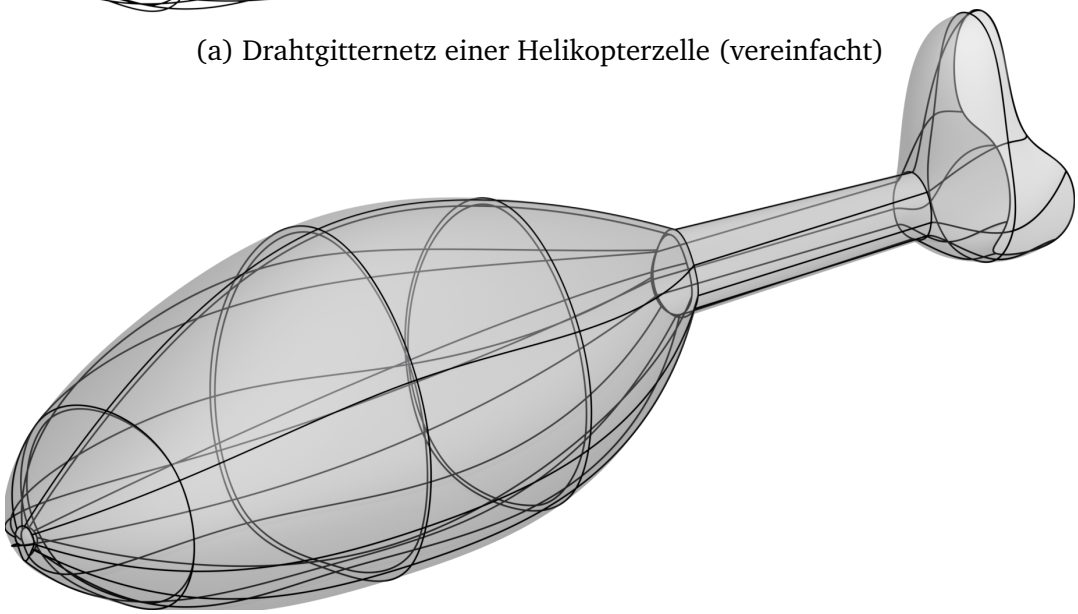
In Abbildung 3.3 sind exemplarisch die Drahtgittermodelle einer Hubschrauberhauptzelle, des Heckauslegers und des Fenestrons dargestellt (obere Abbildung), die anschließend mit jeweils einer einzigen GORDON-Fläche interpoliert werden (untere Abbildung).

[21] Mit einem parallelen oder nachgelagerten weiteren Verarbeitungsprozess lassen sich mehrfach gekrümmte Bauteile mit einem Strangpressverfahren als Basis erzeugen, dann aber mit deutlich erhöhtem Aufwand.

[22] Gordon-Surfaces stellen per Definition eine C2-Stetigkeit der Interpolationsfläche sicher. Für eine umfassende Darstellung des Themenkomplexes sei an dieser Stelle auf [Gordon, 1969] und [Siggel et al., 2019] verwiesen.



(a) Drahtgitternetz einer Helikopterzelle (vereinfacht)



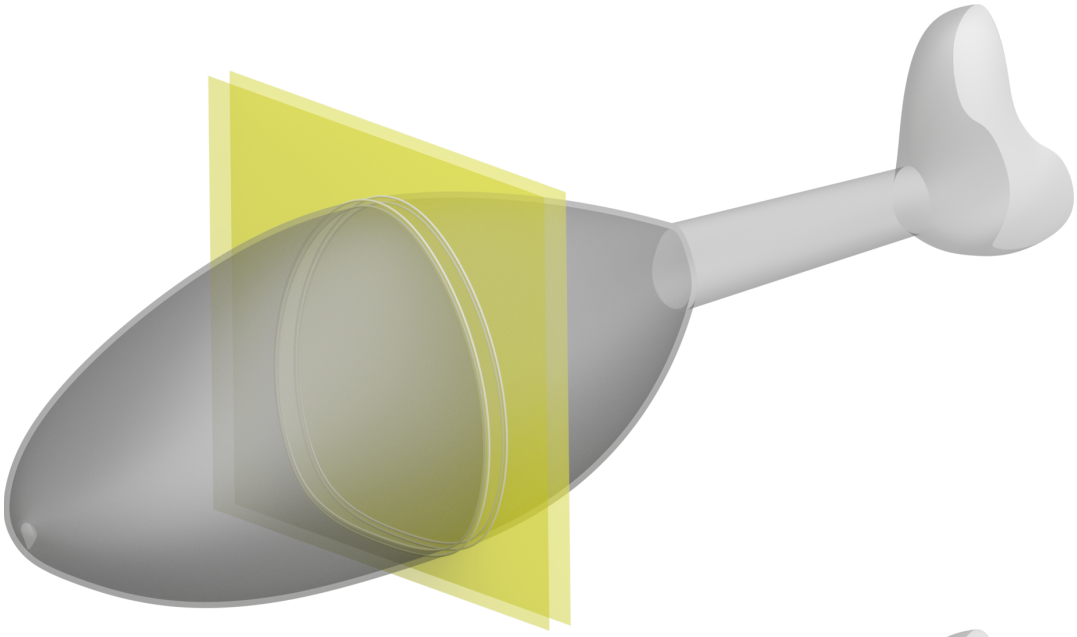
(b) Flächenmodell der obigen Helikopterzelle (halbtransparente Darstellung)

Abbildung 3.3: Vom Drahtgittermodell zur Flächenkonstruktion

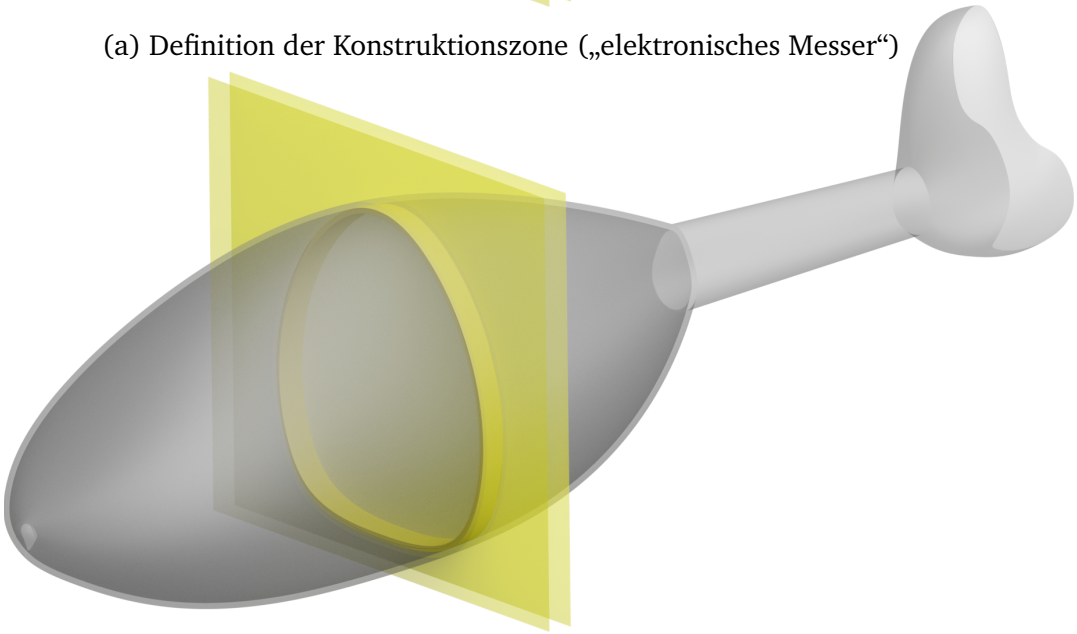
Die Parametrik und Topologie der Form kann über das Drahtgittermodell vollumfänglich variiert werden. Durch die Anwendung der Classfunction Shapefunction Transformation (CST)-Methode (siehe [Kulfan, 2008]) kann eine möglichst geringe Anzahl beschreibender Parameter erreicht werden. Für eine genauere Beschreibung dieser Formmodellierung sei auf [Schöne, 2020] – eine im Rahmen dieser Arbeit betreuten Bachelorarbeit – verwiesen. Es sei angemerkt, dass die CST-Methode keine zwingende Voraussetzung im Entwurfsrahmenwerk darstellt und prinzipiell auch eine beliebige andere gegebene Parametrisierung vorgenommen werden kann.

Das elektronische Messer In vielen Anwendungsfällen ist davon auszugehen, dass die Primärform der Konstruktion als Designvorgabe in Form einer festen Randbedingung (vgl.

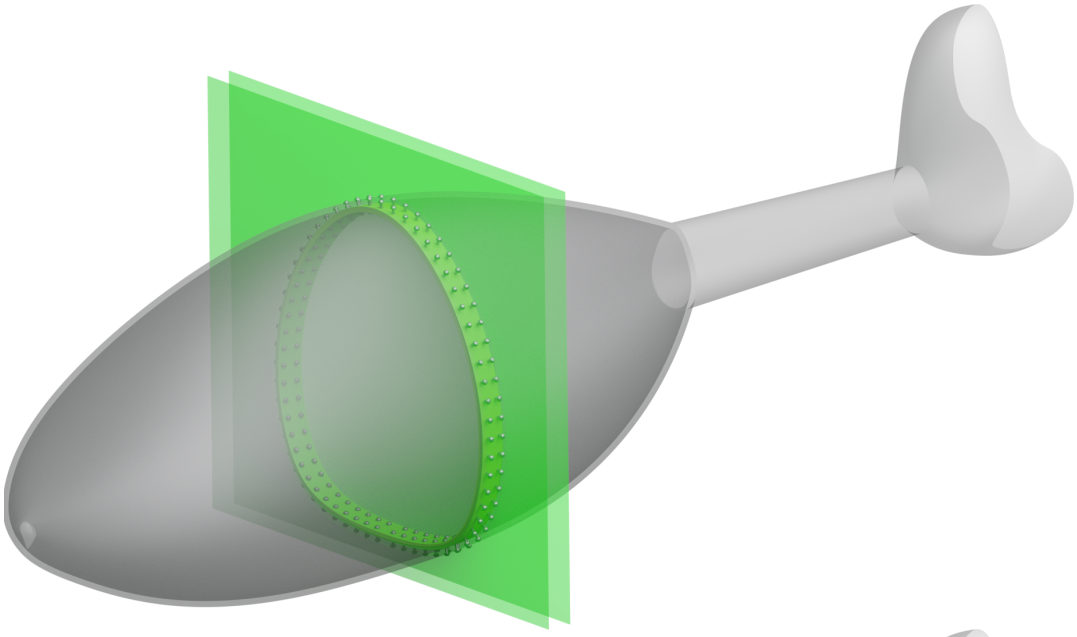
Abschnitt 3.2.1) in die Problemdefinition eingeht. In diesem Sinne ist das in einem Forward-Engineering Prozess initial generierte oder das alternativ in einem Reverse-Engineering Prozess aufgearbeitete Geometriemodell, als sogenannte *ungestörte Primärform* aufzufassen. Das Festlegen von Randbedingungen – z. B. aus der Domäne der Prozess- und Fügeverfahren – wird für den Fall, dass hierfür Änderungen an der Form vorgenommen werden müssen, als *Störung* der Primärgeometrie aufgefasst und eine solche Randbedingung fortan als *geometriewirksame Randbedingung* bezeichnet. Im Beispiel der Helikopterzelle aus Abbildung 3.3b ist es nicht möglich, die gesamte Zelle als Integralbauteil herzustellen. Aus diesem Grund muss die Zelle in Einzelteile zerlegt und diese wiederum zu einer Baugruppe zusammengefügt werden. Die gesamte Vorgehensweise dieses Umkonstruktionsprozesses ist in Abbildung 3.4a - d aufgezeigt.



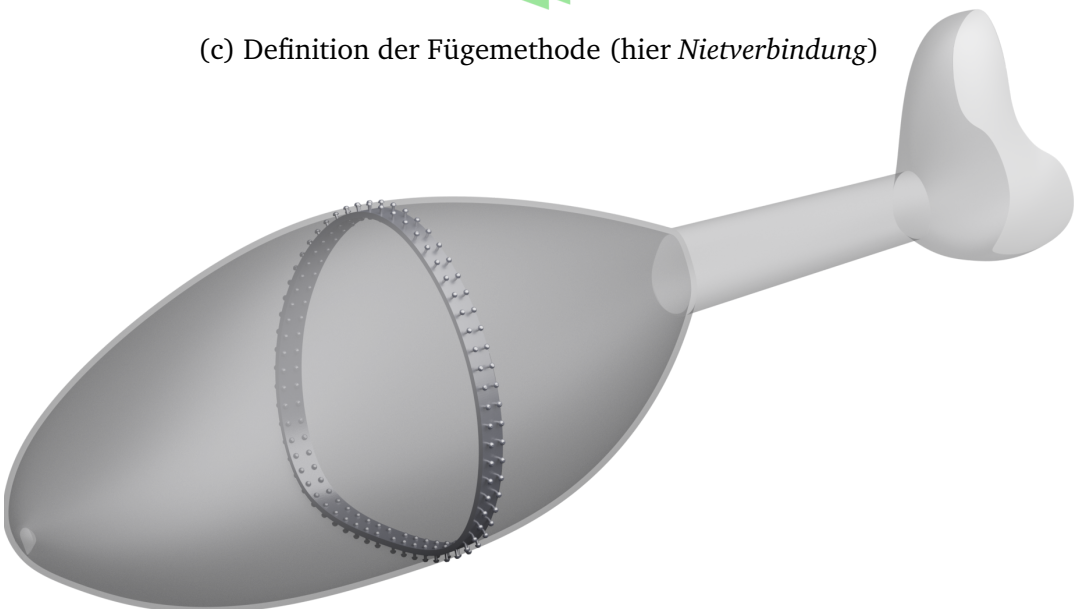
(a) Definition der Konstruktionszone („elektronisches Messer“)



(b) Definition der Stoßart (hier *Laschenstoß*)



(c) Definition der Fügemethode (hier *Nietverbindung*)



(d) Umkonstruiertes Geometriemodell

Abbildung 3.4: Automatisierte Umkonstruktion der Geometrie

Je nach anzuwendender Verbindungstechnik wird eine bestimmte Übergangsform – wie beispielsweise eine Überlappung – zwischen zwei benachbarten Bauteilen zwingend benötigt. In diesen im weiteren Verlauf als *Konstruktionszonen* bezeichneten Bereichen muss die Primärform automatisiert lokal verändert werden, um den gestellten Randbedingungen zu genügen und die globale Systemkonsistenz sicherzustellen. Zur Einbringung geometriewirksamer Randbedingungen liegt dem Entwurfsrahmenwerk die Vorstellung eines *elektronischen Messers* zugrunde, mit dessen Hilfe am Rand der Konstruktionszone jeweils ein digitaler Trennschnitt durch das Bauteil durchgeführt werden kann. Auf diese Weise lässt sich ein Segment aus der Primärgeometrie heraustrennen und durch ein passendes Drahtgittermodell ersetzen, das als Ausgangsbasis für

die weitere Umkonstruktion dient (vgl. Abbildung 3.4a). Das Drahtgittermodell leitet sich aus den Schnittkurven des elektronischen Messers mit dem Bauteil ab und kann durch das Setzen weiterer Schnitte innerhalb der Konstruktionszone nahezu beliebig weiter verfeinert werden.

Das elektronische Messer selbst ist in der Bildfolge 3.4a - d der Einfachheit halber als Ebenenschnitt durch das gesamte Bauteil dargestellt. Die Umsetzung ist im Allgemeinen aber nicht auf einen solchen beschränkt. Es sind ebenfalls nur Ausschnitte aus der Primärgeometrie sowie sonstige Schnitte mit beliebig komplexer Werkzeuggeometrie umsetzbar. Nach der Definition der Konstruktionszone wird anschließend definiert, in welcher Art die *Sekundärgeometrie* auszuführen ist (siehe Abbildung 3.4b). Die Sekundärgeometrien werden im Entwurfsrahmenwerk in einer generischen *Stoßdatenbank* vorgehalten. Dabei handelt es sich um geometrische Grundübergangsformen, wie sie z. B. nach [Houldcroft, 1990] oder [Wittel et al., 2019] üblicherweise ausgeführt werden. Die verschiedenen Formen der Sekundärgeometrien sind in Tabelle 3.4 aufgezählt. Der „Sonderstoß“ unter Punkt acht deutet an, dass es ebenfalls möglich ist, individuelle Übergangsformen in die Datenbank aufzunehmen. Wurde der Stoßtyp nicht als feste Randbedingung in das Entwurfsrahmenwerk eingebracht, wird an dieser Stelle automatisiert eine Variantenkonstruktion im Rahmen eines Re-Engineering-Prozesses initiiert, bei der alle mit dem aktuellen Entwurfskontext verträglichen Übergangsformen parallel weiterverfolgt werden. Das Auftrennen der Primärform – gegebenenfalls an verschiedenen Positionen – und die Einführung zusätzlich benötigter Übergänge ist als topologische Änderung der Primärform anzusehen. Es gilt dabei zu beachten, dass die Grundform nur innerhalb der Konstruktionszone verändert und Änderungen der Krümmung, Dicke und anderer Eigenschaften insgesamt minimal ausgeführt werden sollten. Auf die Primärform bezogene Parameter – wie z.B. eine vorgegebene Gesamtlänge oder ein Gesamtgewicht – dürfen durch die Sekundärgeometrie(n) nicht verändert oder überschritten werden.

Tabelle 3.4: Verschiedene Arten von Sekundärgeometrien

1. Stumpfstoß
2. Überlappungsstoß
3. Verzahnungsstoß (Joggle-Stoß)
4. Laschenstoß (Parallelstoß)
5. T-Stoß
6. Eckstoß
7. Schrägstoß
8. Sonderstoß

Als letzter Schritt muss eine Festlegung der Fügemethode beziehungsweise Verbindungstechnik getroffen werden (vgl. Abbildung 3.4c). Analog zum Stoßtyp hängt es auch an dieser Stelle von der Formulierung der Randbedingung ab, ob es zu einer weiteren Variantenkonstruktion durch Variation in der Fügetechnik kommt oder nicht. Da das Fügeverfahren potenziell Auswirkungen auf den Übergangstyp haben kann – und auch vice versa – ist eine starre Reihenfolge der in Abbildung 3.4 zur vereinfachten Erklärung chronologisch dargestellten Einzelschritte kein geeignetes Entwurfsmodell. Das Finden der passenden Entwurfsreihenfolge wird in Abschnitt 3.2.3 noch im Detail diskutiert. Zusammengefasst kann das elektronische Messer ganz allgemein zur Definition sogenannter Konstruktionszonen genutzt werden, die der Einbringung geometriewirksamer Randbedingungen in Form von Sekundärbauteilen, aber auch der Ausschnittdefinition (für z. B. für Fenster oder Türen) dienen kann. Auf diese Weise wurde auch der Rumpfspant, der in Abschnitt 5.3.1 als Anwendungsbeispiel gezeigt wird, definiert.

Material

Im Ingenieurentwurf spielt das für einzelne Bauteile oder das Gesamtprodukt zum Einsatz gebrachte Material eine wesentliche Rolle. Hierbei können unter anderem die Widerstandsfähigkeit gegenüber äußeren Lasten, wie beispielsweise Kräften, Momenten oder thermischen Lasten (siehe Abschnitt 3.2.2), aber auch die Funktionserfüllung, wie z. B. die elektrische oder magnetische Leitfähigkeit, die Herstellung und viele weitere Bereiche angeführt werden. Im Entwurfsrahmenwerk kann das Material sowohl eine – abhängig von der Modellierung instanz- oder feldbasierte (siehe Tabelle 3.2) – Anforderung als auch einen Funktionsträger darstellen.

Das Material – im ingenieurwissenschaftlichen Kontext auch häufig als Werkstoff bezeichnet – lässt sich allgemein in sogenannte Werkstoffhauptgruppen unterteilen, die sich aus Unterschieden in den atomaren Bindungskräften ableiten (siehe [Reuter, 2021, S. 109]). Auf oberster Abstraktionsebene hat sich eine Unterteilung, wie sie auch in Tabelle 3.5 dargestellt ist, etabliert (vgl. [Ashby, 2005, S. 29; Reuter, 2021, S. 109]).

Tabelle 3.5: Werkstoffhauptgruppen

1. metallische Werkstoffe
2. Kunststoffe
3. Keramiken
4. Gläser
5. hybride Werkstoffe
6. natürliche Werkstoffe

Aufgrund der Vielzahl bekannter und im Produktentwurf praktisch eingesetzter Materialien hat die Ausgestaltung der Materialontologie starke Auswirkungen auf die Applikation und resultierende Entwurfssequenz im Entwurfsrahmenwerk[23]. Eine klassische Modellierung stellt eine sich an den Werkstoffhauptgruppen orientierende Aufgliederung in verschiedene Materialklassen dar, die z. B. durch Anwendung der FCA (siehe Abschnitt 2.2.3) immer weiter ausspezifiziert werden, sodass sich eine hierarchische Ontologiestruktur großer Tiefe ergibt.

Die Blätter dieser Ontologie sind gemäß der in Abschnitt 3.2 getroffenen Definition den festdiskreten Typen zuzuordnen, sodass nur ein einziges Material bei Randbedingungsspezifikation auf Blattebene zulässig wäre. Hinsichtlich des Materials gäbe es dadurch keine Variationsvielfalt im Konstruktionsprozess mehr. Die Wahl einer Materialklasse aus einem Zwischenniveau als Randbedingung – beispielsweise aus den metallischen Werkstoffen – hat eine Variationsinstanziierung aller erbenden Unterklassen zur Folge. Die Festlegung auf ein Blatt der Ontologie kann durch Einführung eines Maximaltiefenparameters eingeschränkt werden. Bei unterschiedlicher Tiefe der verschiedenen Subontologien – beispielsweise angenommen, metallische Werkstoffe wären tiefer (und somit detaillierter) modelliert als keramische Werkstoffe – ergäben sich valide, aber in ihrer Ausdetaillierung bezüglich des Materials verschiedene Produkte.

Bei einer Materialexploration können tiefe Hierarchien zu einer erhöhten Anzahl an Iterationszyklen führen. Neben der hierarchischen Modellierung ist es daher auch möglich, die verschiedenen Materialien nach ihren Eigenschaften zu klassifizieren und die Modellierung zu gestalten. Dies wird aufgrund der höheren Flexibilität gegenüber hierarchischen Ontologien in vielen Ansätzen zur Materialmodellierung verfolgt[24]. In diesem Zusammenhang sei darauf hingewiesen, dass der gewählte Ontologiemodellierungsansatz prinzipiell völlig unabhängig

[23] Eine detaillierte Darstellung dieses Sachverhalts folgt in Abschnitt 3.2.3.
[24] Siehe hierzu z. B. [de Baas, 2017], [Ashino, 2010] oder [Huanyu Li et al., 2020].

vom theoretischen Konstrukt und der Funktionsweise des Entwurfsrahmenwerks ist und bei äquivalenter Modellierungsmächtigkeit dieselben Produktvarianten mit unterschiedlichen Ontologien erzeugt werden können. Es ergeben sich ausschließlich anwendungsbezogene und implementatorische Unterschiede, die an dieser Stelle nicht weiter diskutiert werden sollen.

Herstellungs- und Verbindungstechnik

Hinter der Domäne der Herstellungs- und Verbindungstechnik, die in Abbildung 3.2b abstrakt mit dem Begriff Herstellung bezeichnet wurde, werden im Rahmen dieser Arbeit einerseits das Herstellungs- bzw. Fertigungsverfahren wie beispielsweise Tiefziehen, Drehen oder Fräsen, andererseits die bei aus Einzelteilen oder Bauteilen assemblierten Zusammenbauten benötigten Verbindungstechniken und Fügeverfahren wie z. B. Schrauben, Schweißen oder Nieten verstanden. Die Herstellung kann in diesem Sinne im Rahmen des hier vorgestellten RTE-Frameworks eine Anforderung (vgl. Tabelle 3.2) oder eine Funktion (siehe Abschnitt 3.2.2) darstellen, die primär der Fertigung eines Bauteils oder der Verbindung zweier oder mehrerer Bauteile als Zweck dient. In der DIN-Norm 8580 werden die Hauptfunktionen Schaffen der Form, Ändern der Form und Ändern der Stoffeigenschaften genannt [DIN 8580, S. 7]. Diese werden in der Folge in die Unterfunktionen Zusammenhalt schaffen, Zusammenhalt beibehalten, Zusammenhalt vermindern und Zusammenhalt vermehren untergliedert. Diese funktionale Dekomposition der Fertigungsdomäne motiviert die Einführung der sechs Hauptgruppen der Fertigungsverfahren (siehe [DIN 8580, S. 7]). Die Hauptgruppen sind in Tabelle 3.6 aufgezählt und dienen später als Basisklassen für die Ontologiemodellierung. Die Tatsache, dass es sich bei allen Begriffen um substantivierte Verben handelt, unterstreicht den primären Charakter dieser Domäne als Funktion im Produktentwurf. Die Hauptgruppen werden in [DIN 8580] darüber hinaus in zahlreiche Untergruppen kategorisiert, welche ebenfalls als Vorlage für die Ausdetaillierung der Ontologie verwendet werden können. In dieser Arbeit wird exemplarisch die Hauptgruppe des Fügens betrachtet (siehe Abschnitt 4.2.6) und später in der Anwendung (siehe Kapitel 5) demonstriert. Die restlichen Fertigungsverfahren können aber völlig analog dazu Eingang in das Entwurfsrahmenwerk finden. Was die Einteilung der Verbindungen anbetrifft, kann diese nach den Merkmalen „starr, gelenkig oder elastisch“, „lösbar oder unlösbar“ oder „stoffschlüssig, formschlüssig oder kraftschlüssig“ erfolgen [Ehrlenspiel und Meerkamm, 2013, S. 468].

Tabelle 3.6: Hauptgruppen der Fertigungsverfahren

1. Urformen
2. Umformen
3. Trennen
4. Fügen
5. Beschichten
6. Stoffeigenschaften ändern

Bei der Modellierung muss eine dieser Unterteilungen zugrunde gelegt werden. Wie bereits bei der Werkstoffmodellierung erwähnt, sei hier nochmals angemerkt, dass die Festlegung auf eine Modellierungsart für die Generierungsergebnisse des Frameworks keine Rolle spielt, solange sie von äquivalenter Generizität und Flexibilität ist. Die Auswirkungen der Modellierung beziehen sich auf laufzeittechnische Aspekte – hier insbesondere auf die benötigte Anzahl an Iterationszyklen (siehe Abschnitt 4.4) – und die Benutzerfreundlichkeit bei der Anwendung.

Last

Unter der Domäne Last werden alle möglichen inneren und äußeren Lasten subsumiert, die einen Einfluss auf ein System haben können oder infolge derer eine Systemreaktion auftreten kann. Dabei kann es sich um mechanische Lastgrößen wie Kräfte oder Momente, aber auch um thermodynamische Lasten wie Wärmeeintrag oder Druck, elektrische oder magnetische Lasten, chemische Lasten und viele weitere handeln. Die gängigsten Kategorien aus der Lastdomäne sind in Tabelle 3.7 zusammengefasst. Bei mechanischen Lasten müssen für eine mögliche Antwort des Systems entsprechende Lagerdefinitionen wie z. B. eine feste Einspannung festgelegt sein. Je nach Art der Krafteinleitung werden verschiedene Beanspruchungsarten – beispielsweise Pressung, Druck, Zug, Biegung oder Torsion – unterschieden [Ehrlenspiel und Meerkamm, 2013, S. 504]. Innerhalb des RTE-Frameworks handelt es sich bei der Last um eine reine Anforderung beziehungsweise Randbedingung. Aus modellierungstechnischer Sicht müssen alle genannten Lastkategorien und die zusätzlich benötigten Lagerdefinitionen in der Lastdomäne abgebildet werden. Da es sich bei der Last in aller Regel um eine symbolische und nicht um eine diskrete Variable handelt (vgl. Tabelle 3.2) – wie beispielsweise beim gewählten Materialmodell –, ist es nicht zielführend, ihr bei Nichtspezifikation durch Variation in Form eines Re-Engineerings zu begegnen. Sinnvoller ist in diesem Zusammenhang, von einer Nachrechenaufgabe bei spezifizierter Last in Form einer feldbasierten Randbedingung mit der dahinterstehenden Fragestellung: „Hält die Konstruktion der Last stand?“ zu einer allgemeinen Lastberechnung – wie z. B. der maximal ertragbaren Last oder der Berechnung von Eigenfrequenzen – überzugehen und diese als Variantenbewertungsgröße im Konstruktionsprozess zu berücksichtigen. Dadurch werden die Werte der entsprechenden Instanzvariablen berechnet, ohne dass diese in den Anforderungen explizit gefordert wurden.

Tabelle 3.7: Lastkategorien

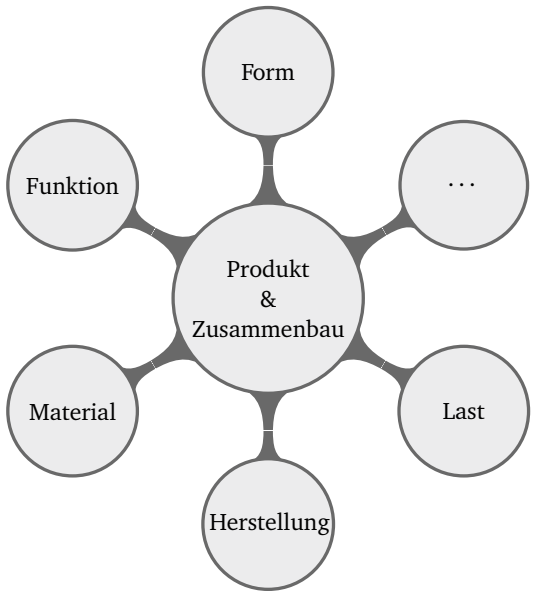
1. mechanische Lasten
2. thermodynamische Lasten
3. elektrische Lasten
4. magnetische Lasten
5. elektrische Lasten
6. chemische Lasten

3.2.3 Modellierung der Entwurfsabfolge im automatisierten Entwurf

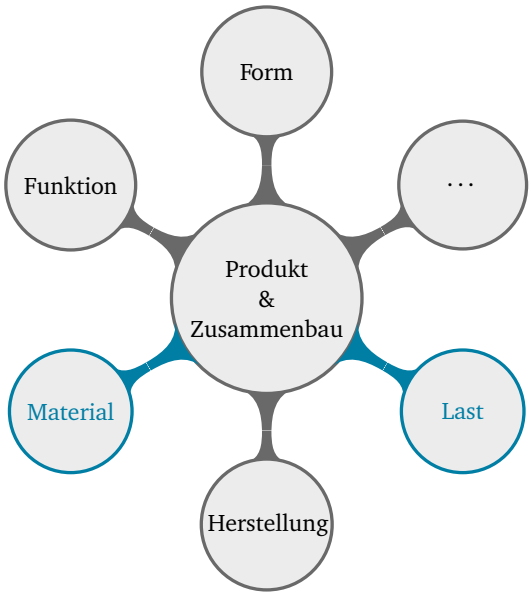
Der Konstruktionsprozess ist ein Vorgang, der mit einer abstrakten Beschreibung beginnt und in einem konkreten Produkt als Resultat aus vielen Einzelentscheidungen endet (siehe Abschnitt 2.2.1). In einem automatisierten Entwicklungs- und Konstruktionsframework gilt es, diese hochgradig problemspezifischen Einzelentscheidungen zu modellieren und abzubilden. Wie im vorangegangenen Abschnitt aufgezeigt wurde, hat eine Konstruktionsaufgabe im Ingenieurentwurf im Allgemeinen eine Vielzahl an Randbedingungen, Anforderungen, Funktionen und Zwängen aus unterschiedlichen Domänen, die oftmals voneinander abhängen und sich dadurch gegenseitig beeinflussen. Unterschiedliche Randbedingungen führen dabei ganz natürlich auf alternative Produkte mit differierenden Teilkomponenten. In klassischen Entwurfsmethodiken wird auf einer Metaebene des Entwurfs eine starre Abfolge von Konkretisierungsschritten unterstellt, entlang derer die Konstruktion verfeinert wird (z. B. klassisches PAHL/BEITZ-Schema

Abbildung 2.8). Die tatsächliche Entscheidungsabfolge auf der Mikroebene wird dabei weg abstrahiert bzw. gar nicht aufgelöst und somit dem/der Ingenieur/-in bei der Lösungsfindung in einem hochgradig iterativen, manuellen Prozess überlassen [Glock, 1997, S. 21].

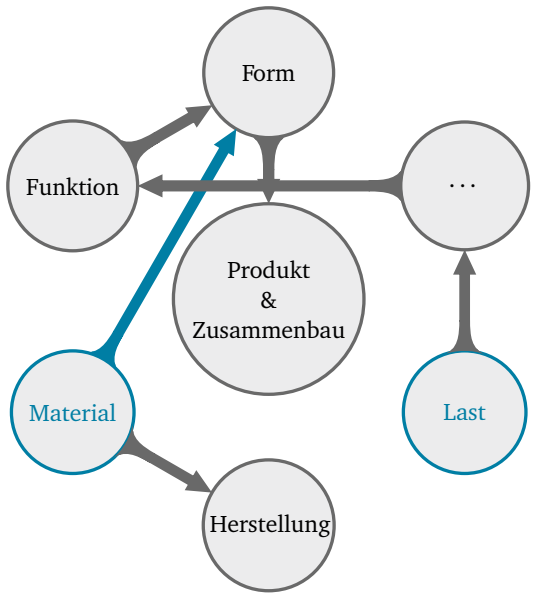
Dieser Vorgang kann ohne Beschränkung der Allgemeinheit mit einem starren Abfolgemodell nicht hinreichend modelliert werden. Die Entscheidungsreihenfolge muss vielmehr dynamisch von den definierten Randbedingungen im situativen Kontext der jeweiligen Konstruktionsaufgabe abhängen. Ausschlaggebend ist dabei, welche Anforderungen als *konstant* (gegeben) und welche als *variabel* (gesucht) betrachtet werden müssen. In Abbildung 3.5 ist die Auswirkung spezifizierter Randbedingungen auf die Entwurfsabfolge vereinfacht illustriert.



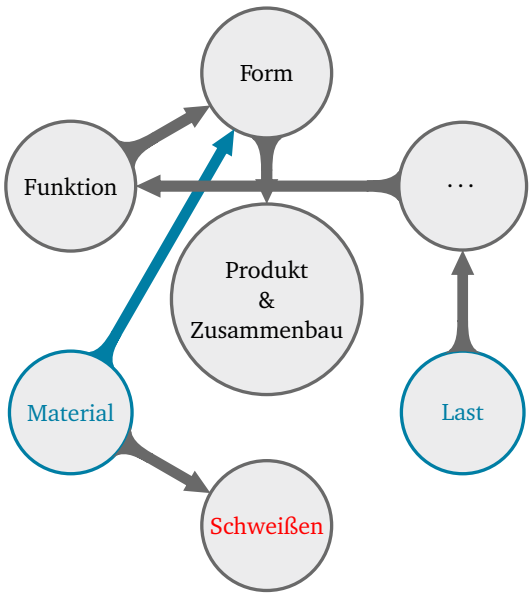
(a) Unspezifiziertes Problem



(b) Spezifikation von Randbedingungen



(c) Aufbau des gerichteten Graphen



(d) Auskonstruktion durch Spezifikationsvariation

Abbildung 3.5: Ableitung der Entwurfsabfolge im Entwurfsrahmenwerk (schematisch)

In der Philosophie des RTE sind als Ausgangsbasis einer jeden Konstruktionsaufgabe zunächst alle Anforderungen, Randbedingungen und Funktionen noch unbestimmt und damit für den Interpreter des Entwurfsrahmenwerks gleichberechtigt (siehe Abbildung 3.5a). Werden nun Konkretisierungen oder Spezifizierungen in einigen Domänen vorgenommen – beispielsweise der Werkstoff und die zu ertragenden physikalischen Lasten als gegeben spezifiziert – müssen diese Vorgaben im automatisierten Konstruktionsvorgang berücksichtigt werden, um ein in diesem Sinne valides Produkt hervorbringen. Die Spezifikation der Randbedingungen ist in Abbildung 3.5b durch die farbliche Hervorhebung symbolisiert. Material und Last sind somit für das Entwurfsrahmenwerk fortan nicht mehr als variable Randbedingungen zu betrachten, sondern in variabel-diskrete Randbedingungen (vgl. Tabelle 3.2)[25] übergegangen.

Eine Randbedingung kann theoretisch zunächst überall in der Entwurfsabfolge berücksichtigt werden. Tatsächlich gibt es aber implizite Abhängigkeiten, die sich im implementierten Entwurfsmodell aus der Formulierung der Domänenontologien und deren Interdependenzen sowie der sonstigen festen Randbedingungen und spezifizierten Funktionen im aktuellen Konstruktionskontext ergeben. Die Interdependenzen des zugrunde gelegten Ontologiemodells sind ein Resultat der „korrekten“ Modellierung der zwischen den Domänen real vorhandenen Kopplungen. Auch wenn die Modellbildung im Grundsatz nicht eindeutig bestimmt ist, wird im weiteren Verlauf eine in diesem Sinne korrekte Modellierung der Domänenzusammenhänge als Grundvoraussetzung für die fehlerfreie Funktionalität des Frameworks vorausgesetzt. Dies liegt in der Tatsache begründet, dass ein falsches Modell der Realität im Entwurfsrahmenwerk in der Konsequenz nicht – oder nur zufällig – auf ein valides Produkt führen kann.

Die Anwendung formalisierter Verfahren zur Ableitung der Ontologien, wie z. B. der FCA (siehe Abschnitt 2.2.3), kann bei der Modellierung helfen, die Korrektheit sicherzustellen. Ein Teil der beschriebenen impliziten Zusammenhänge kann aus den statischen Ontologien abgeleitet und durch die Grundreihenfolge, in der die verschiedenen Domänen innerhalb des Frameworks verarbeitet werden, berücksichtigt werden. Zur Bestimmung dieser Grundreihenfolge müssen die zugrunde liegenden Ontologien als ein gemeinsamer Graph aufgefasst werden, der durch die Verwendung gleicher Begriffe und Eigenschaften in unterschiedlichen Domänen entsteht. Ein weiterer Teil der Graphenstruktur ergibt sich aus den Generalisierungen und Assoziationen innerhalb der Domänen. Auf diesem Ontologiegraphen kann nun der Graphenalgorithmus des topologischen Sortierens[26] oder vergleichbare Verfahren angewendet werden und für Teilbereiche, die eine partielle Ordnung[27] bilden, eine sinnvolle Reihenfolge abgeleitet werden.

Bei der Grundreihenfolgebestimmung wird die Gleichberechtigung der Randbedingungen aus allen Domänen vorausgesetzt und eine „natürliche“ Prozessreihenfolge, unabhängig von gesetzten Randbedingungen abgeleitet. Durch die Iterationszyklen des RTE-Kreislaufs ist die Grundreihenfolge zwar theoretisch beliebig modellierbar, eine geschickte Wahl dieser Sequenz kann aber die Anzahl benötigter Iterationszyklen über alle Domänen hinweg verringern. Über die auf statischen Überlegungen basierende Grundreihenfolge hinaus müssen die spezifizierten

[25] In diesem Beispiel soll angenommen werden, dass die Randbedingungen Material und Last nicht auf Blattebene der Ontologie definiert wurden und somit keine fest-diskreten Randbedingungen darstellen.

[26] siehe z. B. [Saake und Sattler, 2021, S. 490–492]

[27] siehe z. B. [Kemper und Reimers, 2022, S. 18]

gegebenen Größen aus dem aktuellen Entwurfskontext in der Entwurfssequenz dynamisch berücksichtigt werden. Stellt man sich eine Repräsentation des aktuellen Entwurfszustands als zunächst ungerichteten Graphen vor, so entsteht durch das Festlegen von Randbedingungen in den zu berücksichtigenden Teildomänen daraus ein in Teilen gerichteter Graph. Dies ist in Abbildung 3.5 durch die Änderung der Domänenverbindungen von ungerichteten Linien (siehe Abbildung 3.5b) zu gerichteten Pfeilen (siehe Abbildung 3.5c) impliziert. Aus den gerichteten Abhängigkeiten der zur Laufzeit vorhandenen Objekte ergibt sich eine Richtungsabhängigkeit des Graphen der verschiedenen Domänen, die aber nur im Kontext der jeweils aktuell spezifizierten Randbedingungen gültig ist[28]. In diesem gerichteten Graphen muss nun ein Weg gefunden werden, um von den festgelegten variabel-diskreten (*Zwischenontologieebenendefinitionen*) beziehungsweise fest-diskreten Randbedingungen (*Blattebenendefinitionen*) auf die noch freien Entwurfsvariablen zu schließen, diese anschließend verträglich zu instanziieren und dadurch in den Entwurfskontext zusätzlich einzubringen. Die Richtungsabhängigkeit führt auf die gesuchte Abfolge der verschiedenen Domäneninstanziierungen und -konkretisierungen („Entwurfsabfolge“). Da eine im zeitlichen Kontext später vorgenommene Spezifizierung potenziell auch Auswirkungen auf frühere Instanziierungen und Konkretisierungen haben kann, muss dieser Vorgang in einem mehrfach iterativen Prozess wiederholt durchlaufen werden.

An dieser Stelle sei angemerkt, dass für die Entwurfsabfolge im Allgemeinen keine eineindeutige Lösung existiert. Dies hängt damit zusammen, dass in voller Allgemeinheit keine bijektiven Zusammenhänge zwischen den Domänen existieren[29]. Wichtig ist in diesem Zusammenhang zunächst nur, dass überhaupt eine gültige Konstruktionssequenz gefunden werden kann. Die grundsätzliche Existenz mindestens eines Lösungswegs wird für ein wohldefiniertes Problem an dieser Stelle als These postuliert und bildet die Grundvoraussetzung für ein automatisiertes RTE. Auch wenn ein direkter Beweis dieser These unmöglich zu erbringen ist, kann die Umkehrung dieser Annahme Aufschluss über die Gutmütigkeit des Problems bieten: Wäre die Annahme falsch, wäre es prinzipiell unmöglich, überhaupt ein valides Produkt als Ergebnis eines Konstruktionsvorgangs erhalten zu können. Die Realität zeigt hier aber aus der Existenz von Abermillionen entworfener und korrekt arbeitender Produkte heraus, dass die Kontraposition falsch sein muss und plausibilisiert damit die Gültigkeit der Grundthese. Trotz dieser Erkenntnis ist es selbstverständlich weiterhin möglich, dass durch eine nicht ausreichende Anzahl einschränkender Randbedingungen in endlicher Zeit kein Lösungsweg gefunden werden kann oder die Lösung nach zahlreichen Iterationen nicht konvergiert. Der Konstruktionsvorgang muss folglich im Entwurfsrahmenwerk mit einer Fehlermeldung abgebrochen werden. Im Falle einer Schar an Lösungssequenzen wird der Einfachheit halber der erste gefundene Lösungspfad ausgewählt. Die Auswirkungen alternativer Sequenzen auf das Produkt sollen an dieser Stelle nicht weiter betrachtet und stattdessen diesbezüglich auf Kapitel 5 verwiesen werden[30].

[28] Diese kann insbesondere der statisch bestimmten „natürlichen“ Reihenfolge bei gleichberechtigten Randbedingungen widersprechen.

[29] Aufgrund der nicht vorhandenen Bijektivität ist die Abbildung nicht umkehrbar bzw. deren Existenz nicht gesichert. Daher wird bei der Modellierung mit Entwurfssprachen die Methodik der „systematischen Vorwärtsvariation“ [Rudolph, 2002, S. 103]) angewendet

[30] Bemüht man wieder die mathematische Analogie eines nicht linearen Gleichungssystems als Entwurfsmodell, so kann aus deklarativer Sicht gesagt werden, dass bei Übereinstimmung der Anzahl an Randbedingungen und

In der Vorgehensweise nach Abbildung 3.5 werden als Letztes noch die verbleibenden freien Variablen instanziiert oder noch abstrakt vorgehaltene Instanzen durch Spezifikationscasts konkretisiert. Durch diesen Vorgang wird das Produktmodell sukzessive detailliert und mit neuen Informationen angereichert. Während beim Forward-Engineering der Fokus auf einer einzigen Variante liegt, werden beim Re-Engineering an dieser Stelle alle verträglichen Varianten aus den Domänen heraus gebildet. Hierdurch entstehen somit die gesuchten validen Produkte, die die Menge an Randbedingungen erfüllen. Dies ist in Abbildung 3.5 durch das Ändern des abstrakten Begriffs „Herstellung“ (siehe Abbildung 3.5c) auf „Schweißen“ (siehe Abbildung 3.5d) angedeutet. Allgemein werden bei diesem Vorgang verschiedene Hierarchieebenen in den beschreibenden Ontologien überstrichen. Die vom Entwurfsrahmenwerk gewählte Variationstiefe ist dabei immer maximal und liegt somit entweder auf Blattebene der zugehörigen Domänenontologie oder auf dem Wert der relativ angegebenen Maximaltiefenvariablen[31]. Hintergrund dieser Vorgehensweise ist, dass durch das Framework immer maximal konkret konstruiert werden soll und dadurch Objektinstanziierungen aus Zwischenontologieebenen vermieden werden. Ausgenommen davon sind Fälle, in denen dies explizit durch die Angabe der angesprochenen Maximaltiefenvariablen von dem/der Benutzer/-in gefordert wird.

Die Existenz, der Ursprung und eine mögliche Ableitung einer zu einem Entwurfsprozess gehörenden dynamischen Entwurfssequenz wurde in diesem Abschnitt plausibilisiert und anhand eines sehr abstrakt gehaltenen Beispiels illustriert. Die Variabilität der Lösungssequenz stellt ein wesentliches Differenzierungsmerkmal zu anderen Ansichten und Arbeiten in der Literatur dar, da dort wie bereits angesprochen auf einer festen Entwurfsabfolge aufgesetzt wird. Eine konkrete produktbezogene Umsetzung der hier rein abstrakt beschriebenen Ableitung der dynamischen Entwurfssequenz ist in Abschnitt 4.4 dargestellt. Dabei handelt es sich um eine im Rahmen dieser Arbeit umgesetzte, prototypische Implementierung zur Prinzipdemonstration.

der Freiheitsgrade potenziell Lösungswege gefunden werden können. Dabei hängt jedoch trotz algorithmisch erzeugbarer Lösungssequenzen die Lösbarkeit von der Verträglichkeit der Gleichungen untereinander ab.

[31] Sollte die Maximaltiefenvariable die Blattebene überschreiten, wird stattdessen die Blattebene instanziiert.

4 Implementierung eines Frameworks für Round-Trip-Engineering

> *Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt.“*
>
> **— Ludwig Wittgenstein**
> (österreichischer Philosoph)

In diesem Kapitel wird eine prototypische Implementierung der im vorherigen Kapitel deskriptiv beschriebenen Systematik des Round-Trip-Engineering im modellbasierten Ingenieurentwurf vorgestellt. Die Grundlage für die Implementierung bilden die in Abschnitt 2.2.4 beschriebenen graphenbasierten Entwurfssprachen (GBES). Das Kapitel beginnt mit einer Übersicht über den Einsatz der GBES im Gesamtkontext des Frameworks. Danach schließt sich die Vorstellung der Ontologiemodellierung der in Abschnitt 3.2.2 beschriebenen Entwurfsdomänen in Form von Klassendiagrammen an. Es folgt eine Beschreibung der Umsetzung der inkrementellen Konstruktionsdetaillierung für das Forward- und Re-Engineering und die Konstruktionsabstraktion für das Reverse-Engineering durch die Formulierung generischer Regeln. Zuletzt wird die Umsetzung der dynamischen Entwurfssequenz dargelegt, wobei ein innovativer Ansatz verfolgt wird, der in dieser Form noch in keiner anderen Entwurfssprachenanwendung in der Arbeitsgruppe oder in der Literatur zum Einsatz kam und daher neu ist.

Kapitelübersicht

4.1 Round-Trip-Engineering mit graphenbasierten Entwurfssprachen

In diesem Kapitel wird dargelegt, wie graphenbasierte Entwurfssprachen (GBES) und die Entwicklungsumgebung DC43 (siehe Abschnitt 2.2.4) als Basis zur Implementierung eines Frameworks für Round-Trip-Engineering (RTE) in dieser Arbeit genutzt werden.

Die Fähigkeiten der GBES als Werkzeug zur Entwurfsautomatisierung im Allgemeinen konnte bereits anhand zahlreicher Beispiele demonstriert werden. Insbesondere das in Abschnitt 3.1.1 beschriebene Forward-Engineering stellt den „klassischen“ Anwendungsbereich der GBES dar, da in diesem Fall der generierende Aspekt einer Initiallösung im Vordergrund steht. Diese vorwärts gerichtete Entwurfssynthese wurde im Rahmen abgeschlossener Dissertationen an der Fakultät Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart unter anderem anhand von Satelliten [Groß, 2014], Flugzeugkabinen [Motzer, 2015], SCR-Systemen [Vogel, 2016] und Raumstationen [Fink, 2017] demonstriert. Reverse- und Re-Engineering als weitere Bausteine des in Kapitel 3 eingeführten RTE standen bisher nicht im Fokus der Betrachtungen.

Der hochdynamische Ablauf des RTE ist in der klassischen Darstellung der Informationsarchitektur der GBES (siehe Abbildung 2.14) nicht vollständig darstellbar. Ungeachtet dessen soll anhand der daran angelehnten Abbildung 4.1 ein erster Überblick über die Rolle der GBES für die Implementierung des Frameworks und dessen Prozesse aufgezeigt werden.

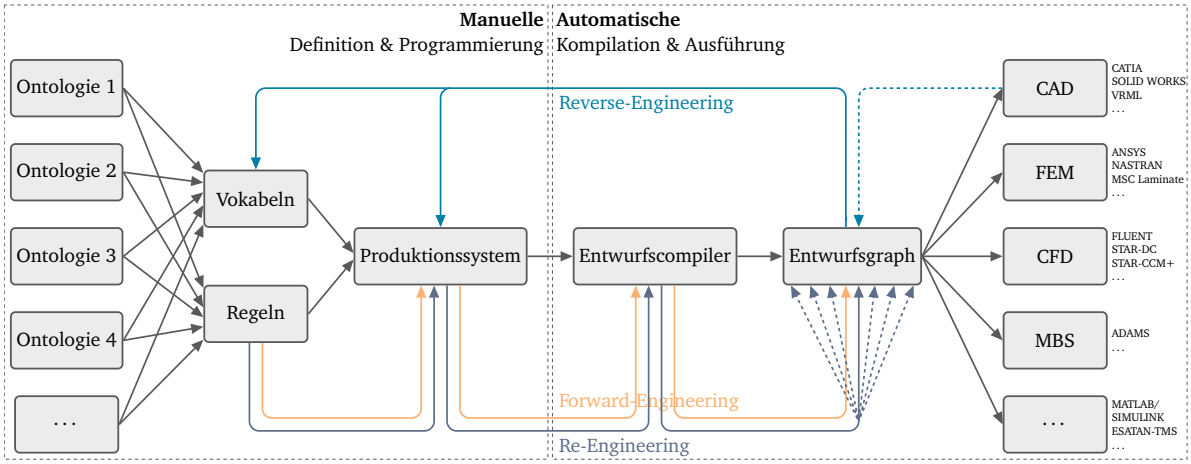


Abbildung 4.1: Graphenbasierte Entwurfssprachen als Framework für Round-Trip-Engineering, angelehnt an Abbildung 2.14

Wie in Abschnitt 2.2.4 bereits dargelegt, basieren GBES auf einem definierten Satz an Vokabeln, Regeln und einem Produktionssystem, dessen Anweisungen im Entwurfscompiler in den Entwurfsgraphen als zentralem Datenmodell übersetzt werden. Die im Entwurfsgraphen in abstrakter Form vorgehaltenen Informationen und Daten können anschließend über Schnittstellen in die verschiedenen Analyse- und Simulationsprogramme exportiert werden.

Der Entwurfsgraph wird in den meisten Fällen über die Sequenz der Regeln hinweg inkrementell mit Wissen angereichert. Durch die lokalen Modelltransformationsvorschriften in den Regeln kann der vorwärts gerichtete Entwurfsprozess detailliert abgebildet werden. Voraus-

setzung hierfür ist die Definition der benötigten Vokabeln in suffizienter Granularität für die betrachtete Entwurfsaufgabe. Die Philosophie des RTE unterscheidet drei Entwurfsvorgänge, die in unterschiedlichen Entwurfskontexten zutage treten und abgebildet werden müssen.

Das Forward-Engineering wird durch einen vollständigen Durchlauf der Entwurfssprache bis zum finalen Entwurfsgraphen repräsentiert (siehe orange Pfeile in Abbildung 4.1). Die initial anzulegenden Randbedingungen, Anforderungen und Funktionen können genauso wie die im weiteren Verlauf des Entwurfs benötigte inkrementelle Anreicherung des Modells (Entwurfsgraph) mit Wissen in Form von Regeln formuliert werden. Die Abfolge der Regeln ist durch das Aktivitätsdiagramm eindeutig nach vorne gerichtet und kann als Umsetzung einer statischen Entwurfssequenz gedeutet werden, entlang derer das Produkt konkretisiert wird.

Das Reverse-Engineering ist ein Prozess, der zwangsläufig auf einem vorhandenen Modell – im einfacheren Fall in Form eines Entwurfsgraphen oder komplexer in der Umsetzung in Form eines domänenspezifischen Programmmodells – aufsetzen muss. Das Ziel dieses Prozesses ist, das vorhandene Modell zu analysieren, die vorhandenen Elemente und deren Zusammenhänge zu interpretieren und infolgedessen vor dem Hintergrund eines als Klassendiagramm modellierten Weltmodells in einen breiteren Zusammenhang zu generalisieren.

Auf dieser Grundlage ist es möglich, alternative Konstruktionsansätze zu eruieren, die mit den im aktuellen Entwurfskontext auftretenden Randbedingungen und sonstigen Zwängen (Constraints) verträglich sind. Diese Alternativenbildung kann auf verschiedenen Abstraktionsniveaus ansetzen, die während des Entwurfs überstrichen werden. Abhängig von der gewählten Implementierung kann sich dies in der Verwendung alternativer Komponenten bis hin zu einem Wechsel der Prinziplösung äußern. Läuft der vorwärts gerichtete Entwurf beispielsweise in eine „Sackgasse“, in der der Graph nicht mehr weiter gegen eine Lösung konvergiert, kann ein solcher Mechanismus helfen, die Verfolgung eines alternativen Konstruktionspfads zu initiieren.

Für die Implementierung eines solchen Vorgangs eigenen sich ebenfalls Regeln einer Entwurfssprache, die dergestalt formuliert sein müssen, dass sie auf Basis des dem aktuellen Entwurfsartefakt zugehörigen Klassendiagramms eine Abstraktion der Konstruktion bewirken. Die hierfür benötigte Technik wurde in Abschnitt 3.1.3 als Abstraktionscast bezeichnet und wird im weiteren Verlauf dieses Kapitels in Abschnitt 4.4 noch genauer beschrieben.

In Abbildung 4.1 ist der Vorgang des Reverse-Engineering mit blauen Pfeilen hervorgehoben. Im einfacheren Fall des Aufsetzens auf einem Entwurfsgraphen kann dieser direkt interpretiert und durch die Ausführung spezieller, revers formulierter Regeln im Produktionssystem den Vorgang in Gang setzen. Der zweite durchgezogene blaue Pfeil, der zurück zum Vokabular reicht, symbolisiert das Zusammenspiel des Reverse-Engineering mit dem zugrunde liegenden Klassendiagramm. Die Interpretation erfolgt wie bereits erwähnt immer vor dem Hintergrund eines Domänenweltmodells. Nur auf dieser Grundlage ist ein Abstraktionscast in der Praxis zu implementieren. Handelt es sich beim Einstieg in das Reverse-Engineering um ein konkretes Programmmodell wie beispielsweise ein CAD-Modell, muss über einen Mustererkennungsmechanismus (vgl. Abschnitt 3.1.2) zunächst eine äquivalente Graphenrepräsentation aus dem Domänenmodell erschlossen werden (gestrichelter blauer Pfeil). Auch hierbei spielt das Klassendiagramm als Referenz wieder eine entscheidende Rolle. Die nachgelagerten Schritte

entsprechen anschließend exakt dem bereits beschriebenen Vorgehen bei gegebenem Graphen.

Das Re-Engineering wird beim Produktentwurf im Rahmen des RTE-Schemas immer dann benötigt, wenn ein *unterbestimmter* Entwurfskontext vorliegt. In diesem Zusammenhang sei noch einmal das Analogon des mathematischen Gleichungssystems (MGS) bemüht. Zur Lösung eines unterbestimmten MGS müssen Annahmen getroffen und teilweise Werte geraten werden, die am Ende auf Verträglichkeit mit den Gleichungen geprüft werden. Analog dazu ist die Variantenbildung im Falle eines unterbestimmten Entwurfskontexts zu verstehen. Konkret auf die Implementierung bezogen sind zum betrachteten Zeitpunkt, zu dem ein Re-Engineering möglich ist, Entwurfsartefakte im Sinne instanziierter Ontologieklassen vorhanden, deren Typ noch weitere Subtypen auf Ontologieebene besitzen. Diesem Fall wird im Entwurfsrahmenwerk durch einen „generierenden Spezifikationscast“ begegnet, bei dem eine generische Klasse auf all ihre Subtypen gecastet wird und dabei jeweils eine neue Variante gebildet wird. Somit werden die vorhandenen Freiheitsgrade des Entwurfskontexts genutzt, um den Entwurfsraum zu explorieren und eine konkret angepasste Lösung ableiten zu können.

In Abbildung 4.1 symbolisieren die dunkelblauen Pfeile den Vorgang des Re-Engineerings. Durch die Auffächerung im hinteren Teil soll die Variantenbildung angedeutet werden. Das Re-Engineering zählt wie das Forward-Engineering zu den konkretisierenden Vorgängen, wodurch im Umkehrschluss keine Abstraktionscasts in den Regeln auftreten dürfen. Durch die Zuordnung einzelner Regeln zu sogenannten Variationsblöcken können mehrere Varianten eines zu entwerfenden Produkts im Rahmen des Re-Engineerings abgebildet werden (vgl. Abschnitt 2.2.4). Jede Variante wird durch einen eigenen Entwurfsgraphen repräsentiert, wobei die Varianten anschließend anhand einer geeigneten Metrik bewertet und gegeneinander abgewogen werden können, um im situativen Kontext die beste Lösung zu finden.

Ein weiterer Vorgang, der eine Mischform aus Forward-Engineering und Re-Engineering darstellt, ist die gezielte, manuelle Änderung des Entwurfskontexts durch Löschen oder Hinzufügen von Randbedingungen, Anforderungen oder Funktionen. Eine solche Modifikation kann im Rahmen eines neu initiierten Forward-Engineering ebenfalls auf eine Produktvariante führen.

In den folgenden Abschnitten werden alle für die Implementierung benötigten Modellierungsbausteine wie Klassendiagramme, Aktivitäts- und Regelmechanismen anwendungsbezogen und ergebnisorientiert vorgestellt. Zuletzt wird detailliert beleuchtet, wie die Regelsequenz dynamisch vor dem Hintergrund des lokalen Entwurfskontextes abgeleitet werden und der Kreisprozess des RTE in GBES implementiert werden kann. Es sei an dieser Stelle angemerkt, dass der Umfang und Detaillierungsgrad der Elemente keinen Anspruch auf Vollständigkeit erhebt. Vielmehr soll eine genaue Vorstellung vermittelt werden, wie das Konzept des RTE implementiert und angewendet werden kann. Die ungeachtet dessen vorhandene Praxisrelevanz belegen die in Kapitel 5 vorgestellten Beispiele als Resultat der hier gezeigten Implementierung.

4.2 Ontologiedefinition als Klassendiagramm

Klassendiagramme stellen ein essenzielles Modellierungselement graphenbasierter Entwurfssprachen dar (siehe Abschnitt 2.2.4). In Abschnitt 3.2.2 wurden die für das RTE wichtigen

Domänen hergeleitet und beschrieben. In diesem Abschnitt soll nun die Modellierung der zugehörigen Ontologien in Form von Klassendiagrammen vorgestellt werden. Die Klassendiagramme sind zur besseren Übersichtlichkeit nicht als Screenshot der Originalklassendiagramme aus DC43 visualisiert. Sie sollen die Struktur und Zusammenhänge verdeutlichen und enthalten aus diesem Grund auch nicht alle im Original vorhandenen Konstruktoren, Methoden und Eigenschaften, sondern nur solche, die für das Verständnis unabdingbar sind. Darüber hinaus sind sehr detailreiche, tiefere Ebenen der Klassendiagramme teilweise ebenfalls nicht abgebildet, da sie den sinnvollen Umfang in einem Textdokument sprengen.

4.2.1 Metaebene

Die Abhängigkeiten der relevanten Domänen untereinander werden in der vorliegenden Implementierung über ein extra Klassendiagramm abgebildet. Der Vorteil bei diesem Vorgehen ist, dass so ein gemeinsamer Vokabelvorrat auf einer Metaebene definiert werden kann, der von allen Domänenklassendiagrammen als Schnittstelle für den In- und Output verwendet werden kann und dadurch die semantische Interoperabilität sichergestellt und, wie bereits in Abschnitt 3.2.1 erwähnt, die Anzahl der Schnittstellen der Domänen untereinander reduziert werden kann. Die Detailmodellierung der jeweiligen Domäne wird in einem separaten Domänenklassendiagramm abgebildet, in welchem die relevanten Klassen aus der Metaebenendefinition über einen Importmechanismus verknüpft und erweitert werden können. In Abbildung 4.2 sind die relevanten Klassen und Abhängigkeiten auf der Metaebene dargestellt.

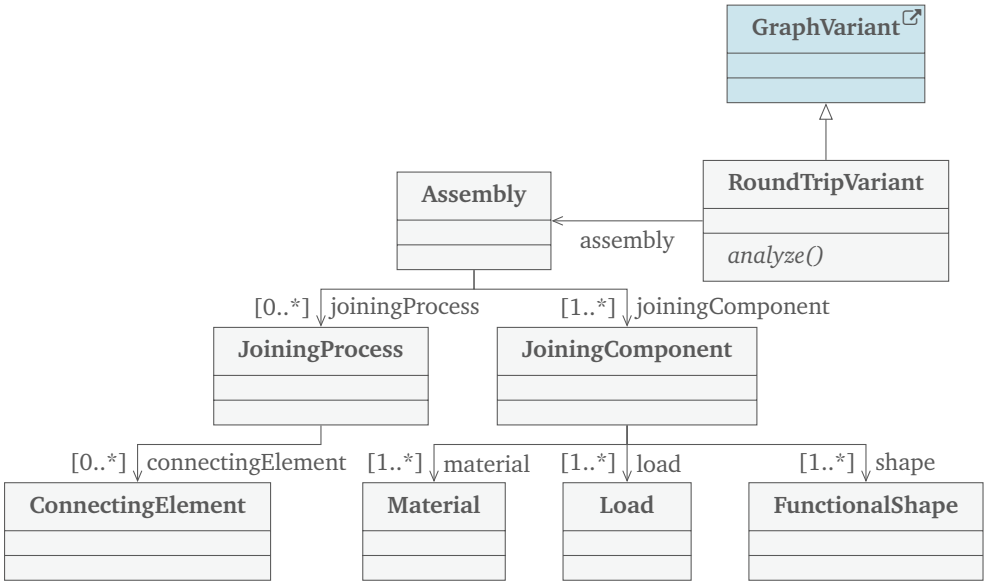


Abbildung 4.2: Klassendiagramm für die Modellierung des Round-Trip-Schemas

Die Klasse **GraphVariant** ist – erkennbar an der blauen Hintergrundfarbe und dem Pfeilsymbol – importiert aus einem anderen Klassendiagramm und stammt in diesem Fall aus der in DC43 integrierten Graphenvariation (siehe Abschnitt 2.2.4). Sie wird für die Modellierung der Variantenauffächerung im Re-Engineering benötigt. Für die konkrete Umsetzung ist die von **GraphVariant** erbende Klasse **RoundTripVariant** zuständig. Die benötigten Anweisungen

werden in der Methode *analyze()* gegeben. Im Wesentlichen wird dort definiert, welche Eigenschaften der Varianten später für die Auswertung zugänglich gemacht werden sollen. Hierfür wird der assoziative Zugriff auf die Klasse **Assembly** benötigt, die den Zusammenbau einer Variante eines mit dem Framework auszulegenden Gesamtproduktes repräsentiert.

Die Klasse **Assembly** ist direkt oder indirekt mit allen Domänenhauptklassen assoziiert. Dadurch sind für sie alle anderen Klassen sowie deren Eigenschaften über die entsprechenden Assoziationen bzw. Assoziationsketten referenzierbar. Der Zusammenbau setzt sich aus einer oder mehreren Komponenten (**JoiningComponent**) zusammen. An diese Komponenten sind die Domäneneigenschaften des Materials (**Material**), der physikalischen Belastung (**Load**) und der Geometrie (**FunctionalShape**) geknüpft, jeweils mit einer eins-zu-vielen-Assoziation.

Für jedes n-Tupel aus zusammenhängenden Subkomponenten eines Zusammenbaus wird ein zugehöriger Fügeprozess (**JoiningProcess**) modelliert. Während ein Zusammenbau aus mindestens einer Komponente besteht, muss es für den Fall einer einzigen vorhandenen Komponente nicht zwangsläufig einen Fügeprozess geben. Aus diesem Grund weist die zugehörige Fügeprozessassoziation im Unterschied zu den restlichen Assoziationen eine Multiplizität von null bis unendlich auf. Die baumartige Struktur der Assoziationen im Metaklassendiagramm ermöglicht einen indirekten Zugriff der Klasse **RoundTripVariant** auf alle weiteren modellierten Klassen durch Assoziationsketten. Die Multiplizitäten erfordern dabei allerdings einen potenziell „multi-iterativen“ Ansatz, um alle relevanten Informationen zu erhalten. Beispielsweise liefert die Assoziationskette *assembly.joiningComponent* eine Liste mit 1..* Komponenten zurück, über die iteriert werden muss. Soll auf Informationen auf einer noch tieferen Ebene – z. B. das Material – zurückgegriffen werden, kommt hierdurch eine weitere Iteration hinzu et cetera.

4.2.2 Funktion

Die Funktionsdomäne wurde in Abschnitt 3.2.2 bereits allgemein beschrieben und die Hauptmodellierungsklassen motiviert. Auf diesen Überlegungen aufbauend wurde das Klassendiagramm, das in Abbildung 4.3 zu sehen ist, abgeleitet. Im Metaklassendiagramm (siehe Abbildung 4.2) hat die Funktionsklasse keinen Eingang gefunden, da sie in dieser sehr abstrakten Formulierung einen Zusammenhang mit (fast) allen dort definierten Klassen aufweisen würde und dies der Übersichtlichkeit nicht förderlich ist. Sie soll aus diesem Grund in den Domänenklassendiagrammen direkt zur weiteren Ausspezifikation importiert werden.

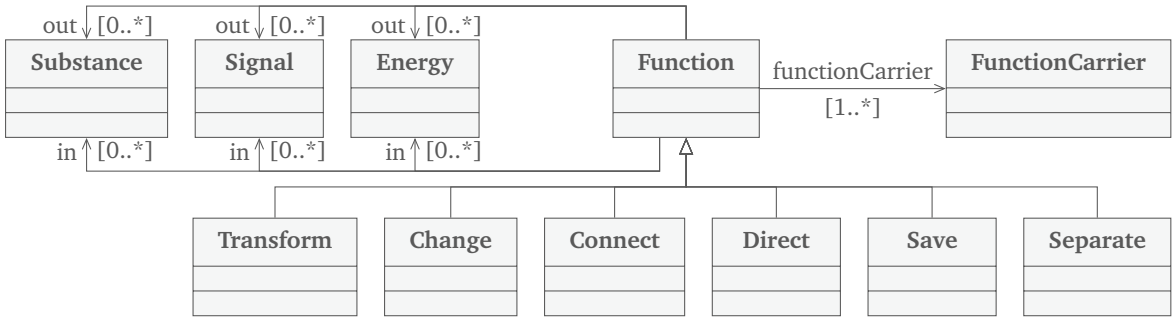


Abbildung 4.3: Klassendiagramm für die Funktionsmodellierung

Die zentrale Klasse **Function** besitzt jeweils eine eingehende (in) und ausgehende (out) Assoziation auf die drei Klassen Stoff (**Substance**), Signal (**Signal**) und Energie (**Energy**). Durch diesen Zusammenhang lässt sich das Verständnis der Funktion als Black-Box-Ansatz bereits modellieren. Zusätzlich gibt es einen Funktionsträger (**FunctionCarrier**), der über eine eins zu viele Assoziation (functionCarrier) mit der Funktionsklasse verknüpft ist. Die generische Klasse **Funktion** wird von sechs konkreteren Funktionsklassen erweitert, namentlich wandeln (**Transform**), ändern (**Change**), verknüpfen (**Connect**), leiten (**Direct**), speichern (**Save**) und trennen (**Separate**), wie dies ebenfalls in Abschnitt 3.2.2 beschrieben wurde.

4.2.3 Geometrie

Wie in Abschnitt 4.2.1 dargelegt wurde, wird die Geometrie im Framework auf oberster Ebene durch die Klasse **FunctionalShape** repräsentiert. In Abbildung 4.4 ist die gewählte Modellierung für das Klassendiagramm der Geometriedomäne zu sehen.

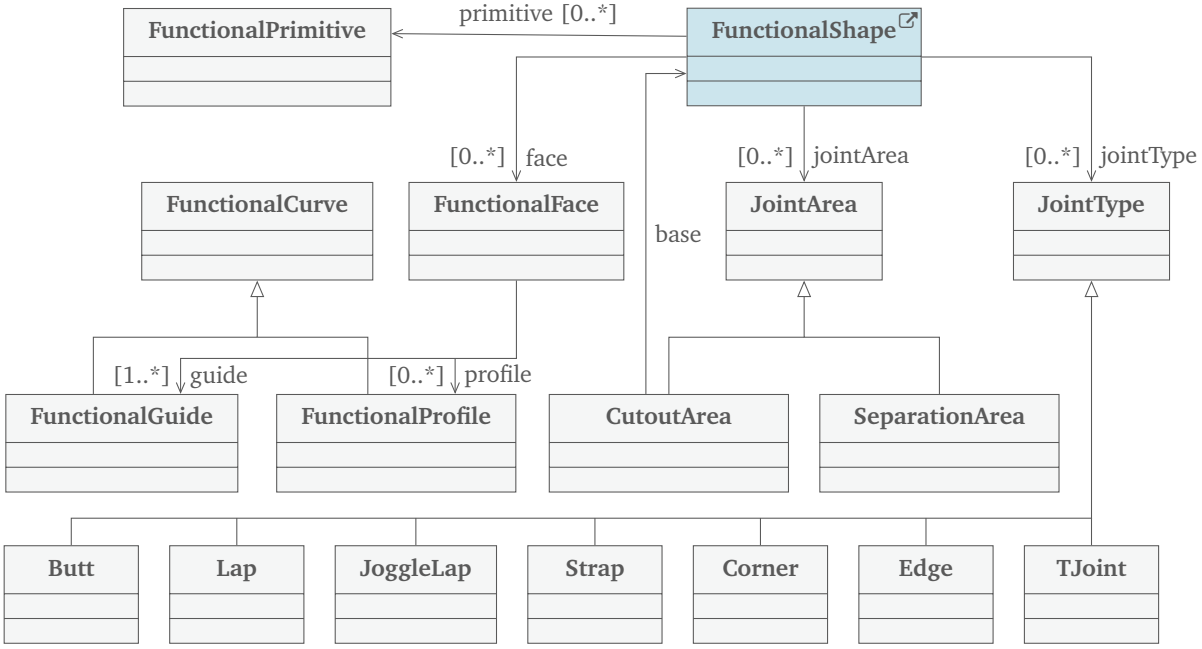


Abbildung 4.4: Klassendiagramm für die Geometriemodellierung

Die Klasse **FunctionalShape** ist aus dem Metaklassendiagramm importiert. Die reine Geometrie eines Bauteils wird wie in Abschnitt 3.2.2 beschrieben durch einen BREP-Ansatz modelliert. Hierfür wird ein Drahtgittermodell durch Profil- (**FunctionalProfile**) und Führungskurven (**FunctionalGuide**) beschrieben. Ein solches Kurvennetzwerk kann mittels einer Fläche (**FunctionalFace**) geschlossen werden.

Intern werden als Flächenrepräsentation Gordon-Surfaces (vgl. Abschnitt 3.2.2) verwendet, sodass das Kurvennetzwerk deren Qualitätsanforderungen zu erfüllen hat. Die Klasse **FunctionalCurve** abstrahiert gemeinsame Eigenschaften der Profil- und Führungskurven. Neben der Grundform des Bauteils sind weitere für das RTE wichtige geometrische Eigenschaften modelliert, insbesondere die Ausgestaltung einer potenziell vorhandenen Fügestelle.

Bei den Fügestellen kann es sich um eine durch Zerteilung entstandene Region des Bauteils handeln (**SeparationArea**) oder aber lediglich um einen Ausschnitt (**CutoutArea**). Dem Ausschnitt wird über die Assoziation *base* noch die konkrete Basisgeometrie zugewiesen, auf der er vorgenommen werden soll. Der eigentliche Übergang an der Fügestelle, der unter anderem auch vom Fügeverfahren abhängig ist, wird durch die Klasse **JointType** repräsentiert. Es sind sieben weitere Klassen – Stumpf- (**Butt**), Überlappungs- (**Lap**), geknickter Überlappungs- (**JoggleLap**), Laschen- (**Strap**), Eck- (**Corner**), Kanten- (**Edge**) und T-Stoß (**T**) – als Spezialisierungen im unteren Bereich von Abbildung 4.4 zu sehen, die die Klasse **JointType** erweitern.

Neben diesen generischen Übergangsformen kann es ein Konstruktionsproblem mit sich bringen, dass eine sehr spezielle Übergangsform gefordert wird. Dies kann einfach durch Hinzufügen einer neuen von **JointType** erbenden Klassen realisiert werden. In diesem Zusammenhang sei auf das Anwendungsbeispiel der Helikopterzelle in Abschnitt 5.3 verwiesen, bei dem eine eigene Übergangsform für die Auskonstruktion eines Rumpfspants definiert wird.

Alternativ zur Formbeschreibung über den BREP-Ansatz können auch Primitive-Geometrien zum Einsatz gebracht werden[1]. Im obigen Klassendiagramm ist dies durch die abstrakte Klasse **FunctionalPrimitve** repräsentiert, die über eine Assoziation mit der Klasse **FunctionalShape** verbunden ist. Im Anwendungsbeispiel des Satelliten (siehe Abschnitt 5.2) wird die Klasse **FunctionalPrimitve** verwendet und für diesen Zweck noch weiter ausspezifiziert.

4.2.4 Lasten

Die physikalischen Lasten sind auf Metaniveau bereits als Klasse **Load** eingeführt worden. In Abbildung 4.5 ist die darüber hinaus gehende Modellierung der Lastendomäne dargestellt.

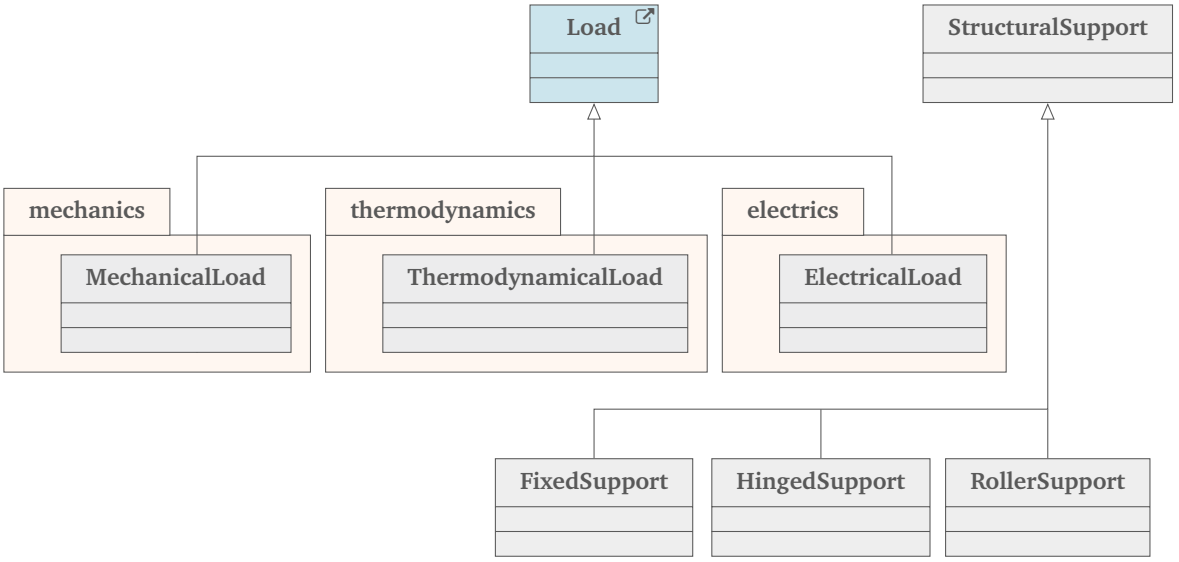


Abbildung 4.5: Klassendiagramm für die Lastmodellierung

Zunächst wurde eine Unterteilung der Lasten in mechanische (**MechanicalLoad**), thermodynamische (**ThermodynamicalLoad**) und elektrische (**ElectricalLoad**) Lasten vorgenommen.

[1] Hierfür wird in der Literatur meist der Begriff CSG verwendet.

Daneben existiert eine Vielzahl weiterer Unterklassen, die je nach Problemstellung ergänzt werden können. Um eine Antwort des Systems auf eine mechanische Last modellieren zu können, bedarf es einer Lagerung (**StructuralSupport**). Auf diesem Niveau der Modellierung wird hier ein Lager in Form von Spezialisierungen noch in ein Festlager (**FixedSupport**), Gelenklager (**HingedSupport**) und ein Loslager (**RollerSupport**) unterteilt. Die weitere Ausdetaillierung der Lasten ist in Unterklassendiagramme – symbolisiert durch die ockerfarbenen Pakete mechanics, thermodynamics und electrics – abgebildet. Diese existieren in der Implementierung, sind aus Gründen der Übersichtlichkeit in dieser Übersicht aber nicht dargestellt.

4.2.5 Material

Die Hauptklasse der Materialdomäne wurde auf Metaniveau bereits als **Material** eingeführt. Die weitere Modellierung wurde inspiriert von der gängigen Unterteilung der Werkstoffe (vgl. Abschnitt 3.2.2). Das zugehörige Domänenklassendiagramm ist in Abbildung 4.6 dargestellt.

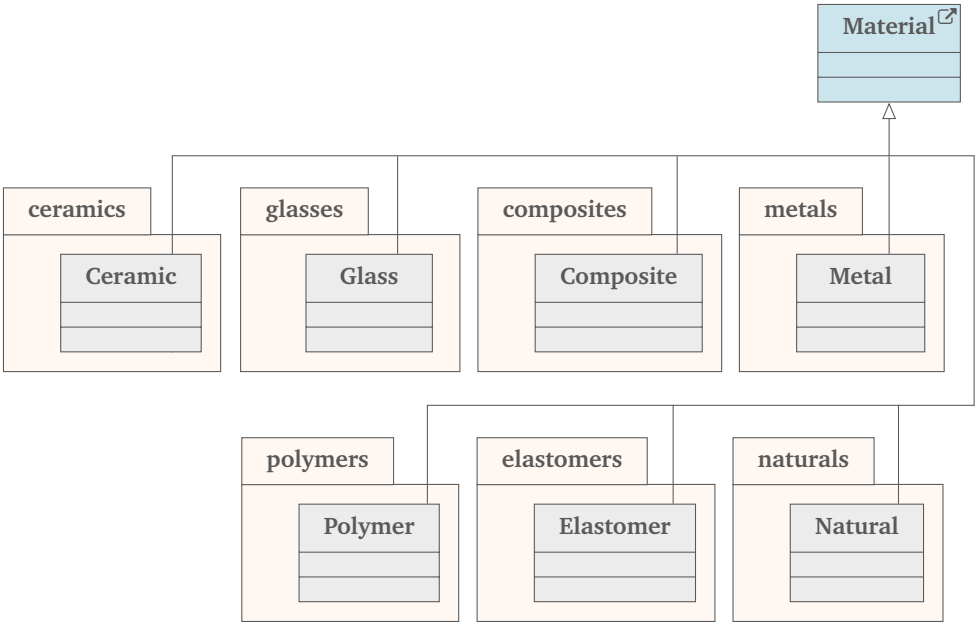


Abbildung 4.6: Klassendiagramm für die Materialmodellierung

Die das Material erweiternden Klassen sind die keramischen Werkstoffe (**Ceramic**), Gläser (**Glass**), Faserverbundmaterialien (**Composite**), Metalle (**Metal**), Polymere (**Polymer**), Elastomere (**Elastomer**) und natürlichen Werkstoffe (**Natural**). Analog zur Lastdomäne sind die ausdetaillierten Materialklassen in der lauffähigen Implementierung in Unterklassendiagramme ausgelagert, die aber aus demselben Grund nicht Teil dieser Darstellung sind.

4.2.6 Verbindungstechniken und -elemente

Die Domäne der Fügeverfahren wird auf Metaklassendiagrammebene durch die Klasse **JoiningProcess** repräsentiert. In Abbildung 4.7 ist das den Fügeverfahren zugehörige Domänenklassendiagramm zu sehen.

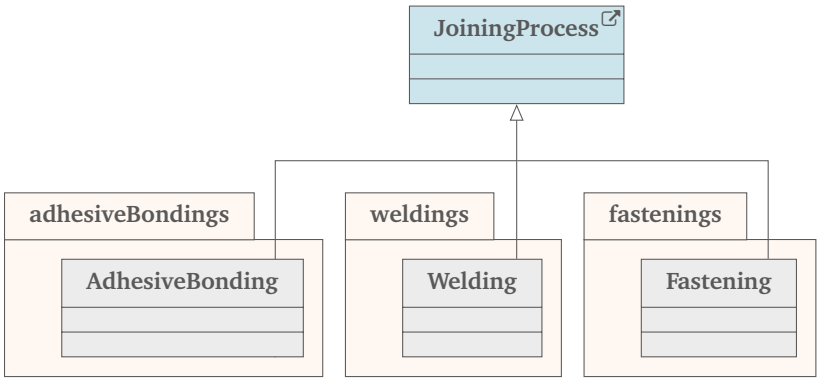


Abbildung 4.7: Klassendiagramm für die Modellierung der Verbindungstechniken

Die aus dem Metaklassendiagramm importierte Klasse **JoiningProcess** wird durch drei spezialisierenden Hauptklassen, namentlich Schweißen (**Welding**), Kleben (**AdhesiveBonding**) und kraftschlüssiges Verbinden (**Fastening**) erweitert. Analog zu den Lasten und Materialien sind weitere Detailmodellierungen dieser Fügeverfahren in Unterklassendiagramme ausgegliedert, aber nicht vollumfänglich dargestellt. Ein Teil davon findet sich später in Abschnitt 5.2.

Die Verbindungselemente sind aus Gründen der Flexibilität und Übersichtlichkeit in einem eigenen Domänenklassendiagramm modelliert. Dieses ist in Abbildung 4.8 grafisch aufbereitet.

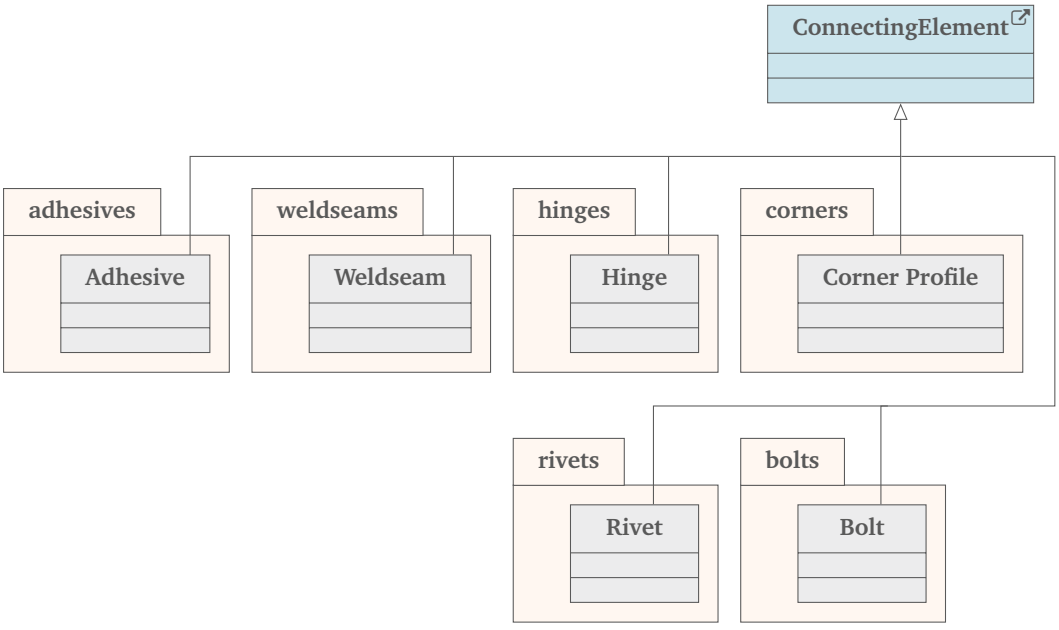


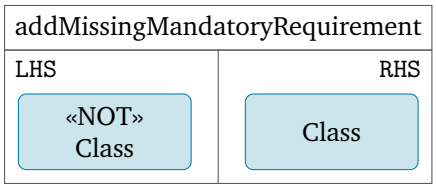
Abbildung 4.8: Klassendiagramm für die Modellierung der Verbindungselemente

Im Klassendiagramm für die Modellierung der Verbindungselemente ist die zugehörige abstrakte Klasse (**ConnectingElement**) aus dem Metaklassendiagramm importiert. Da es sich bei all den erweiternden Klassen um zusätzlich angebrachte Elemente handelt, sind diese teilweise vom gewählten Fügeverfahren abhängig. Bei den stoffschlüssigen Verbindungen wie der Klebe- oder der Schweißverbindung wird in der Regel ein Füllmaterial eingebracht, das durch die Klassen Kleber (**Adhesive**) und Schweißfülldraht (**Weldseam**) repräsentiert wird. Bei

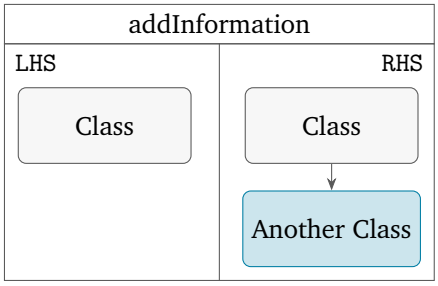
kraftschlüssigen Verbindungen, wie dem Schrauben oder Nieten, werden Verbindungselemente wie Schrauben (**Bolt**) oder Niete (**Rivet**) benötigt. Darüber hinaus können zusätzliche Brückenelemente wie Winkelelemente (**Corner Profile**) oder Scharniere (**Hinges**) für die Verbindung zweier Bauteile eingesetzt werden. Wie auch bei den Verbindungstechniken gibt es unzählige Spezialisierungsklassen dieser Verbindungselemente, die wiederum in Unterklassendiagramme ausgelagert sind. Aus diesem Grund sind die Oberklassen in Paketen gruppiert dargestellt. Ein Spezialisierungsklassendiagramm ist für die Klasse der Schrauben in Abbildung 5.8 zu finden.

4.3 Kontextbasierte Konkretisierung des Entwurfs durch Regeln

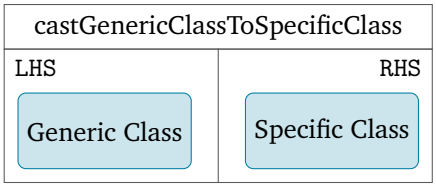
Ganz allgemein ermöglicht die Formulierung generischer Regeln eine auf den situativen Kontext zugeschnittene Anreicherung des Entwurfsgraphen durch Modelltransformationen und damit eine exakte Abbildung der inkrementellen Entwurfskonkretisierung, wie sie auch in den gängigen Entwurfsmethodiken beschrieben wird (siehe Abschnitt 2.2.3). In Abbildung 4.9 sind vier essenzielle Mechanismen auf Regelebene, die für die Abbildung der im nächsten Kapitel beschriebenen Implementierung der dynamischen Regelsequenz benötigt werden, dargestellt.



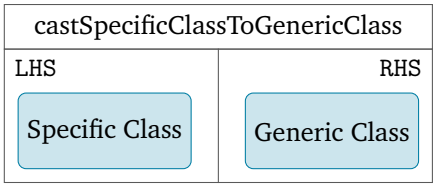
(a) Anreicherung des Graphen bei Nichtspezifikation mandatorischer Randbedingungen



(b) Anreicherung des Graphen mit ableitbaren Zusatzinformationen



(c) Implementierung des Spezifikationscasts



(d) Implementierung des Abstraktionscasts

Abbildung 4.9: Regelformulierungen für die Implementierung der dynamischen Regelsequenz

In Abbildung 4.9a wird illustriert, wie auf Regelebene nicht spezifizierte, aber mandatorische Randbedingungen in den aktuellen Entwurfskontext eingebracht werden können. Auf der Linkshandseite der Regel wird dazu über das Schlüsselwort „`NOT`“ das Nichtvorhandensein eines definierten Elements auf dem aktuellen Graphen abgeprüft. Ist diese Bedingung erfüllt – das gesuchte Objekt somit im aktuellen Entwurfskontext nicht vorhanden –, wird durch die Anweisung auf der Rechtshandseite dieser Regel ein solches mandatorisches Element erzeugt. Meist wird die Rechtshandseite einer solchen Regel auf abstraktester Ebene gegenüber dem zugehörigen Klassendiagramm definiert, es ist in Ausnahmefällen aber ebenfalls möglich, eine

konkretere Klasse zu wählen[2]. Auf diese Weise wird folglich eine unbestimmt-diskrete Randbedingung (vgl. Tabelle 3.2) in den aktuellen Entwurfskontext eingebracht. Als Beispiel für eine mandatorische Randbedingung kann abermals das Material bemüht werden, denn ein reales Produkt benötigt zwangsweise einen Werkstoff, aus dem es hergestellt werden soll. Sollte dieser nicht durch den/die Anwender/-in abstrakt oder konkret vorgegeben worden sein, wird der Graph durch eine solche Regel *automatisiert* mit Zusatzwissen angereichert. Dieses Wissen muss a priori durch die Formulierung einer entsprechenden Regel vorgehalten und nur im Fehlfall zur Laufzeit aus dem aktuellen Entwurfskontext folgend ergänzt werden. Wird die Linkshandseite nicht gematcht, wird die Regel durch einen angegebenen Alternativpfad übersprungen.

In Abbildung 4.9b ist eine weitere Möglichkeit für die Anreicherung des Graphen mit Zusatzinformation dargestellt. In diesem Fall wird bei Vorhandensein einer bestimmten Klasse, ein neues Objekt erzeugt. Alternativ dazu, können in der gesuchten Klasse auch Eigenschaften ergänzt werden[3]. Diese Art der Regelformulierung wird in der vorgestellten Implementierung des Frameworks in allen Domänen eingesetzt. Oftmals werden auch die Regelformulierungen aus den Abbildungen 4.9a und 4.9b kombiniert. Als eindrückliches Beispiel hierfür kann die Geometriedomäne (siehe Abschnitt 4.2.3) angeführt werden. Wird durch den/die Anwender/-in ein Kurvennetzwerk in Form von **FunctionalGuides** und **FunctionalProfiles** vorgegeben, muss es eine Regel geben, die aus diesen Linien die Anweisung zur Erzeugung der **FunctionalFace** gibt. Damit dies allerdings nicht bei jedem iterativen Durchlauf der Domänenaktivität erneut geschieht, wird das Nichtvorhandensein eines Flächenelements mit in die Regelbedingung (`LHS`) aufgenommen. Durch diesen Mechanismus wird sichergestellt, dass für jeden Variantengraphen immer nur eine Fläche für ein definiertes Kurvennetzwerk erzeugt wird bzw. vorhanden ist.

In Abbildung 4.9c wird die Formulierung eines Spezifikationscasts (vgl. Abschnitt 3.1.3) gezeigt. Eine dergestalt formulierte Regel überführt ein abstrakt vorhandenes Element in ein konkreteres und stellt folglich ein wesentliches Werkzeug der Entwurfskonkretisierung dar.

Abschließend wird in Abbildung 4.9d ein Abstraktionscast in einer Regel gezeigt. Dabei wird eine spezifische Klasse auf eine abstraktere Klasse vor dem Hintergrund des zugehörigen Klassendiagramms gecastet. Dieser Vorgang wird in erster Linie beim Reverse-Engineering benötigt. Wie bereits erwähnt, wird erst durch die Kombination verschiedener Elemente die dynamische Regelsequenz ermöglicht, womit sich der nachfolgende Abschnitt beschäftigt.

4.4 Dynamische Entwurfssequenz als Aktivitätsdiagramm

Die Implementierung der dynamischen Entwurfssequenz basiert auf mehreren verknüpften Mechanismen auf Aktivitätsdiagrammebene der Entwurfssprache. Wesentlich sind dabei die im vorausgegangenen Kapitel beschriebenen Formulierungen von Regeln sowie die im Folgenden vorgestellten Variationsblöcke und Iterationsschleifen. In Abbildung 4.10 ist eine vereinfachte Subaktivität der Materialdomäne präsentiert. Es handelt sich um eine Kombination aus Anreicherungsregeln, Nichtspezifikationsregeln, Spezifikationscasts und Variationsblöcken.

[2] Die Definition auf abstraktester Ebene garantiert eine maximale Entwurfsraumgröße und sorgt dafür, dass keine potenziell günstigen Varianten unbegründet ausgeschlossen werden.

[3] Dies ist in Abbildung 4.9b nicht dargestellt.

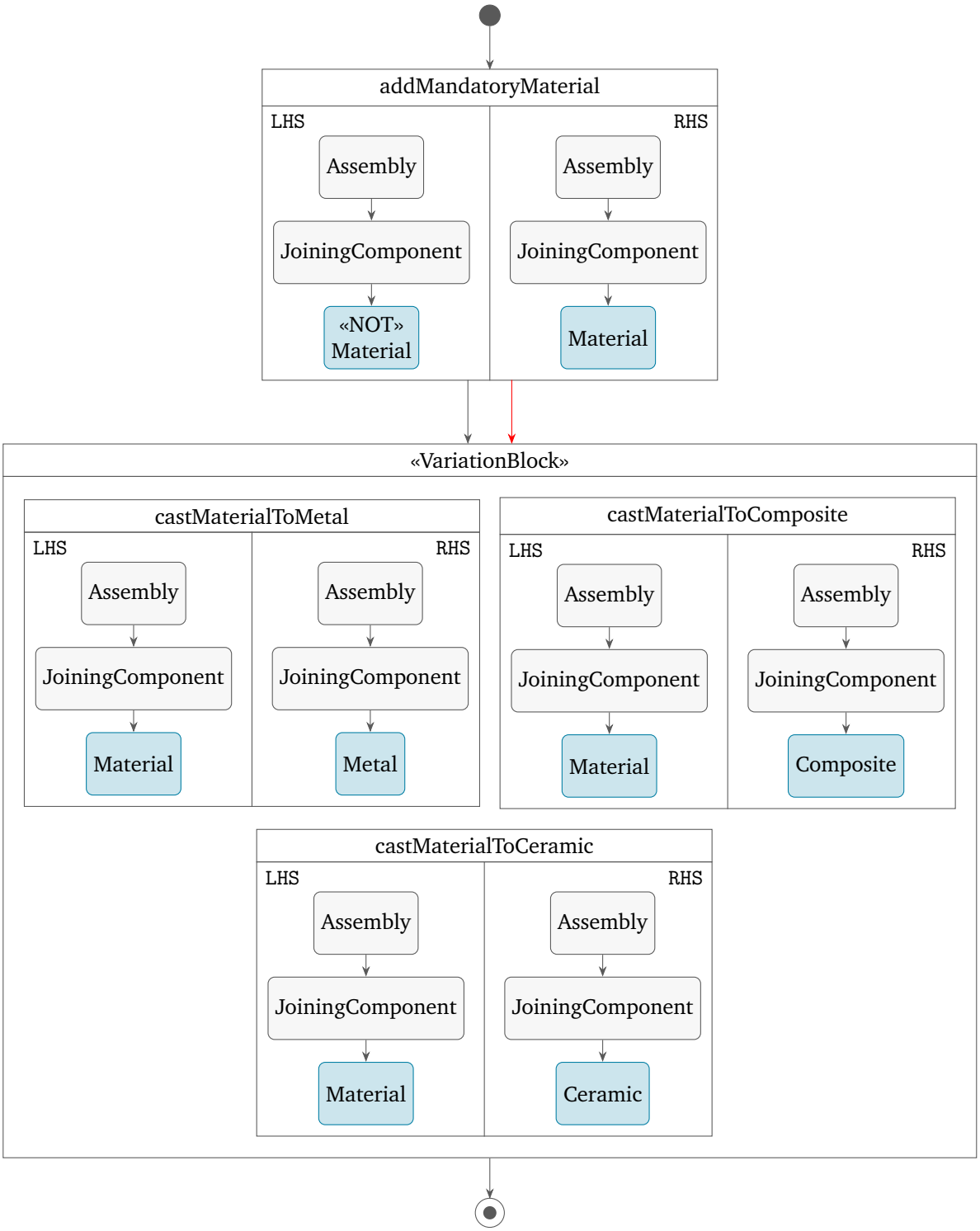


Abbildung 4.10: Domänensubaktivität mit Variationsblock am Beispiel des Materials

Zunächst wird zur Laufzeit die Nichtspezifikationsregel addMandatoryMaterial durchlaufen und somit im Falle des Nichtvorhandenseins eine Instanz vom Typ Material erzeugt. Sollte die erste Regel in Abbildung 4.10 fehlschlagen, da bereits eine abstrakte oder konkrete Materialinstanz vorhanden ist, wird der in roter Farbe gekennzeichnete alternative Kontrollfluss verfolgt und somit in jedem Fall der darunter befindliche Variationsblock als Nächstes aufgerufen.

Ein Variationsblock nimmt den zum Zeitpunkt des Aufrufs vorhandenen Graphen und fertigt davon exakt so viele Kopien an, wie es definierte Regeln innerhalb des Blocks gibt. Die Regeln werden dann auf die zugehörige Graphenkopie – den Variantengraphen – angewendet. Durch diesen Vorgang entstehen die beim Re-Engineering (vgl. Abschnitt 3.1.3) geforderten Produktvarianten völlig automatisiert. Die Anzahl der Regeln in einem Variationsblock ist nicht begrenzt und in diesem Beispiel für eine bessere Übersichtlichkeit auf drei Regeln beschränkt.

Handelt es sich bei den beinhaltenden Regeln – wie im vorliegenden Fall – um Spezifikationsregeln im Sinne eines Spezifikationscasts, werden durch diesen Mechanismus immer konkretere Varianten erzeugt. Die Spezifikationscasts sind der Übersichtlichkeit halber in diesem Beispiel als graphische Regeln ausgeführt. In der tatsächlichen Implementierung handelt es sich um Code-basierte Regeln, die in einem Durchlauf abhängig vom spezifizierten Maximaltiefenparameter nicht nur einmalig ausgeführt werden, sondern direkt alle auf der tiefstmöglichen Definitionsebene des zugehörigen Klassendiagramms möglichen Varianten erzeugt.

Hierbei handelt es sich um die weiter unten noch genauer beschriebene innere Iterationsschleife des Frameworks. Analog zu dem in Abbildung 4.10 gezeigten Aufbau müssen die Aktivitäten für jede Domäne formuliert werden. Die Einbettung der domänenspezifischen Aktivitäten in den Gesamtkontext der Entwurfssprache für RTE ist in Abbildung 4.11 schematisch illustriert. Hierbei soll die Bedeutung dreier ineinander geschachtelter Iterationsschleifen für die Implementierung der dynamischen Entwurfssequenz herausgearbeitet werden.

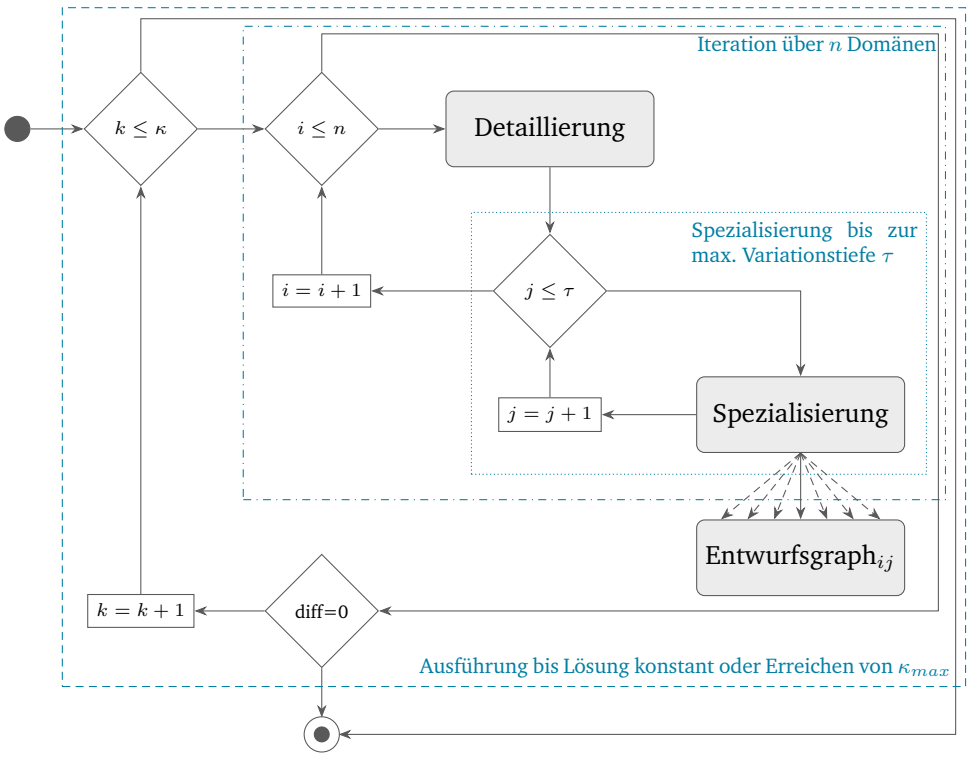


Abbildung 4.11: Dreistufiges Iterationsschema der Implementierung der dynamischen Entwurfssequenz in GBES

Die Abfolge der Regeln, die innerhalb des Frameworks ausgeführt werden, wird in erster Linie von den vom/von der Anwender/-in in Form eines initialen Entwurfsgraphen (*Anforderungsgraph*) spezifizierten Randbedingungen determiniert[4]. Da in voller Allgemeinheit kein zyklenfreier Anforderunsgraph angenommen werden kann, kann die Regelsequenz nicht a priori vollumfänglich algorithmisch beispielsweise durch topologisches Sortieren oder andere Analyseverfahren bestimmt bzw. definiert werden (vgl. Abschnitt 3.2.3).

Auch ein zunächst zyklenfreier Anforderungsgraph kann in Folge der Ausführung einer nachgelagerten Regel innerhalb des Frameworks durch die Anreicherung mit Details in einen zyklischen Graphen transformiert werden. Die tatsächliche Regelsequenz der Entwurfssprache wird aus diesem Grund innerhalb der hier vorgestellten Implementierung erst zur Laufzeit durch einen mehrfach iterativen Prozess bestimmt. Hierfür werden, wie in Abbildung 4.11 zu sehen, in das zunächst statische Aktivitätsdiagramm drei Iterationsschleifen eingeführt.

Die innerste Schleife (gepunkteter Bereich in Abbildung 4.11) ist für das Aufspannen des Entwurfsraums aus den beteiligten Domänenontologien heraus zuständig. Die Anzahl an Schleifendurchgängen pro Domäne lässt sich aus der Modellierungstiefe des jeweils zugehörigen Domänenklassendiagramms ableiten. Der zugehörige Maximaltiefenparameter ist ontologieinhärent und daher zunächst laufzeitunabhängig, kann aber auch durch eine manuelle Vorgabe in Form einer festen Randbedingung durch den/die Anwender/-in überschrieben werden. Während der Ausführung der Entwurfssprache bleibt der Maximaltiefenparameter dann aber fortan konstant. Hinter dem Platzhalter „Spezialisierung“ verbirgt sich ein domänenspezifisches Unterprogramm mit Variationsblock, wie dies exemplarisch für die Materialdomäne in Abbildung 4.10 vorgestellt wurde. Die Spezifikationscasts innerhalb der Variantenblöcke der Domänenaktivitäten führen auf die verschiedenen Produktvarianten, die, wie bereits erwähnt, jeweils durch einen eigenen Entwurfsgraphen repräsentiert werden („Entwurfsgraph$_{ij}$“).

Um die Spezialisierungsschleife herum, ist eine weitere Iteration, die als *Domänenschleife* bezeichnet wird, eingebettet. In dieser wird über alle n vom Framework zu berücksichtigenden Domänen iteriert. In dieser Schleife werden darüber hinaus Anreicherungsregeln in den Domänenaktivitäten ausgeführt, die auf alle Variantengraphen angewendet werden müssen („Detaillierung“). Die Anzahl der Durchläufe der Domänenschleife kann erst zur Laufzeit durch die Auswertung der im aktuellen Ausführungskontext angegebenen Randbedingungen bestimmt werden. Dabei sei unbedingt darauf hingewiesen, dass durch die domänenübergreifende und iterative Ausführung aller Aktivitäten durch das Entwurfsrahmenwerk und die damit einhergehende Wissensanreicherung des Graphen zur Laufzeit potenziell weitere Randbedingungen hinzukommen können und sich dadurch die tatsächlich benötigte Schleifenanzahl nochmals ändern kann. Dieser Sachverhalt liegt in der iterativen Ausführung des Frameworks selbst begründet, da in einem erneuten Durchgang nun ein und dieselbe Subaktivität mit einem geänderten Graphen durchlaufen wird und somit potenziell Graphenmuster vorhanden sein können, die zur Ausführung vormals noch nicht matchender Regeln führen können. Die Endlichkeit dieser Prozedur ist durch den pro Domäne gewählten Maximaltiefenparameter oder im Falle der Nichtspezifikation durch die endliche Tiefe der Domänenklassendiagramme aber in

[4]Eine zentrale Rolle spielt dabei die Festlegung der festen Randbedingungen.

jeden Fall sichergestellt. Es soll an dieser Stelle nicht unerwähnt bleiben, dass es aufgrund der Klassen- und Parametervielfalt in bestimmten Anwendungsfällen zu sehr langen Laufzeiten bzw. Speicherproblemen bei der Ausführung kommen kann. Eine starke Eingrenzung des zu untersuchenden Bereichs durch den/die Anwender/in sollte diesem Trend allerdings entgegenwirken.

Aus genau diesem Grund wird noch eine letzte Iterationsschleife als Klammer um die beiden anderen Iterationsschleifen herum eingeführt, die, nachdem alle Durchläufe auf innerer und mittlerer Ebene abgeschlossen wurden, noch einmal all diese Schritte initiiert. Diese äußere Schleife wird so oft wiederholt, bis sich keine weiteren Änderungen der gebildeten Variantengraphen mehr ergeben[5] und somit das Abbruchkriterium erreicht wurde. Alternativ kann ein fest vom/von der Anwender/-in gesetzter äußerer Maximaliterationswert κ_{max} spezifiziert werden, nach dessen Erreichen der Vorgang automatisch abgeschlossen wird.

Aus informationstheoretischer Sicht kann die Einführung der drei Iterationsschleifen als konkrete Implementierung eines Graphenalgorithmus – im vorliegenden Fall einer iterativen Tiefensuche – mit den Mitteln eines Aktivitätsdiagramms und den speziellen Regelformulierungen aufgefasst werden. Durch den beschriebenen Mechanismus werden zur Laufzeit und basierend auf dem Entwurfsgraphen wiederkehrend die verschiedenen Regeln besucht und die auf dem Klassendiagramm basierenden Expansionsmechanismen in Gang gesetzt.

Durch die Einführung der Iterationsschleifen in der Darstellung der Informationsarchitektur wird darüber hinaus deutlich, dass für etwaige Analysen benötigte Exporte (z. B. Simulationen) nicht erst am Ende eines Entwurfssprachenlaufs, sondern fortwährend und wiederkehrend zur Laufzeit der Entwurfssprache getriggert werden können. Technisch ist es auch in der klassischen Vorgehensweise ohne Weiteres möglich, jederzeit Exporte zur Laufzeit zu initiieren.

Die bereits an früherer Stelle in der vorliegenden Arbeit thematisierte Kontextabhängigkeit der Regeln sei an dieser Stelle noch einmal ins Gedächtnis gerufen. Sie rührt von der Tatsache her, dass sich Regeln immer auf den Zustand des Graphen vor der Regelausführung beziehen. Ändert sich dieser Zustand beispielsweise durch Transformationen in nachfolgenden Regeln, so kann eine ursprünglich nicht ausführbare beziehungsweise matchende Regel plötzlich doch potenziell ausführbar werden. Dies ist der Grund, warum ein iteratives oder qualitativ äquivalentes Vorgehen bei der Ableitung der Regelsequenz unabdingbar ist.

Im Umkehrschluss kann aus dieser Tatsache gefolgert werden, dass die Regelsequenz somit nur für ein festes Set an Randbedingungen als konstant angesehen werden kann. Obwohl sowohl parametrische als auch topologische Änderungen unter bestimmten Voraussetzungen in einer festen Regelsequenz abgebildet werden können, – dies wurde in zahlreichen Vorarbeiten bereits gezeigt – hängt es stark von der Formulierung der einzelnen Regel ab, wie flexibel die Entwurfssprache auf geänderte Randbedingungen reagieren kann. Durch das Einfügen von Zweigstellen (sog. Decision Nodes) in den Aktivitäten kann dies auch in klassischen Entwurfssprachenformulierungen teilweise abgefangen werden. In voller Allgemeinheit können starre Aktivitäten aber nicht den gesamten Entwurfsraum abbilden, da in diesem Fall jede Eventualität in einer eigenen „if-else-Formulierung“ abgefangen werden müsste.

[5] Dabei wird der letzte Stand des jeweiligen Variantengraphen vor der aktuellen Iteration mit dem aktuellen Graphen verglichen und detektiert, ob sich Knoten, Kanten oder Eigenschaften der Objekte geändert haben.

5 Anwendungsbeispiele

> *Es ist nicht genug zu wissen – man muss auch anwenden.*
> *Es ist nicht genug zu wollen – man muss auch tun.“*
>
> **— Johann Wolfgang von Goethe**
> (deutscher Dichter und Naturforscher)

In diesem Kapitel wird die Anwendung und Ausführung der im vorangegangenen Kapitel beschriebenen Implementierung anhand mehrerer Beispiele gezeigt. Dadurch soll sowohl verifiziert werden, dass die in Abschnitt 3.1 erdachte Philosophie korrekt umgesetzt wurde, als auch validiert werden, dass die Implementierung sinnvolle Ergebnisse liefert.

Um eine hinlängliche Bandbreite an Anwendungsfällen abzudecken, wurden drei verschiedene Beispiele aus unterschiedlichen Bereichen konzipiert. Jedes einzelne Beispiel legt dabei den Hauptfokus auf einen anderen Aspekt der Implementierung.

Zunächst wird anhand von Blechbauteilen die Formvariation in Zusammenhang mit einer Verbindungstechnik gezeigt. Dabei steht die Formdefinition über Kurvennetzwerke und deren automatische Umkonstruktion unter den gegebenen Randbedingungen im Vordergrund.

Es folgt die Demonstration des Reverse-Engineering am Beispiel eines Satellitengehäuses, wobei die Abstraktion einer gegebenen Konstruktion auf Basis eines vorhandenen Entwurfsartefakts vor dem Hintergrund objektorientierter Universalontologien demonstriert wird.

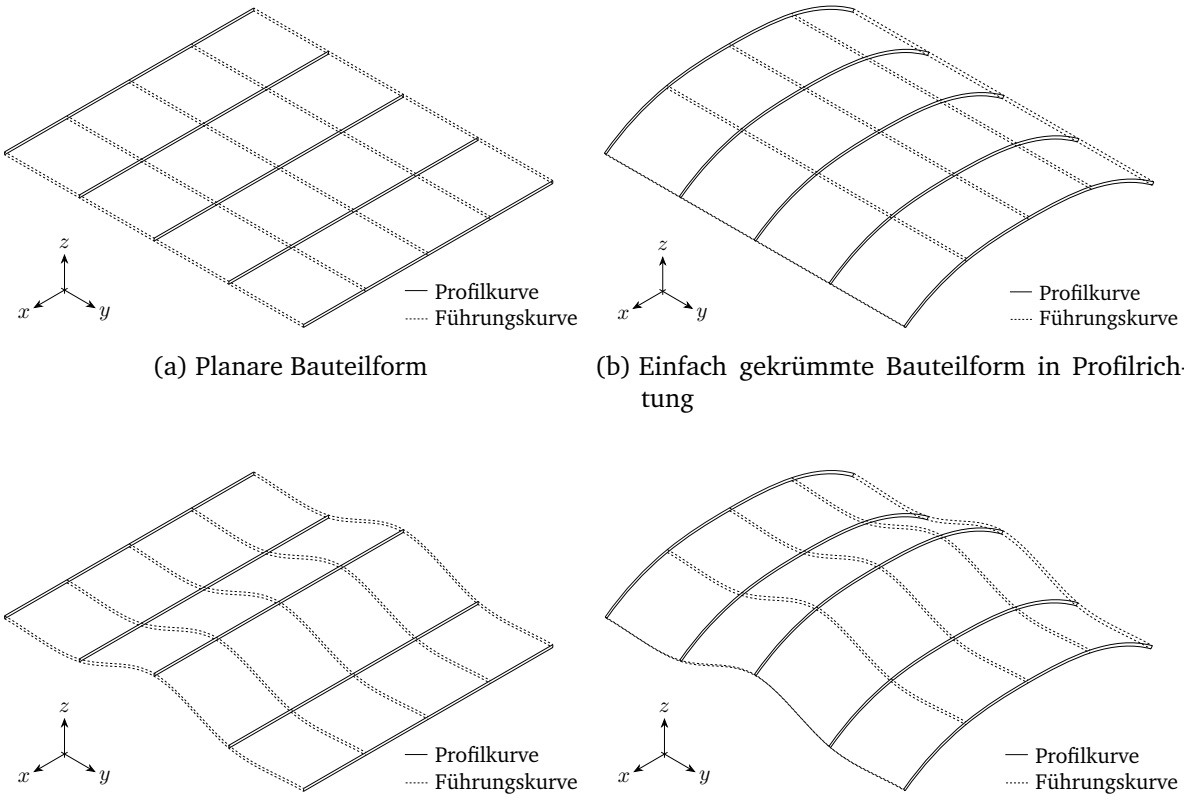
Im letzten Beispiel wird die automatisierten Konstruktion von Hubschrauberstrukturelementen vorgestellt, wobei konkret ein Spant als Strukturbauteil mit erhöhter Komplexität sowie Tür- und Fensterausschnitte behandelt werden. Dieses Beispiel ist aus einer Zusammenarbeit mit Airbus Helicopters hervorgegangen. Es belegt die Eignung des Frameworks für den industriellen Einsatz am Übergang vom Vor- zum Detailentwurf und zeigt darüber hinaus die Anwendung des „elektronischen Messers“ für Strukturausschnitte wie Fenster oder Türen.

Das Kapitel schließt mit einer inhaltlichen und implementatorischen Diskussion der in dieser Arbeit vorgestellten Methodik und der erzielbaren Ergebnisse.

Kapitelübersicht

5.1 Verbindungen dünnwandiger Strukturbauteile

In diesem Abschnitt soll die Anwendung des Entwurfsrahmenwerks am Beispiel von Blechbauteilkonstruktionen auf Basis verschiedener Verbindungstechniken aufgezeigt werden. Hierbei soll die Definition der Geometrie und deren Umkonstruktion am Übergang zweier Bauteile (Stoßstelle) herausgestellt werden. Als Erstes muss die Modellierung der Grundform des Blechbauteils vorgenommen werden. Für diesen Zweck wird, wie in Abschnitt 3.2.2 beschrieben, über einzelne Kurven zunächst ein Drahtgitternetzwerk beschrieben, das anschließend durch ein oder mehrere Gordon-Surfaces interpoliert wird. In Abbildung 5.1 sind vier verschiedene Drahtgitternetzwerke mit unterschiedlicher geometrischer Komplexität aufgeführt.



(a) Planare Bauteilform

(b) Einfach gekrümmte Bauteilform in Profilrichtung

(c) Einfach gekrümmte Bauteilform in Führungsrichtung

(d) Mehrfach gekrümmte Bauteilform

Abbildung 5.1: Definition von Ober- und Unterseite der Blechgeometrie über Profil- und Führungskurven unterschiedlicher Krümmung

Die abgebildeten Drahtgittermodelle bestehen aus jeweils 2 Kurvennetzwerken für die Ober- und Unterseite des Bauteils. Die Kurvennetzwerke der Ober- und Unterseite setzten sich aus jeweils 10 Kurven zusammen. Die einzelnen Kurven sind ihrerseits wiederum über jeweils fünf Kontrollpunkte und die zugehörigen Tangentialvektoren als NURBS repräsentiert.

Bei der Definition des Drahtgitters wird zwischen Profil- und Führungskurven unterschieden. Die Profilkurven sind in der Abbildung als durchgezogene Linien dargestellt, die Führungskurven gestrichelt. Die später zur Anwendung gebrachte Interpolation mit Gordon-Flächen kann als Erweiterung einer typischen Loft-Funktion angesehen werden, bei der beliebig viele

Rand- und Führungskurven vorgegeben werden können. Die Führungskurven sind die Kurven, entlang derer die Fläche während des Interpolationsvorgangs extrudiert wird. Sowohl Profil- als auch Führungskurven sind infolgedessen feste Interpolationsbereiche der gesuchten Fläche. Durch diesen Sachverhalt wird nochmals eindrücklich verdeutlicht, warum sich alle Profil- und Führungskurven einer zu beschreibenden Fläche per definitionem schneiden müssen.

Von der planaren Grundform in Abbildung 5.1a ausgehend, wird die Position der Kontrollpunkte der Profilkurven verändert, was auf Abbildung 5.1b führt. In Abbildung 5.1c sind die Profilkurven im Vergleich zu Abbildung 5.1a unverändert, jedoch die Kontrollpunkte der Führungskurven modifiziert. Zuletzt ist in Abbildung 5.1d eine Kombination der Veränderungen aus Abbildung 5.1b und Abbildung 5.1c zu sehen, die als Ausgangsform für den weiteren Verlauf dieses Beispiels dienen soll. Durch die zweifache Krümmung des Bauteils soll die Übertragbarkeit der Vorgehensweise auf andere Anwendungsfälle durch die gesteigerte geometrische Komplexität plausibilisiert werden. Prinzipiell sind im Entwurfsrahmenwerk die Anzahl beschreibender Kontrollpunkte, Kurven und Kurvennetzwerke frei wählbar und infolgedessen sowohl parametrische als auch topologische Varianten beliebiger Bauteilgeometrien abbildbar.

In Abbildung 5.2 ist das Ergebnis der Interpolation des Drahtgitters aus Abbildung 5.1d mit den im Framework integrierten Gordon-Surfaces zu sehen.

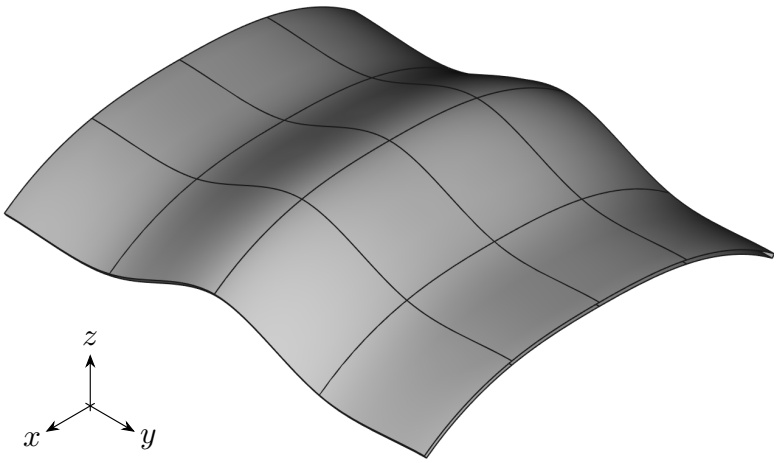


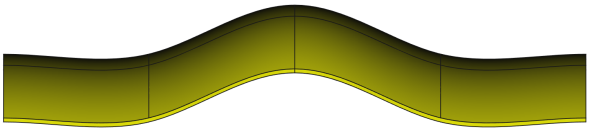
Abbildung 5.2: Blechgeometrie nach der Interpolation der Profil- und Führungskurven aus Abbildung 5.1d

Die Geometrie des Blechbauteils ist durch insgesamt vier Flächen begrenzt, eine obere (positive z-Richtung), eine untere (negative z-Richtung), eine linke (positive x-Richtung) und eine rechte (negative x-Richtung). Die vordere (positive y-Richtung) und hintere Fläche (negative y-Richtung) sind bewusst offen gelassen, da es sich in diesem Beispiel nur um einen lokalen Ausschnitt aus einem größeren Bauteil handeln soll. Die Grundform der Geometrie dieses Beispiels ist damit definiert und dient für die weiteren Schritte als unbestimmte Randbedingung.

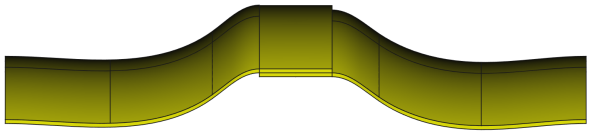
In der vorliegenden Konstruktionsphilosophie wird die Ausgangsform als *ungestörte* Geometrie bezeichnet, die durch geometriewirksame Randbedingungen in eine *gestörte* – und

somit bestimmte – Geometrie überführt wird[1]. Beim in diesem Abschnitt betrachteten Beispiel der Blechbauteile kann eine solche Randbedingung beispielsweise aus einer Begrenzung der Maximalfläche herrühren, da das Ausgangsmaterial gewöhnlich als Rollenware (fast) unbegrenzter Länge, aber beschränkter Breite zur Verfügung steht. Überschreitet die gewünschte Grundform diese Breite, muss das Bauteil aus mehreren Einzelteilen zusammengesetzt werden. Das Zusammensetzen erfordert wiederum eine an einer definierten Stelle zum Einsatz gebrachte Verbindungstechnologie und eine dazu passende Ausführung der Stoßgeometrie.

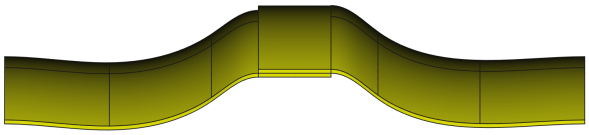
In Abbildung 5.3 sind die fünf im Entwurfsrahmenwerk in Form graphenbasierter Entwurfsmuster vorimplementierte, generische Stoßgeometrien bereits auf die in diesem Beispiel gegebene Primärform angewendet dargestellt.



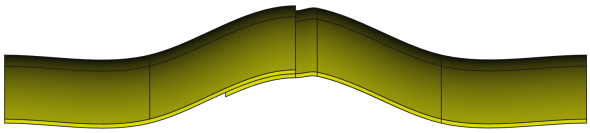
(a) Stumpfstoß (Butt)



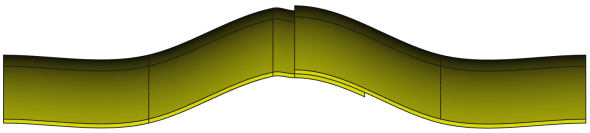
(b) Überlappungsstoß (Lap)



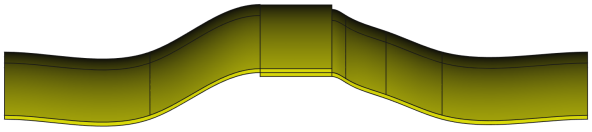
(c) Invertierter Überlappungsstoß (Lap inverted)



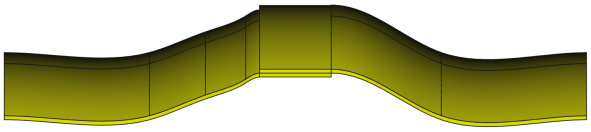
(d) Überlappungsstoß mit konturfolgendem S-Schlag (Joggle Lap)



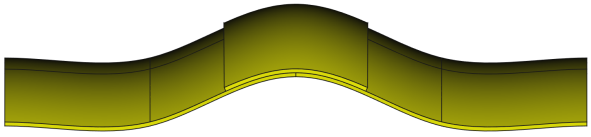
(e) Invertierter Überlappungsstoß mit konturfolgendem S-Schlag (Joggle Lap inverted)



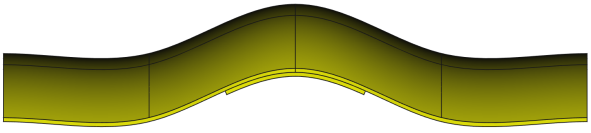
(f) Überlappungsstoß mit geradem Mittelteil und S-Schlag (Joggle Lap straight mid part)



(g) Invertierter Überlappungsstoß mit geradem Mittelteil und S-Schlag (Joggle Lap straight mid part inverted)



(h) Laschenstoß außen (Strap)



(i) Laschenstoß innen (Strap underneath)

Abbildung 5.3: Verschiedene im Entwurfsrahmenwerk vordefinierte Stoßarten

[1]Unter geometriewirksamen Randbedingungen sind solche Randbedingungen zu verstehen, die eine zwangsläufige Veränderung der Ausgangsform mit sich bringen.

Von oben nach unten handelt es sich in Abbildung 5.3 um Stumpf-, Überlappungs-, konturfolgenden Überlappungs-, geraden Überlappungs- und Laschenstoß. An dieser Stelle sei noch einmal an das Formklassendiagramm aus Abbildung 4.4 erinnert, in der die gezeigten Stoßgeometrievarianten bereits abstrakt definiert wurden. Daneben besteht für den Anwender die Möglichkeit, der Bibliothek eigene Stoßgeometriemuster hinzuzufügen. Dies setzt allerdings Grundkenntnisse in Entwurfssprachen, Java und OpenCASCADE® [Open Cascade, 2022] voraus.

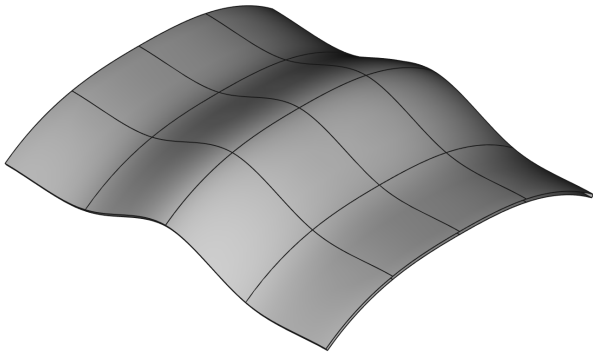
Mit Ausnahme des Stumpfstoßes (siehe Abbildung 5.3a), bei dem der Effekt nicht auftritt, ist gut zu erkennen, dass die Grundgeometrie an der Schnittstelle des digitalen Messers durch die Einführung des Stoßes lokal verändert wird. Während ein Laschenstoß lediglich gebietsweise die Dicke des Bauteils verändert, sind bei den Überlappungsstößen 5.3b - 5.3g zusätzlich auch die Änderungen der Steigung und Krümmung der Ausgangsgeometrie innerhalb eines von der definierten Überlappungslänge abhängigen Bereichs zu erkennen. Diese als Störungen der Ausgangsform aufgefassten Veränderungen haben zur Folge, dass eine weitere Iteration im Entwurfsrahmenwerk benötigt wird, um zu überprüfen, ob durch diese erst im Verlauf der Auskonstruktion hinzugekommenen Veränderungen ursprünglich erfüllte Randbedingungen, wie beispielsweise die Einhaltung der Bauraumbegrenzung, nun nicht doch verletzt werden. Dies betrifft darüber hinaus auch sämtliche sekundär abhängigen Bilanzgrößen der ausgelegten Varianten. Dazu zählt zum Beispiel ein potenziell gefordertes Maximalgewicht, da das Gesamtgewicht durch Zusatzgeometrien wie eine Lasche oder Verbindungselemente erhöht wird. Zusammengefasst müssen alle festen Randbedingungen mit den getroffenen Konkretisierungen und festgelegten Parametern iterativ auf globale Konsistenz überprüft werden.

Neben den Stoßgeometrien kann das gewählte Fertigungsverfahren ebenfalls Veränderungen an der Geometrie notwendig machen. Soll der Übergang von einem auf ein anderes Blech beispielsweise mit einem mechanischen Verbindungsverfahren wie Verschraubung oder Vernietung realisiert werden, so sind Löcher für die Verbindungselemente vorzusehen. Bei anderen Verfahren wie dem Schweißen oder Kleben werden zwar in der Praxis ebenfalls Veränderungen an der Form des Bauteils hervorgerufen – beispielsweise durch die Schweiß- oder Klebenaht –, diese werden aber typischerweise in der technischen Auslegung geometrisch nicht berücksichtigt.

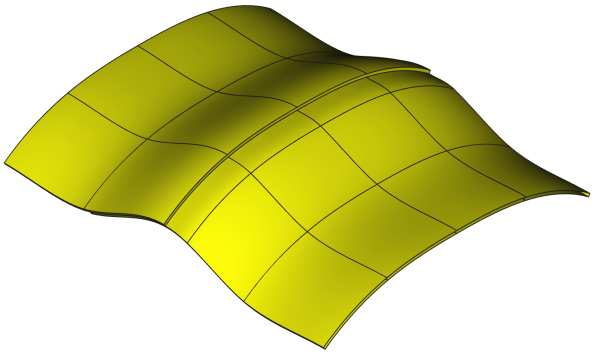
Wenn keine konkrete Übergangsform durch den User in Form einer festen Randbedingung gefordert wird, werden im Entwurfsrahmenwerk nach der Philosophie des Re-Engineerings alle $v_{geo} = 9$ Variationsmöglichkeiten[2] (siehe Abbildung 5.3) in den Entwurf eingebracht, die mit allen weiteren aus den verschiedenen Domänen resultierenden Varianten durchpermutiert werden müssen. In diesem Beispiel liegt der Fokus des Re-Engineerings auf der Variation der Übergangsform und der Verbindungstechnik. Bei den Verbindungstechniken werden mit dem Schweißen, Kleben, Nieten und Schrauben $v_{verb} = 4$ Verfahren berücksichtigt, was auf eine maximale Gesamtvariantenzahl im Beispiel von $v_{max} = v_{geo} \cdot v_{verb} = 9 \cdot 4 = 36$ führt.

In den Abbildungen 5.4 und 5.5 ist das in diesem Beispiel umgesetzte Umkonstruktionsschema schematisch aufbereitet. Einer besseren Übersichtlichkeit geschuldet, sind von den tatsächlich ausgelegten 36 Varianten nur die Konstruktionswege vierer Konstruktionen dargestellt.

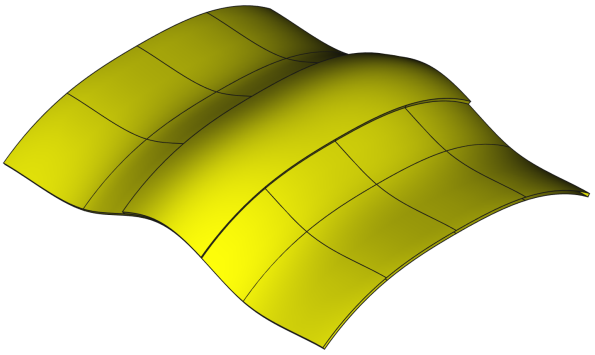
[2] Zusätzliche durch den Anwender spezifizierte Übergangsformen werden selbstverständlich ebenfalls berücksichtigt und erhöhen die genannte Anzahl.



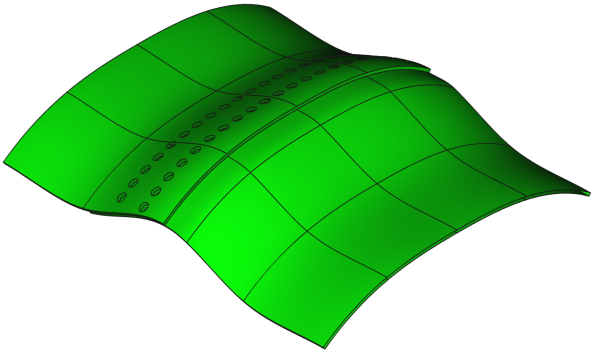
(a) Ungestörte Geometrie



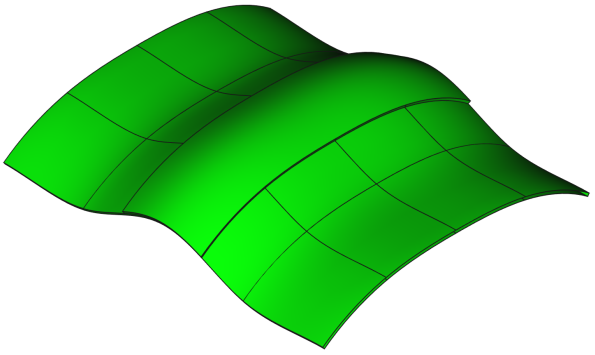
(b_1) Umkonstruierte Geometrie mit Stoßart „Joggle Lap“



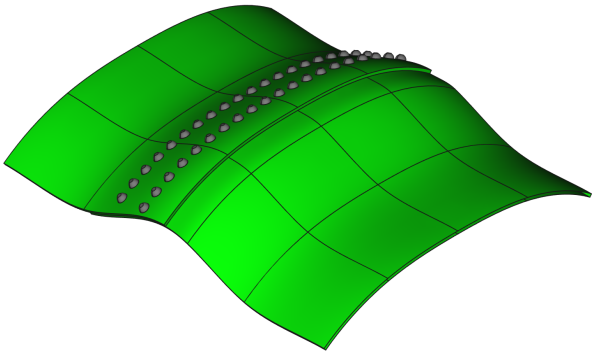
(b_2) Umkonstruierte Geometrie mit Stoßart „Strap“



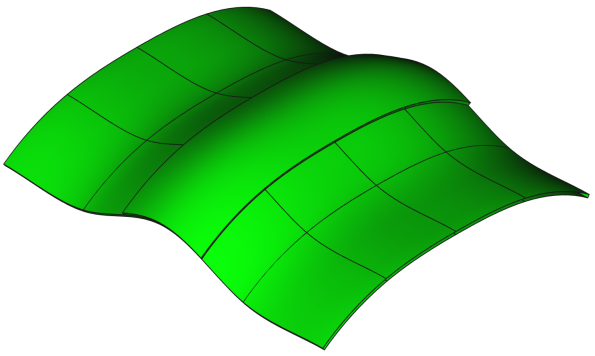
(c_1) Bohrungen in Joggle Lap-Joint



(c_2) Keine weitere Vorbereitung des Strap-Joints notwendig

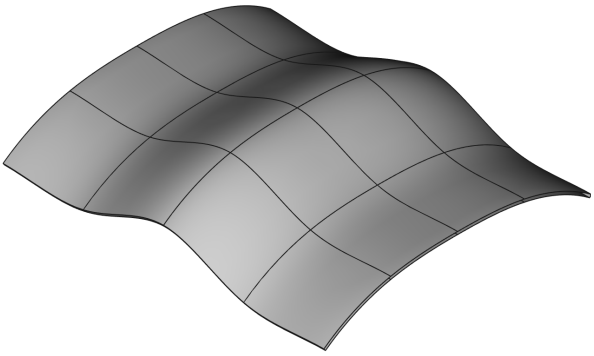


(d_1) Joggle Lap-Joint mit Bohrungen und Nieten

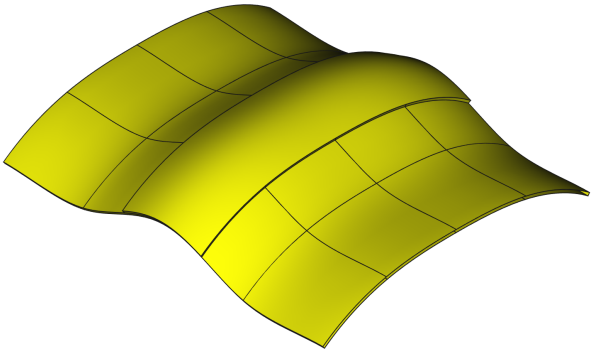


(d_2) Strap-Joint geklebt

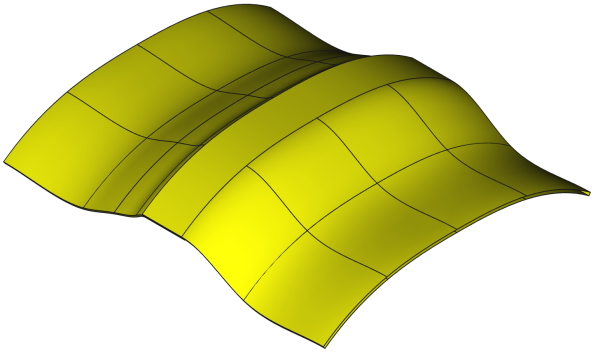
Abbildung 5.4: Umkonstruktionsschema mit Joggle Lap und Strap als Stoßart und Schrauben und Kleben als Verbindungstechnik



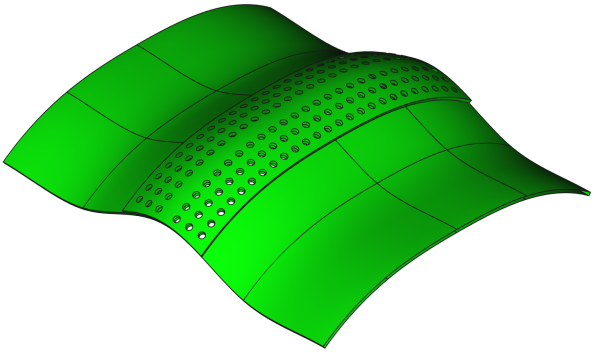
(a) Ungestörte Geometrie



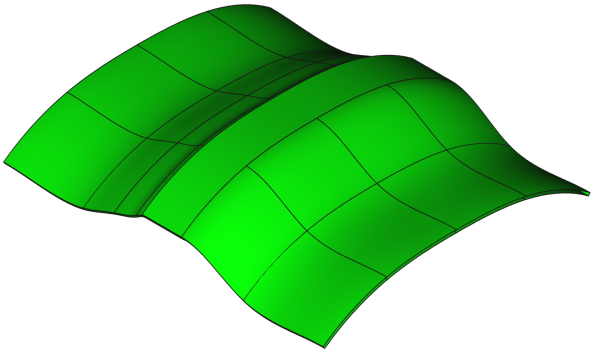
(b_1) Umkonstruierte Geometrie mit Stoßart „Strap“



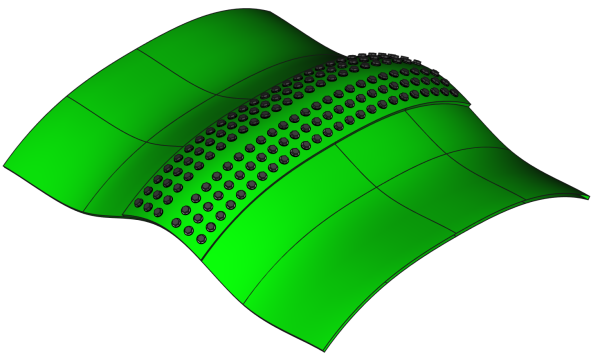
(b_2) Umkonstruierte Geometrie mit Stoßart „Joggle Lap straight mid part“



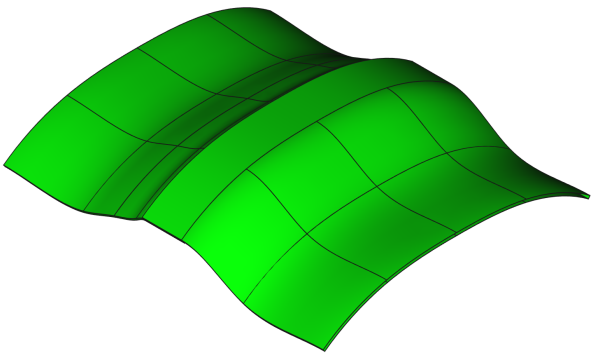
(c_1) Bohrungen in Strap-Joint



(c_2) Keine weitere Vorbereitung des Joggle Lap straight mid part-Joints notwendig



(d_1) Strap mit Bohrungen und Schrauben



(d_2) Geschweißter Joggle Lap straight mid part

Abbildung 5.5: Umkonstruktionsschema mit Strap und Joggle Lap straight mid part als Stoßart und Schrauben und Schweißen als Verbindungstechnik

Ausgehend von der ungestörten Grundgeometrie (Abbildung 5.4 bzw. 5.5 a) werden jeweils zwei mögliche Stoßgeometrien gezeigt (Abbildung 5.4 bzw. 5.5 b_1 – b_2). Auf diese Stoßgeometrien werden mögliche Verbindungstechniken angewendet (Abbildung 5.4 bzw. 5.5 c_1 – c_2) und etwaige benötigte Verbindungselemente ergänzt (Abbildung 5.4 bzw. 5.5 d_1 – d_2). Von den oben bereits erwähnten maximal möglichen 36 Varianten sind nicht alle Kombinationen valide Konstruktionslösungen. Beispielweise verlangen alle mechanischen Verbindungsverfahren eine Überlappungs- oder Laschenübergangsform, sodass Kombinationen mit einem Stumpfstoß ausgeschlossen sind. Dies ist in den erzeugenden Regeln kodiert und bedeutet im Umkehrschluss, dass die nicht möglichen Kombinationsvarianten in diesem Fall erst gar nicht generiert werden. Die entsprechenden Klassen müssen dazu so modelliert werden, dass ihnen bekannt ist, welche Kombinationen mit anderen Domäneneigenschaften verträglich sind. Ist dieses Wissen nicht a priori kodiert, würden potenziell invalide Varianten zunächst trotzdem generiert, könnten aber anschließend durch eine Metrik oder den auswertenden Ingenieur ausgeschlossen werden.

Die Anzahl ausgeschlossener Varianten wird, analog zur Regelsequenz, erst zur Laufzeit dynamisch aus dem Entwurfskontext heraus abgeleitet und kann im Allgemeinen nicht im Vorhinein bestimmt werden. Im vorliegenden Beispiel ergeben sich $v_{ges} = 34$ valide Varianten.

5.2 Satellitengehäuse

In diesem Anwendungsbeispiel soll die Funktionsweise des Entwurfsrahmenwerks im Kontext des Reverse-Engineering (siehe Abschnitt 3.1.2) demonstriert und der praktische Nachweis der Existenz der in Abschnitt 3.1.2 beschriebenen inversen Transformation erbracht werden. Hierfür muss das zugehörige Geometriemodell mit einem Mustererkennungsmechanismus analysiert und daraus ein abstraktes Modell des Satellitengehäuses rekonstruiert werden. Dieses Anwendungsbeispiel wurde in [Schopper und Rudolph, 2018] in englischer Sprache bereits veröffentlicht. Das Geometriemodell des Satelliten ist in Abbildung 5.6 gezeigt.

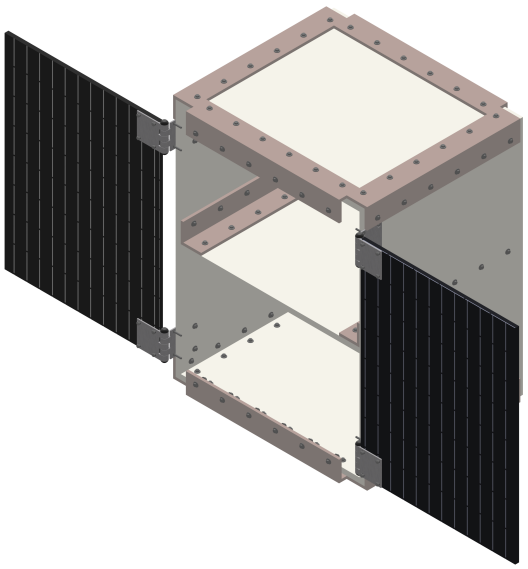


Abbildung 5.6: Satellitengehäuse (vordere Platte ausgeblendet) [Schopper und Rudolph, 2018]

Das Satellitengehäuse stellt sich als eine zusammengesetzte Struktur aus sieben Aluminiumplatten (vorne, hinten, oben, unten, links, rechts und Mitte) dar. Diese sind mit zehn identischen Winkelelementen und jeweils 12 Schrauben pro Winkel zu einer Gehäusestruktur verbunden. Des Weiteren sind an den Außenseiten zwei Solarpaneele angebracht, die jeweils mit zwei Scharnieren mit dem Gehäuse verbunden sind. Sowohl auf der Gehäuse- als auch auf der Paneelseite sind die Scharniere mit jeweils vier Schrauben befestigt. Für den Modellrekonstruktionsprozess wird davon ausgegangen, dass die entsprechende CAD-Datei im Standard for the Exchange of Product Model Data (STEP)-Format[3] vorliegt, darin alle beschriebenen Elemente separat repräsentiert sind und nicht als ein einziges „verschmolzenes“ Bauteil.

Um ein abstraktes Modell aus einem konkreten Geometriemodell in Form einer CAD-Datei zu rekonstruieren, müssen vor dem Hintergrund verschiedener Basisontologien aus den unterschiedlichen Bereichen, aus denen das Modell zusammengesetzt sein könnte, fehlende Modellinformationen in einem Mustererkennungsprozess erschlossen und dem Modell hinzugefügt werden (vgl. Abschnitt 3.1.2). Da es sich beim Satelliten in diesem Anwendungsbeispiel um eine aus Einzelteilen zusammengefügte Baugruppe handelt, muss beispielsweise Wissen über Fügeprozesse vorgehalten werden, um das abstrakte Modell abzuleiten. Die in diesem Zusammenhang benötigte Basisontologie wurde mithilfe von FCA erstellt und als Klassendiagramm implementiert (siehe Abbildung 4.7). In Abbildung 5.7 ist das Unterklassendiagramm mit der Erweiterung der Klasse **Fastening** aus Abbildung 4.7 dargestellt.

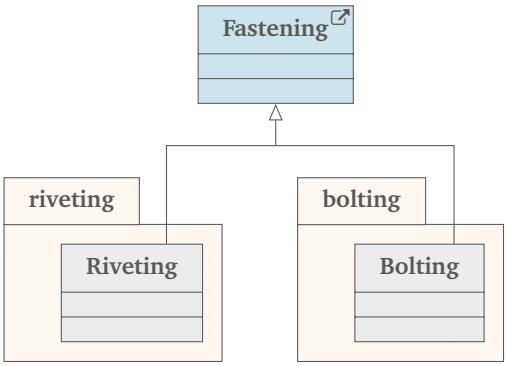


Abbildung 5.7: Klassendiagramm für die Modellierung mechanischer Verbindungstechniken

Die beiden die Klasse **Fastening** erweiternden Klassen **Riveting** und **Bolting** repräsentieren die kraftschlüssigen Verbindungsverfahren, die in diesem Anwendungsbeispiel angewendet werden können. Das obige Unterklassendiagramm ist um weitere Verbindungsverfahren, die aktuell dem Stand der Technik entsprechen oder gegebenenfalls erst in Zukunft erfunden werden, erweiterbar. Gemäß der Modellierung in Abbildung 4.2 hat jedes Verbindungsverfahren potenziell ein oder mehrere Verbindungselemente (Assoziation **connectingElement**). Die zu den beiden Verfahren passenden Verbindungselemente können gemäß dem in Abbildung 4.8 abgebildeten Klassendiagramm wiederum Nieten (**Rivet**) oder Schrauben (**Bolt**) sein. Es existiert eine schier unendliche Anzahl von Spezialisierungen dieser beiden Klassen, die unter anderem in diversen

[3] Mehr Informationen zum Austauschformat STEP finden sich in [Anderl und Trippner, 2000].

DIN- und ISO-Normen definiert und festgehalten sind. Exemplarisch sind in Abbildung 5.8 einige Beispiele für die implementierten Unterklassen von **Bolt** dargestellt.

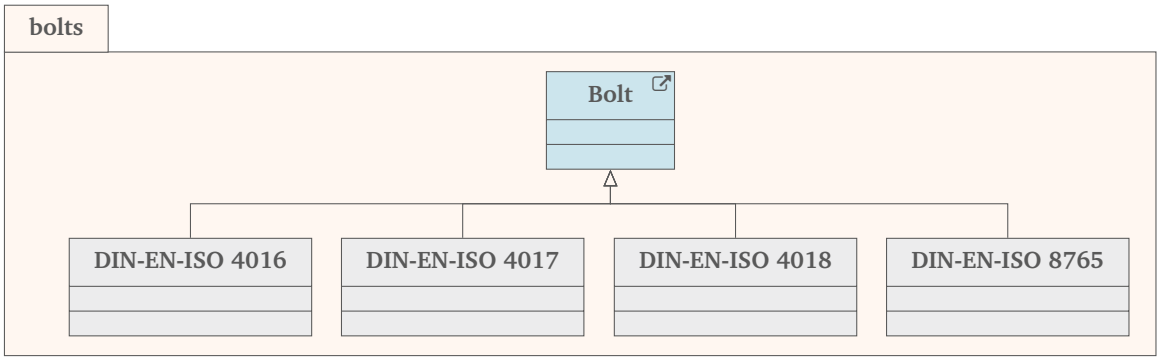


Abbildung 5.8: Unterklassendiagramm zu Abbildung 4.8 für Schrauben

Dieses Klassendiagramm stellt die unterste Ebene der Verbindungselementedomäne in der Fallstudie dar. Alle geometriebehafteten Objekte auf dieser Ebene enthalten in den Klassendiagrammen als Eigenschaften die formbezogenen Π- und ψ-Zentralmomente[4] (siehe Abschnitt 3.1.2).

Zusammengefasst wird in Anlehnung an die FCA das Konzept der Verbindungselemente (und seiner Spezialisierungen) auf eine Menge von Klassen (Bezeichnungen) und eine Menge gemeinsamer Eigenschaften bezogen (z. B. geometrische Zentralmomente mit konkreten Werten). Infolgedessen repräsentieren Klassen auf Objektebene den zur Verfügung stehenden Teilebaukasten und sind somit die Grundlage für das nachfolgend beschriebene Klassifizierungsschema. Wichtig ist hervorzuheben, dass Objekte außerhalb dieses Baukastens bei der Mustererkennung weder erkannt noch in einem späteren Musterersetzungsverfahren variiert werden können.

Klassifikationsschema

Im Gegensatz zur Softwareentwicklung mündet ein Konstruktionsprozess im Engineering in einem realen Produkt, das meistens aus Einzelteilen zu einer Baugruppe zusammengesetzt ist. Für die Verbindung der Einzelteile ist der Einsatz von Fügeverfahren unabdinglich. Jedes einzelne Teil innerhalb einer Baugruppe besitzt eine geometrische Repräsentation, wodurch sich ganz allgemein eine Kopplung zwischen Geometrie und Bezeichnung[5] herleiten lässt. In Abbildung 5.9 sind einige Unterklassen von **FunctionalPrimitve** aus Abbildung 4.4 abgebildet.

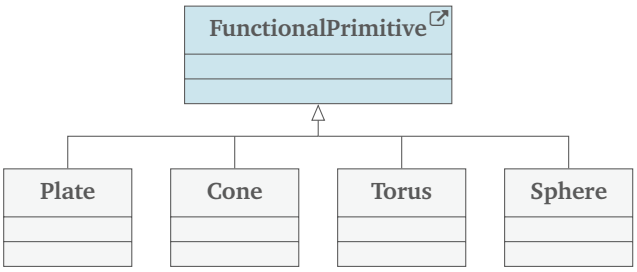


Abbildung 5.9: Unterklassendiagramm zu Abbildung 4.4 für **FunctionalPrimitve**

[4]Die jeweils 19 Parameter pro Zentralmoment sind in Abbildung 5.8 nicht mit dargestellt.
[5]In diesem Fall ist die Bezeichnung der Klassenname.

Analog zu Abbildung 5.8 wird im obigen Klassendiagramm eine konkrete Zuordnung zwischen Form und Bezeichnung modelliert: Eine Bauteilklasse mit konkreter Bezeichnung (z. B. **Plate**) erhält über die gesetzten Eigenschaften eine geometrische Zuordnung über die geometrischen Zentralmomente (z. B. 19 Π- und 19 ψ-Werte).

Beim Reverse-Engineering, einer dem Entwurfsrahmenwerk initial unbekannten Baugruppe, ist die Zuordnung der Einzelteile zunächst unbekannt und muss erst aus dem geometrischen Modellkontext abgeleitet werden. Dieser Vorgang wird im Rahmen dieser Arbeit als Mustererkennungsproblem verstanden (vgl. Abschnitt 3.1.2). Das CAD-Modell in Form einer STEP-Datei, die im Wesentlichen ein Textdokument mit standardisierter Syntax darstellt, muss in ein abstraktes Modell des Satelliten umgewandelt werden. Philosophisch betrachtet wird bei diesem Vorgang eine T2M-Transformation (siehe Abschnitt 3.1.2) umgesetzt. Wie bereits an anderer Stelle erwähnt, ist es evident, dass eine T2M-Transformation das Modell mit zusätzlicher Information anreichern muss. Diese Zusatzinformation kann nicht einfach „aus dem Nichts“ erzeugt werden, sondern muss aus dem vorhandenen – hier geometrischen – Kontext vor dem Hintergrund eines Weltmodells mit Basiswissen *erschlossen* werden. Im Rahmen dieses Anwendungsbeispiels handelt es sich bei diesem Basiswissen um die objektorientierten Domänenklassendiagramme und die dort kodierten Zusammenhänge.

Die Geometrieklassifizierung basiert im Kern auf einem Koeffizientenvergleich. Für die aus dem STEP-File des Satelliten extrahierten Einzelteile werden die μ- und Π-Zentralmomente nach [Kaiser und Rudolph, 2018] berechnet. Die Π-Werte der noch unbekannten Objekte werden dann mit allen in den Domänenklassendiagrammen auftretenden Klassen mit hinterlegten Π-Werten – der Bauteilbibliothek des Entwurfsrahmenwerks – verglichen. Wenn eine Klasse in allen Werten mit einem berechneten Objekt innerhalb einer ϵ-Umgebung[6] übereinstimmt, ist eine Zuordnung gefunden. Dieser Vorgang wird für alle Teile durchlaufen und anschließend ein erstes Modell abgeleitet[7]. Dieses aus dem geometrischen Kontext geschlossene abstrakte Modell für das Satellitengehäuse ist in Abbildung 5.10 als Klassendiagramm gezeigt.

Abbildung 5.10: Erstes abgeleitetes, abstraktes Modell des Satelliten nach der Klassifizierung

Gemäß dem abgeleiteten Klassendiagramm umfasst die betrachtete Baugruppe vier verschiedenen Klassen von Bauteilen: Winkelelemente (**Corner Profile**), Scharniere (**Hinge**), Schrauben (**Bolt**) und Platten (**Plate**). Über die reine Klassifizierung hinaus ist es ebenfalls möglich, die Anzahl im Modell vorhandener Instanzen der gefundenen Klassen anzugeben. In der Fallstudie

[6] Der Wert von ϵ ist im Entwurfsrahmenwerk standardmäßig auf 10^{-4} festgelegt, kann aber vom Anwender auf einen anderen Wert gesetzt werden. Für eine tiefergehende Auseinandersetzung mit der Auswirkung der Wahl verschiedener Werte von ϵ sei auf [Kaiser und Rudolph, 2018] verwiesen.

[7] In diesem Anwendungsbeispiel wird vorausgesetzt, dass sich die zu untersuchende STEP-Datei nur aus dem Entwurfsrahmenwerk bereits bekannten Geometrien zusammensetzt, die durch den Mustererkennungsprozess schließlich erkannt werden können. Andernfalls müsste eine algorithmische Bauteilzerlegung erfolgen, die einer vorgegebenen Heuristik folgt, um zumindest einen Teil der Geometrie in bekannte Einzelentitäten zu überführen. Diese Zerlegung bietet einen guten Anknüpfungspunkt für zukünftige Forschungsaktivitäten.

handelt es sich dabei um neun Instanzen der Klasse **Plate**, vier Instanzen der Klasse **Hinge**, 10 Instanzen der Klasse **Corner Profile** und 152 Instanzen der Klasse **Bolt**. Ein interessanter Nebeneffekt ist die Äquivalenz dieses generierten Modells mit der Stückliste (Bill of Materials (BOM)) der Baugruppe. Die vorgestellte Methodik ist somit ebenfalls dafür anwendbar, die Stückliste einer Baugruppe zu bestimmen oder zu beweisen. Insbesondere im Kontext des vollautomatisierten Entwurfs ist der Beweis der BOM für eine automatisierte Zertifizierung ein nicht zu unterschätzender Gewinn. Voraussetzung hierfür ist allerdings, dass jedes einzelne Teil eine Repräsentation in der Bauteilbibliothek besitzt[8] [9].

Da die erkannten Klassen selbst aus den verschiedenen im Framework implementierten Domänenklassendiagrammen stammen (siehe Abbildungen 4.2, 4.4 bis 4.8, 5.7 und 5.8), können die dort hinterlegten Abhängigkeiten teilweise auf die Klassen in Abbildung 5.10 übertragen werden. Außerdem können weitere Klassen aus der Existenz spezieller Klassen im abstrakten Modell geschlossen werden. Dies basiert auf der Tatsache, dass manche Assoziationen in den Domänenklassendiagrammen als Komposition[10] markiert sind. Aus diesen Existenzbedingungen lässt sich dann das Klassendiagramm für die unbekannte Baugruppe mit zusätzlicher Semantik anreichern. Dieses Klassendiagramm ist in Abbildung 5.11 zu sehen.

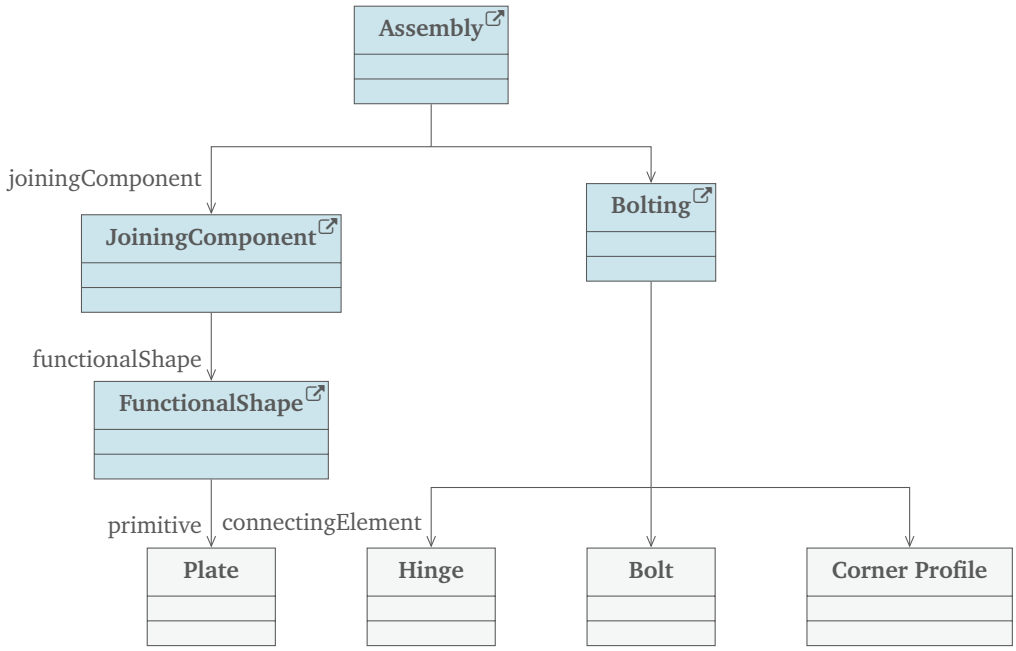


Abbildung 5.11: Resultierendes Klassendiagramm nach Klassifikation und Inferenz

[8] Zu diesem Punkt gibt es seit längerer Zeit Ansätze im Umfeld des wissensbasierten Ingenieurentwurfs (siehe z. B. [Cybenko et al., 1996], [Sung et al., 2002], [Wei et al., 2010] oder [Gembarski et al., 2016]).

[9] Bei der Verwendung von GBES für den Systementwurf ist diese Voraussetzung immer automatisch erfüllt, da der Entwurf nur instanziierte Objekte von Klassen enthalten kann, die dem System bereits bekannt sind. Aus diesem Grund kann diese Eigenschaft auf mehrfache Weise zur Validierung des maschinell generierten Entwurfs eingesetzt werden.

[10] Unter einer Komposition im Sinne der UML versteht man eine Verbindung zwischen zwei Klassen, die eine Existenzabhängigkeit ausdrückt. In DC43 gibt es keine direkte Möglichkeit, Kompositionen im Klassendiagramm zu verwenden, allerdings kann beim Reverse-Engineering eine Assoziation als Komposition interpretiert werden, wenn dies entsprechend hinterlegt ist.

Die klassifizierten Scharniere, Schrauben und Winkel aus Abbildung 5.10 des Satelliten lassen sich durch Inferenz als **connectingElements** eines **JoininigProcesses** interpretieren. Das Vorhandensein eines Objekts vom Typ **Bolt** impliziert den speziellen Typ des Prozesses als **Bolting**. Die Platten werden als **FunctionalPrimitive** vom speziellen Typ **Plate** erkannt.

Aus der Existenz dieser Klassen kann gefolgert werden, dass im Modell die Klasse **FunctionalShape** und somit auch die Klassen **JoininigComponent** und **Assembly** vorhanden sein müssen. Daraus ergibt sich das in Abbildung 5.11 als Klassendiagramm dargestellte abstrakte Modell für die zuvor unbekannte Baugruppe.

Insgesamt kann das auf diese Weise abgeleitete Klassendiagramm als abstraktes Beschreibungsmuster für die analysierte Baugruppe betrachtet werden. Aus philosophischer Sicht korreliert ein solches Muster mit der Vorstellung einer topologischen Hülle für die Baugruppe (vgl. Kapitel 3). Die verknüpfte Fragestellung, wie dieses Muster ausgeprägt werden muss, um das konkretes Modell des Satellitengehäuses zurückzuerhalten und welche Werte die vorgehaltenen Parameter der Objekte haben müssen, ist damit aber noch nicht beantwortet. Dies ist eine zweite Art von Entwurfswissen und im Kontext graphenbasierter Entwurfssprachen für gewöhnlich in den Regeln eines Aktivitätsdiagramms abgebildet.

Im folgenden Abschnitt wird am Beispiel der Platte dargelegt, wie nach der in Abschnitt 3.1.2 beschriebenen Vorgehensweise die Parameter der Bauteilobjekte rekonstruiert werden können und so das rekonstruierte Modell weiter ausdetailliert werden kann.

Rekonstruktion von Modellparametern

Wie bereits mehrfach erwähnt, kann im Klassendiagramm zusätzliches Wissen in Form von Klasseneigenschaften hinterlegt werden. Als Beispiel können hier die 19 Π- und ψ- Zentralmomente genannt werden, die für die Klassifizierung von Platten in die zugehörige Klasse **Plate** geschrieben werden müssen. Grundvoraussetzung für dieses Vorgehen ist somit, dass die Bauteile einer Parametrisierung zugänglich sind. In Tabelle 5.1 sind die vorberechneten, symbolischen Zentralmomente der Platte aufgeführt.

Tabelle 5.1: 19 ψ -Zentralmomente für eine Platte mit Parametern $p_i = \{a, b, c\}$

$\psi_{000} = abc,$	$\psi_{020} = \frac{ab^3c}{12},$	$\psi_{102} = 0,$	$\psi_{003} = 0,$
$\psi_{100} = 0,$	$\psi_{002} = \frac{a\,b\,c^3}{12},$	$\psi_{021} = 0,$	$\psi_{400} = \frac{a^5\,b\,c}{80},$
$\psi_{010} = 0,$	$\psi_{210} = 0,$	$\psi_{012} = 0,$	$\psi_{040} = \frac{a\,b^5\,c}{80},$
$\psi_{001} = 0,$	$\psi_{201} = 0,$	$\psi_{300} = 0,$	$\psi_{004} = \frac{a\,b\,c^5}{80}$
$\psi_{200} = \frac{a^3bc}{12},$	$\psi_{120} = 0,$	$\psi_{030} = 0,$	

Als Nächstes werden die μ-Zentralmomente der Platte bestimmt. Diese werden nach der in [Kaiser und Rudolph, 2018] beschriebenen Methode numerisch berechnet. Bei dieser Vorgehensweise werden die Bauteile zunächst mit Tetraedern vernetzt und anschließend die jeweiligen Zentralmomente aller Tetraeder zum Gesamtzentralmoment des untersuchten Körpers aufsummiert. Diese Berechnungsweise wird zur Bestimmung aller 19 Zentralmomente angewendet. Da die betrachtete Platte als Quader ein geometrisch simples Bauteil darstellt, ist es ebenfalls

möglich, die μ-Zentralmomente analytisch zu berechnen. Die Kantenlänge der Platte betragen $a = 740$, $b = 600$ und $c = 20$ mm. In Tabelle 5.2 sind die 19 numerisch bestimmten μ-Zentralmomente der Platte auf drei Nachkommastellen gerundet aufgelistet[11].

Tabelle 5.2: Berechnete μ-Zentralmomente für die Deckplatte des Satellitengehäuses

$\mu_{000} = 8.880 \times 10^{6}$	$\mu_{020} = 4.052 \times 10^{11}$	$\mu_{102} = 7.529 \times 10^{-7}$	$\mu_{003} = -5.172 \times 10^{-6}$
$\mu_{100} = -2.587 \times 10^{-7}$	$\mu_{002} = 2.960 \times 10^{8}$	$\mu_{021} = -0.002$	$\mu_{400} = 1.438 \times 10^{16}$
$\mu_{010} = 1.949 \times 10^{-7}$	$\mu_{210} = 0.006$	$\mu_{012} = -3.619 \times 10^{-6}$	$\mu_{040} = 3.328 \times 10^{16}$
$\mu_{001} = -4.625 \times 10^{-8}$	$\mu_{201} = -0.001$	$\mu_{300} = 0.002$	$\mu_{004} = 1.775 \times 10^{10}$
$\mu_{200} = 2.664 \times 10^{11}$	$\mu_{120} = 0.026$	$\mu_{030} = 0.016$	

Für symmetrische Bauteile gilt in voller Allgemeinheit, dass sich alle Zentralmomente mit mindestens einem ungeraden Eintrag in einem pqr-Tripel zu null ergeben. Somit offenbart sich die Ungenauigkeit der numerischen Methode zur Bestimmung der μ-Werte direkt aus dem Vergleich der entsprechenden Einträge in Tabelle 5.1 und Tabelle 5.2 (z. B. $\mu_{030} = 0.016 \overset{!}{>} 0$). Aufgrund der beschreibenden Parameterzahl von drei bei der in diesem Beispiel betrachteten Platte muss für die Rekonstruktion der Modellparameter ein Koeffizientenvergleich dreier unabhängiger Zentralmomente aus Tabelle 5.2 mit den entsprechenden parametrischen Zentralmomenten aus Tabelle 5.1 durchgeführt werden[12]. Dadurch ergibt sich beispielsweise mit dem pqr-Satz $\{(000), (200), (020)\}$ das folgende Gleichungssystem

$$\begin{aligned} (000): &\quad abc = 8.880 \times 10^{6} \\ (200): &\quad \tfrac{a^3bc}{12} = 2.664 \times 10^{11} \\ (020): &\quad \tfrac{ab^3c}{12} = 4.052 \times 10^{11} \end{aligned} \tag{5.1}$$

Dieses Gleichungssystem kann mit einem geeigneten numerischen Verfahren gelöst werden[13]. Als Lösung ergeben sich die Werte $a = 739.978$, $b = 600.001$ und $c = 20.006$, die in guter Näherung den originalen Parametern der Platte entsprechen[14]. In Abbildung 5.12 ist das gesamte Vorgehen der Parameterrekonstruktion aus noch einmal in kompakter Form zusammengefasst. Völlig analog zum obigen Vorgehen können die Parameterwerte aller weiteren Bauteile des Satelliten ebenfalls rekonstruiert werden. Zur Vermeidung redundanter Passagen soll aber auf deren explizite Herleitung und Darstellung an dieser Stelle verzichtet werden.

[11] Der geneigte Leser ist dazu aufgerufen, die Ergebnisse mithilfe von Gleichung 3 in [Kaiser und Rudolph, 2018] oder direkt mit den in Tabelle 5.1 gezeigten analytischen Ausdrücken nachzurechnen.

[12] Die sich zu null ergebenden Zentralmomente eigenen sich prinzipiell nicht für die vorgestellte Systematik, da mit Ihnen keine Parameter bestimmt werden können. Da Symmetrien oder symmetrische Anteile in Baugruppen durchaus üblich sind, könnte diese Erkenntnis bei der Wahl der pqr-Tripel in zukünftigen Implementierungen berücksichtigt werden und auf Zentralmomente mit ungeraden Einträgen gänzlich verzichtet werden.

[13] In der im Rahmen der vorliegenden Arbeit erarbeiteten Implementierung wird für diesen Zweck ein *Newton-Raphson*-Algorithmus zum Einsatz gebracht. Dieser hat sich als schnell und mit einem adäquaten Startlösungsschätzer als robust erwiesen. In zukünftigen Implementierungen könnte an dieser Stelle aber auch ein Verfahren, das ohne Startlösung auskommt, verwendet werden.

[14] Die Abweichungen zwischen den originalen und rekonstruierten Parametern geht zu einem großen Teil auf das eingesetzte – i. d. R. numerische und somit niemals exakte – Lösungsverfahren zurück. Eine Aussage über die Güte der Mustererkennung lässt sich daraus nicht ableiten, da die Mustererkennung in einem vorgelagerten Schritt auf Basis der II-Zentralmomente erfolgt (siehe hierzu [Melan, 2003] und [Kaiser und Rudolph, 2018]).

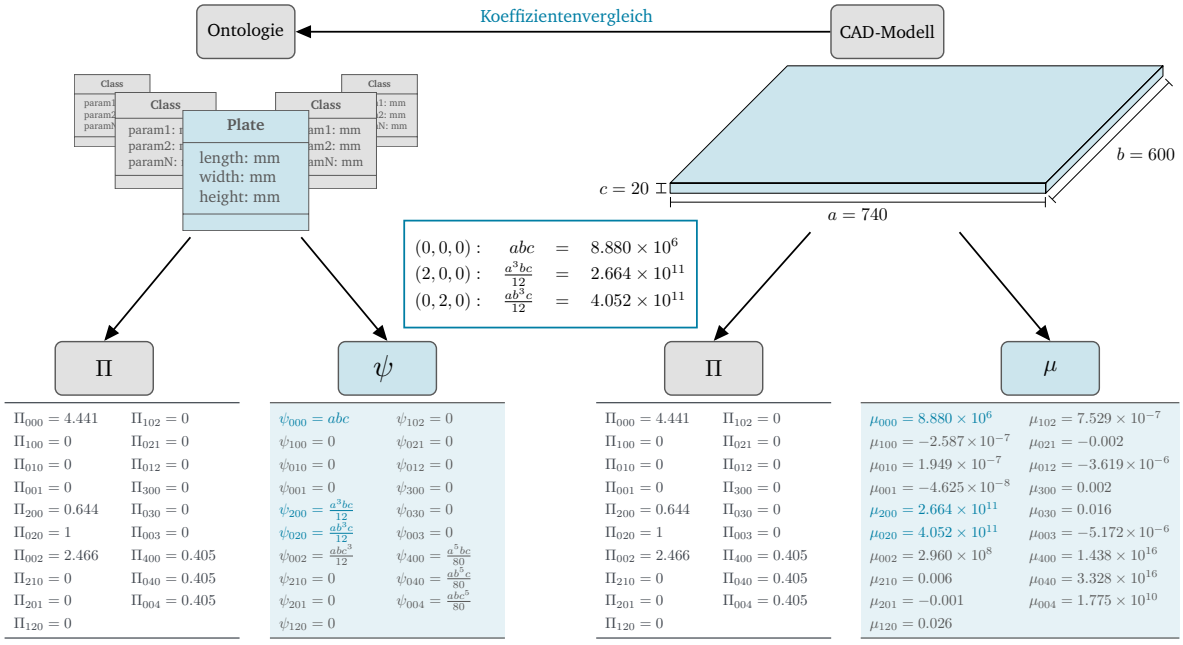


Abbildung 5.12: Parameterrekonstruktionsschema auf Basis geometrischer Zentralmomente

Modellgenerierung des Satellitengehäuses

Bisher wurde gezeigt, wie ein Modell des Satellitengehäuses auf Klassendiagrammebene, die Anzahl und die Parameter der instanziierten Objekte abgeleitet werden können. Das gesuchte Modell entspricht aber dem Entwurfsgraphen, der bei der Ausführung der Regeln im Produktionssystem durch den Entwurfscompiler entsteht (siehe Abschnitt 2.2.4).

Beim Round-Trip-Engineering werden alle Regeln in einem mehrfach iterativen Prozess viele Male durchlaufen. Ist das gesuchte Muster oder auch nur ein Teil dieses Musters Bestandteil des Konditionalteils einer Regel, so wird diese Regel ausgeführt. Durch diesen Mechanismus kann der gesuchte Entwurfsgraph zumindest zum Teil generiert werden. Wie bereits beschrieben wurde, ist es unter den genannten Voraussetzungen möglich, die Art und Anzahl der Objekte aus dem analysierten STEP-File abzuleiten. Mit diesem Wissen können die korrespondierenden Graphenknoten mitsamt ihren Eigenschaften leicht generiert werden.

Jene Knoten, die keine geometrischen Teile des Modells repräsentieren[15] und die Kanten zwischen ihnen sind eine größere Herausforderung. Einige der abstrakten Knoten können aus der Existenz anderer Knoten abgeleitet werden, dies ist aber nicht allgemein für alle Objekte möglich. Nichtsdestoweniger können die möglichen Links im Zielmodell auf die im abgeleiteten Klassendiagramm (siehe Abbildung 5.11) zulässigen Assoziationen beschränkt werden. Wenn das abgeleitete Zielklassendiagramm hinreichend differenziert ist, ist dies ein großer Vorteil, da dies die potenziellen Möglichkeiten stark einschränkt.

Für die genaue Zuordnung der einzelnen Bauteile zu ihren Partnern müssen zusätzliche geometrische Überlegungen unter Berücksichtigung der relativen Positionen der Teile untereinander angestellt werden. Dies geht über den Anwendungsbereich dieses Fallbeispiels hinaus

[15] Darunter können beispielsweise strukturierende oder funktionale Modellelemente fallen.

und ist Anknüpfungspunkt für weitere Forschung im Bereich des Reverse-Engineering.

In Abbildung 5.13 ist der aus dem geometrischen Initialmodell geschlussfolgerte Entwurfsgraph abgebildet, der als Resultat aus dem Reverse-Engineering Prozess hervorgegangen ist.

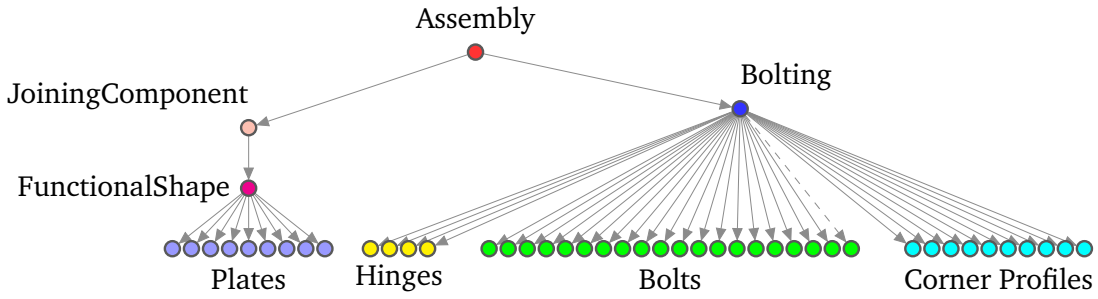


Abbildung 5.13: Resultierender Graph nach Ausführung der Entwurfssprache

Es sei darauf hingewiesen, dass für eine bessere Übersichtlichkeit nicht alle Schraubenobjekte dargestellt sind. Dies ist durch die gestrichelte Verlinkung am rechten Rand der Schrauben im Schaubild angedeutet. Von diesem Punkt aus könnte nun ein Re-Engineering initiiert werden, indem beispielsweise die geschraubte in eine genietete Bauweise überführt wird.

5.3 Hubschrauberstrukturelemente

Die in diesem Beispiel vorgestellte Konstruktion von Strukturelementen in einer Helikopterzelle ist ein Use Case innerhalb des vom Bundesministerium für Wirtschaft und Klimaschutz (BMWK) geförderten Forschungsprojekts COBAIN – Kompakte Hubschrauberstrukturen der nächsten Generation. In Zusammenarbeit mit Airbus Helicopters und weiteren Partnern[16] wurde das Entwurfsrahmenwerk für die Konstruktion eines Rumpfspantes eingesetzt. Mit diesem Beispiel wird unterstrichen, dass die Anwendbarkeit deutlich über den Bereich des Vorentwurfs hinausgeht. Im folgenden Unterkapitel wird zunächst die Ableitung eines Rumpfspantes aus gegebenen Randbedingungen dargelegt. Anschließend wird in Abschnitt 5.3.2 auf die Ableitung von Ausschnittgeometrien wie Türen oder Fenster eingegangen. Wie auch im Beispiel der Blechbauteile wird die Ableitung der Konstruktion aus einer zunächst ungestörten Anfangsgeometrie motiviert, die mittels „elektronischem Messer“ der Einbringung zusätzlicher Randbedingungen zugänglich gemacht wird. Diese Vorgehensweise ist auch in diesem Fall schlüssig, da die äußere Form eine vom Design und der Aerodynamik vorgegebene, feste Entwurfsgröße darstellt.

5.3.1 Rumpfspant

Für die automatisierte Konstruktion des Rumpfspantes müssen zunächst alle gewünschten Randbedingungen spezifiziert werden. Dies betrifft in diesem Beispiel insbesondere die Hüllgeometrien der Rumpfzelle und der Innenkabine des Helikopters, da zwischen diesen Flächen

[16] Unter anderem Ingenieurgesellschaft für intelligente Lösungen und Systeme mbH (IILS mbH), Deutsche Gesellschaft für Luft- und Raumfahrttechnik (DLR) und Fraunhofer-Institut für Gießerei-, Composite- und Verarbeitungstechnik (IGCV)

der für den Spant zur Verfügung stehende Bauraum begrenzt wird. Die beiden genannten Begrenzungsflächen werden zunächst im standardisierten Geometrieaustauschformat STEP in das Entwurfsrahmenwerk importiert. Um eine Variantenkonstruktion zu ermöglichen, muss die Drahtgitterstruktur, mit der das Framework arbeitet (siehe Abschnitte 3.2.2 und 4.2.3), zunächst über einen Reverse-Engineering-Prozess (siehe Abschnitt 3.1.2) erschlossen werden.

Wichtig ist hier zu erwähnen, dass in diesem Beispiel nicht die gesamte Geometrie einer Manipulation zugänglich gemacht werden muss, sondern nur der Bereich der Konstruktionszone des Spants. Hierfür wird in einem mehrstufigen Prozess mit dem „elektronischen Messer“ an einer definierten Stelle ein Segment aus der Zelle *herausgeschnitten* und dieses durch ein passendes, manipulierbares Drahtgittermodell *ersetzt*. In Abbildung 5.14 ist das Heraustrennen der Konstruktionszone aus der Zellgeometrie als erster Schritt schematisch gezeigt.

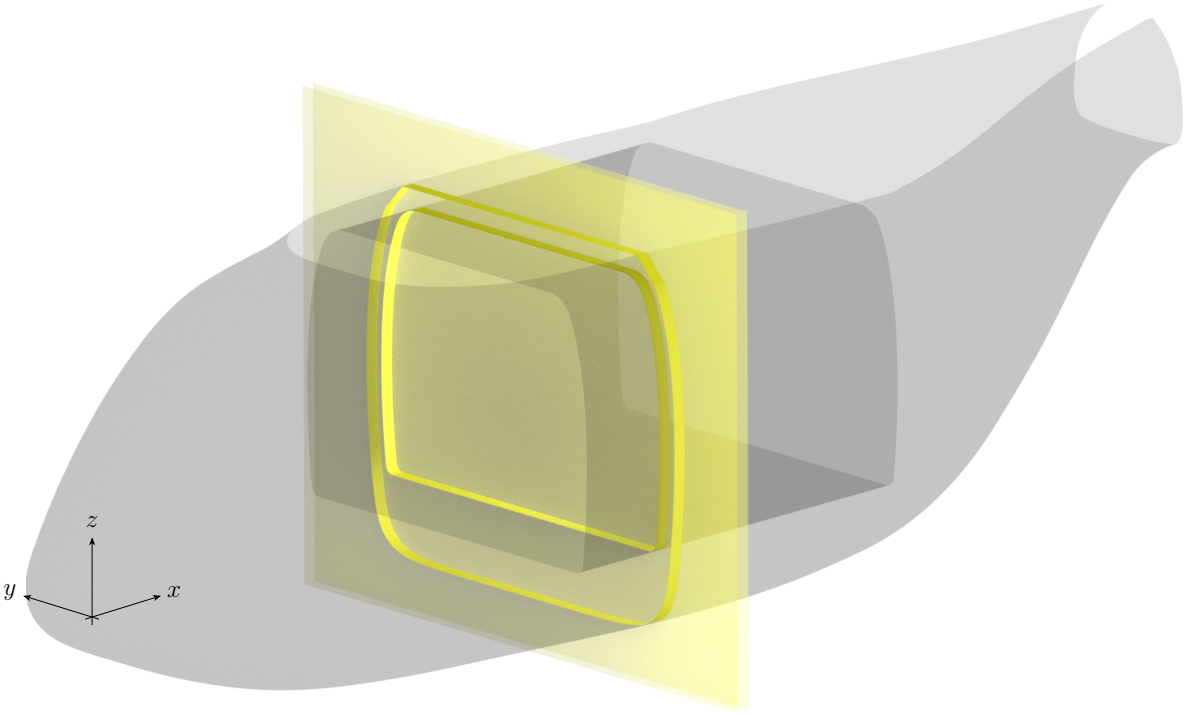


Abbildung 5.14: Heraustrennen der Konstruktionszone aus der Zellgeometrie des Hubschrauberspants mittels „elektronischem Messer“

Die Schnittebenen des „elektronischen Messers“ an der Vorder- und Hinterkante der Konstruktionszone sind in gelber Farbe hervorgehoben. An den resultierenden Schnittlinien mit der Ausgangsgeometrie ist – ebenfalls in Gelb – das herausgetrennte Rumpfsegment zu erkennen.

Nach dem Heraustrennen des Rumpfsegments gilt es als Nächstes, ein problemspezifisches Drahtnetzwerk in die Lücke einzubeschreiben. Hierfür werden wiederholt mit dem „elektronischen Messer“ Schnitte innerhalb der Konstruktionszone gesetzt, um zusätzliche Profilquerschnitte für das Drahtgitternetz abzuleiten. Die Positionen dieser Schnitte hängen einerseits davon ab, welche Form der zu konstruierende Spant später haben soll, andererseits aber auch vom jeweils gewählten Fertigungsverfahren. Anschaulich kann man sich das Drahtgitter als Hilfsliniennetzwerk vorstellen, das benötigt wird, um den Volumenkörper zu konstruieren.

Für die im nachfolgenden Schritt eingeführte „I“-Profilbauweise des Spants in diesem Beispiel ist lediglich ein zusätzliches Zwischenprofil in der Mitte der Konstruktionszone ausreichend. Andere Bauweisen, die ebenfalls noch vorgestellt werden, benötigten ein komplexeres, individuell angepasstes Drahtnetzwerk. Aus diesem Grund werden die Drahtgitternetze zur Laufzeit jedes Mal aufs Neue aus dem Problemkontext abgeleitet. Da dieser Vorgang insbesondere bei komplexeren Geometrien – wie in diesem Fall der Hubschrauberzelle – einen größeren Anteil an der Gesamtlaufzeit ausmachen kann, werden die Drahtgitternetze in der vorliegenden Implementierung zwischengespeichert, sodass sie bei ansonsten gleichbleibenden Randbedingungen in einem Folgelauf der Entwurfssprache nicht erneut generiert werden müssen.

Bleibt man zunächst beim einfachsten Anwendungsfall mit nur einem Zwischenprofil, ergeben sich somit zusammen mit den rechten und linken Randschnitten je drei Profilkurven an der Innen- und Außenseite. Hinzu kommen automatisch aus den Profilkurven abgeleitete Führungskurven, die jeweils an einem Krümmungswechsel der Profilkurven eingeführt und anschließend tangentenstetig zwischen den Profilen interpoliert werden. Von besonderer Bedeutung ist dabei, dass auf allen Profilkurven genau die gleiche Anzahl an Interpolationspunkten abgeleitet wird, sodass stets die Grundbedingung der Drahtgitterdefinition (vgl. Abschnitt 3.2.2) erfüllt ist. Die Führungskurven werden im Entwurfsrahmenwerk standardmäßig hinzugefügt und dienen im Allgemeinen der Möglichkeit einer noch feineren Ausdetaillierung der finalen Form des zu konstruierenden Bauteils. Im vorliegenden Beispiel spielen sie indes keine Rolle, da sie nicht gesondert manipuliert werden. Das resultierende Drahtgitter des zu konstruierenden Rumpfspants ist für den dreiprofiligen Fall in Abbildung 5.15 illustriert.

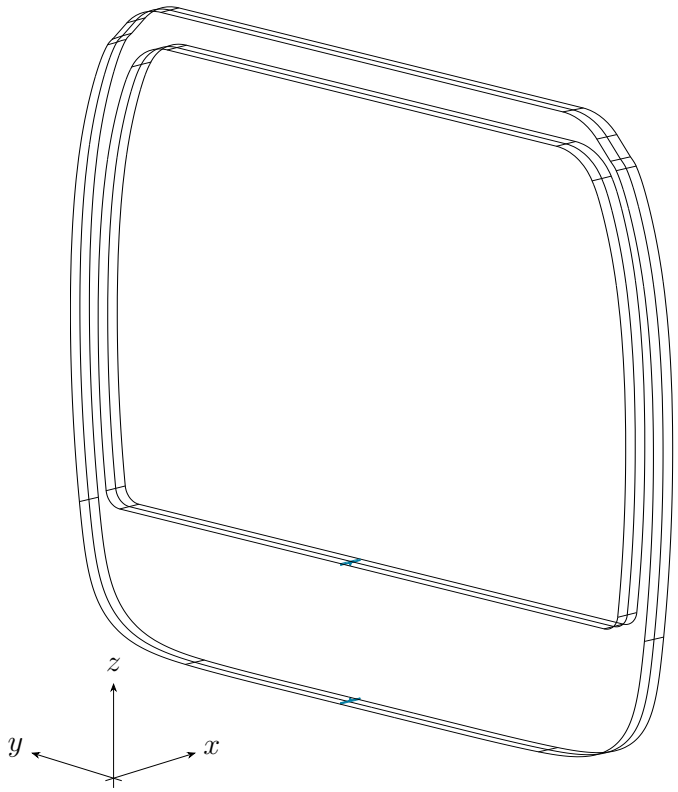
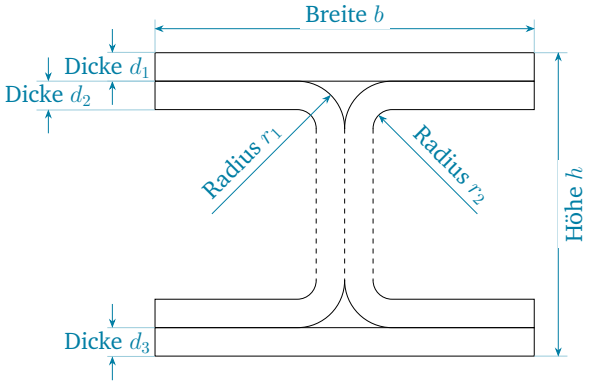


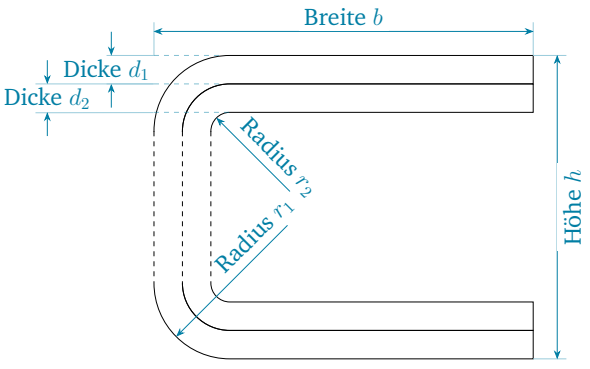
Abbildung 5.15: In die Konstruktionszone einbeschriebene Drahtgitternetzwerke

Wie bereits erwähnt, stellt das Drahtgitter die Ausgangsbasis für die Konstruktion des Volumenkörpers des Spants dar. In diesem Anwendungsbeispiel dient es gleichzeitig als Berandungs- wie auch als Führungskurve mehrerer *Sweep-* und *Extrusionsoperationen* mit unterschiedlichen geometrischen Profilen, die im Folgenden noch genauer vorgestellt werden. Als *Sweep* wird im Kontext des CAD ein Objekt bezeichnet, das beim Abfahren eines Profils entlang eines vorgegebenen Pfades entsteht. Sowohl das Profil als auch der Pfad können dabei eine beliebige, offene oder geschlossene Kurve oder Kurvensegmentfolge[17] darstellen. Die Extrusion kann als eine spezielle Form des Sweeps verstanden werden, bei dem der abgefahrene Pfad eine – in den meisten Fällen orthogonal auf das Basisprofil gefällte – Gerade darstellt. Eine äquivalente Bezeichnung ist die Parallelverschiebung. Im Unterschied zur generischen Sweep-Operation können aus der Extrusion somit nur zylindrische oder prismatische Körper resultieren.

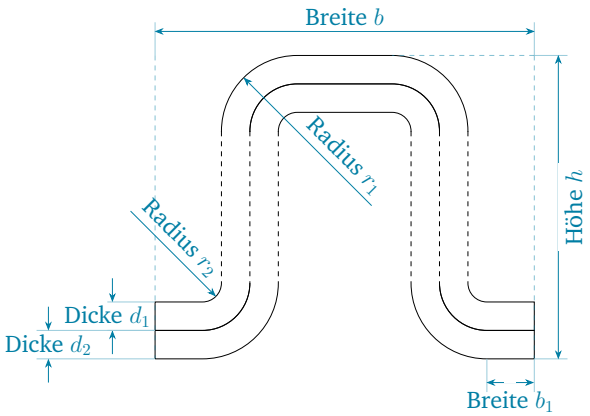
In diesem Beispiel soll zwischen drei im Helikopterentwurf typischerweise zum Einsatz gebrachten Profilformen für Strukturspante unterschieden werden. Das Profil manifestiert sich an einem gedachten Ebenenschnitt durch den Spant an einer beliebigen Position, variiert aber in der Höhe des mittleren Abschnitts je nach Schnittposition (vgl. Abbildung 5.15). Die verschiedenen Profiltypen sind inklusive ihrer Parametrisierungen in Abbildung 5.16 aufgeführt.



(a) Frameprofil „I“



(b) Frameprofil „C“



(c) Frameprofil „Ω“

Abbildung 5.16: Verschiedene Spantprofiltypen mit individuell adressierbarer Parametrik

[17] Die Kurvensegmentfolge wird in diesem Zusammenhang oft als *Wire* bezeichnet.

Wie aus der obigen Abbildung ersichtlich wird, unterscheiden sich die Profile bei identischen Außenabmessungen in Form und Parametrisierung. Das „I“-Profil (siehe Abbildung 5.16a) besteht aus einem oberen und einem unteren Gurt variabler Dicke (d_1 und d_3), aber gleicher Breite. Dazwischen befinden sich zwei symmetrisch orientierte „C“-Profile, die in ihrer Dicke (d_2) und den inneren und äußeren Radien (r_1 und r_2) adaptierbar sind. Das „C“-Profil (siehe Abbildung 5.16b) setzt sich aus einem inneren und einem äußeren „C“ zusammen, deren Dicken (d_1 und d_2) unabhängig voneinander gewählt werden können. Des Weiteren sind die Krümmungsradien r_1 und r_2 veränderbar. Das „Ω“-Profil (siehe Abbildung 5.16c) besteht wie das „C“-Profil aus einem inneren und einem äußeren Element und ist durch eine variable Fußbreite b_1, zwei Krümmungsradien (r_1 und r_2) sowie zwei Dicken (d_1 und d_2) parametrisiert.

In Abbildung 5.16 lassen sich alle abgebildeten Profile in drei Hauptbereiche unterteilen. Visualisiert sind diese Bereiche als ein durchgezogener oberer, ein gestrichelter mittlerer und ein durchgezogener unterer Teil. Mit dieser Darstellung soll herausgestellt werden, dass die verschiedenen Teilbereiche der Profile in unterschiedlichen Operationen eingesetzt werden, um das Volumenmodell des Spants zu generieren. Der obere, durchgezogene Teil der Profile wird entlang der entsprechenden äußeren umlaufenden Profillinie des zugehörigen Drahtgitters gesweeped, der untere analog entlang der entsprechenden inneren Profillinie.

Bei der Wahl der Profilparameter müssen die umlaufenden inneren und äußeren Randgitterkurven als äußere Begrenzungskurven berücksichtigt werden. Wurde durch den User ein maximaler Breitenparameter b_{max} spezifiziert, wird dieser Vorgabe bereits beim Herausschneiden der Konstruktionszone in der benötigten Breite Rechnung getragen. Damit sich als Resultat der Sweep-Operationen oben und unten je ein Volumenkörper (Solid) ergibt, müssen die Profile an der Übergangsstelle zum gestrichelten Bereich geschlossen sein, anderenfalls würde ein Oberflächenmodell entstehen. Dies geschieht im Entwurfsrahmenwerk automatisch.

Der in der Höhe variable mittlere Teil des Volumenkörpers – in Abbildung 5.16 gestrichelt angedeutet – entsteht durch je zwei Extrusionsvorgänge beim „I“- und „C“-Profil beziehungsweise aufgrund des doppelten Mittelteils vier Extrusionen beim „Ω“-Profil. Die pro Extrusiongebiet mittig liegenden inneren und äußeren Drahtgitterlinien müssen hierfür gegebenenfalls erst durch einen Schnitt mit dem „elektronischen Messer“ erzeugt und im zugehörigen Drahtgitter ergänzt werden[18]. Anhand von Abbildung 5.15 lässt sich dies leicht nachvollziehen. Die benötigten Führungslinien für das „I“-Profil sind bereits vorhanden, da die Extrusionszone hier genau mittig liegt. Für das „C“-Profil muss vor der Berandungskurve in positiver x-Richtung eine Schnittlinie im Drahtgitter ergänzt werden. Für das „Ω“-Profil müssen entsprechend zwei weitere Schnittkurven hinzugefügt werden – eine in positiver und eine in negativer x-Richtung–, wobei die Wahl des Parameters b_1 deren Lage im Drahtgitter determiniert.

Anschließend wird pro Konstruktionszone die von der inneren und äußeren Drahtgitterlinie begrenzte Fläche in positiver und negativer x-Richtung extrudiert (Dicke d_1 und d_2), wodurch der linke und rechte Volumenkörper des mittleren Teils des Spants erzeugt wird. Der innere Volumenkörper entsteht analog zum äußeren Bereich des Spants durch eine Sweep-Operation

[18] Mittig bedeutet in diesem Zusammenhang in einer Ebene parallel zur yz-Ebene, in der Mitte der Konstruktionszone. Innen und außen bezieht sich nach wie vor darauf, ob die jeweilige Schnittlinie durch einen Schnitt mit der inneren oder äußeren Bauraumbegrenzung hervorgegangen ist.

mit dem unteren Profilbereich in Abbildung 5.16 als Grundprofil entlang der entsprechenden inneren Führungslinien des zugehörigen Drahtgitters der Konstruktionszone. Die gesamte Vorgehensweise ist für die beschriebenen Profiltypen als Entwurfsmuster im Entwurfsrahmenwerk hinterlegt und kann somit direkt und problemlos auf ähnliche Anwendungsfälle – beispielsweise eine alternative Hubschrauberzellgeometrie oder gar ein Flugzeug – übertragen werden.

Nachdem erläutert wurde, wie die Geometrie des Spants durch das Entwurfsrahmenwerk konkret generiert wird, sollen nun noch exemplarisch einige Ergebnisse präsentiert werden. Die Position der Konstruktionszone kann von dem/der Anwender/in – selbstverständlich innerhalb der durch die initial eingelesene Geometrie gesetzten Grenzen[19] – frei festgelegt werden. Typischerweise besteht die Versteifung einer Helikopterzelle nicht nur aus einem, sondern aus mehreren hintereinander angeordneten Spantelementen, wie dies auch aus dem Flugzeug- oder Schiffsbau bekannt ist. Mit der hier vorgestellten Methodik lassen sich selbstverständlich auch in einem Entwurfslauf gleich mehrere Spanten an verschiedenen Stellen ausprägen, falls gewünscht auch in unterschiedlichen Bauweisen, Materialien und Herstellungsprozessen. Für die Veranschaulichung dieser Aussagen wurden im Rahmen dieses Anwendungsbeispiels exemplarisch drei verschiedene Spanten, an verschiedenen Stellen und in unterschiedlichen Profilbauweisen in die Zelle konstruiert. Das Ergebnis ist in Abbildung 5.17 abgebildet.

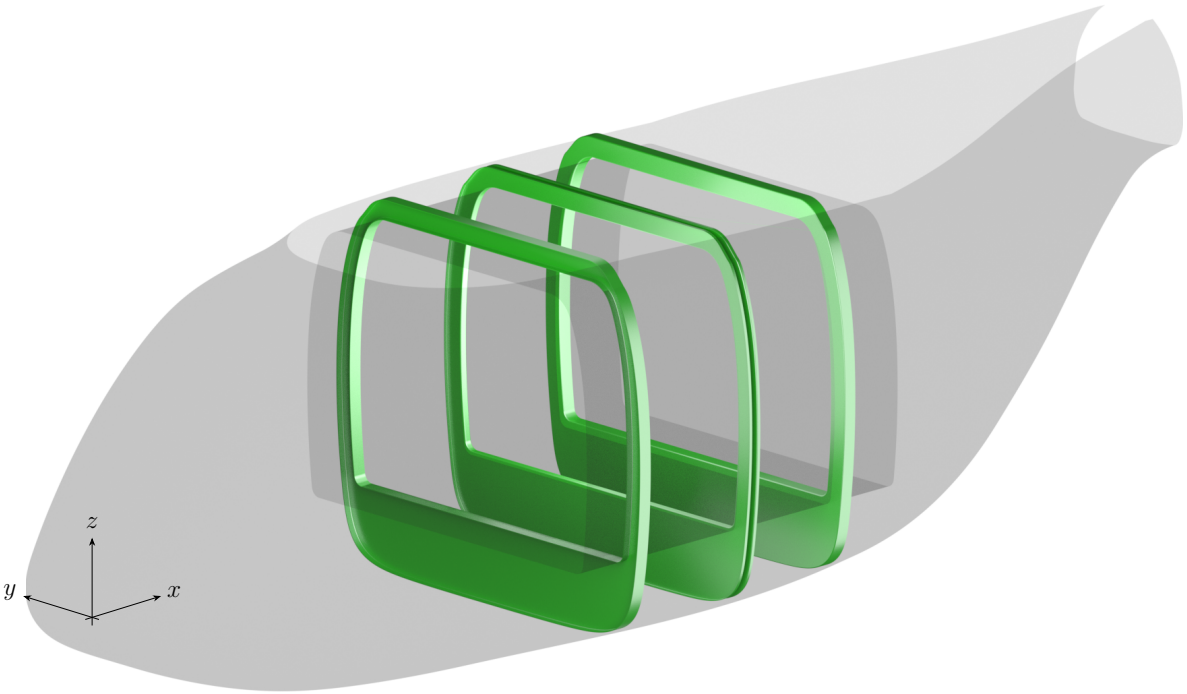


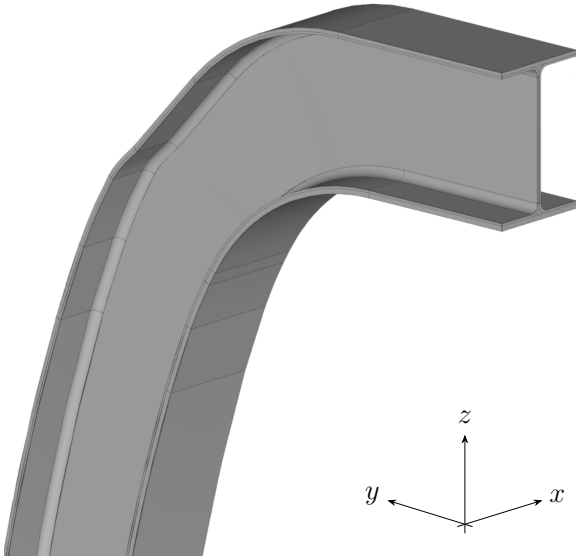
Abbildung 5.17: Automatisierte generierte, verschiedene Varianten von Hubschrauberspanten an unterschiedlichen Positionen innerhalb der Zellgeometrie

Durch die drei in diesem Beispiel definierten Profiltypen (siehe Abbildung 5.16) können entsprechend drei topologisch unterschiedliche Varianten von Spanten vom Entwurfsrahmenwerk

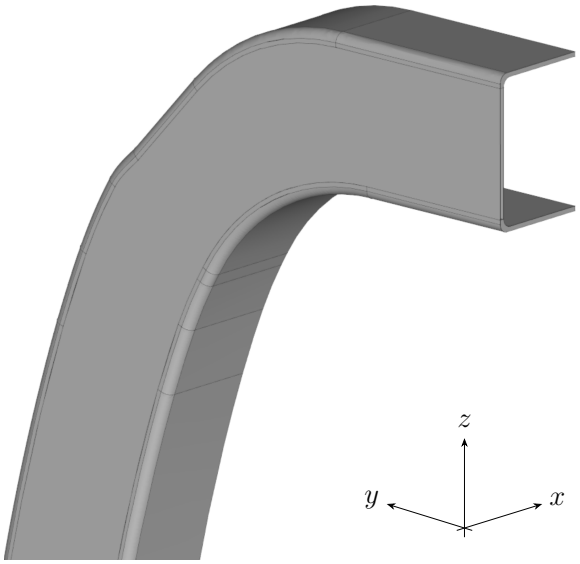
[19] Eine außerhalb der Basisgeometrie determinierte Konstruktionszone führt zu einem direkten Abbruch des Konstruktionsvorgangs.

ausgeprägt werden. Zusätzlich können die Parameter der einzelnen Profilformen in der Theorie nahezu beliebig[20] variiert werden, was zu einer Potenzierung der Variantenvielfalt führt.

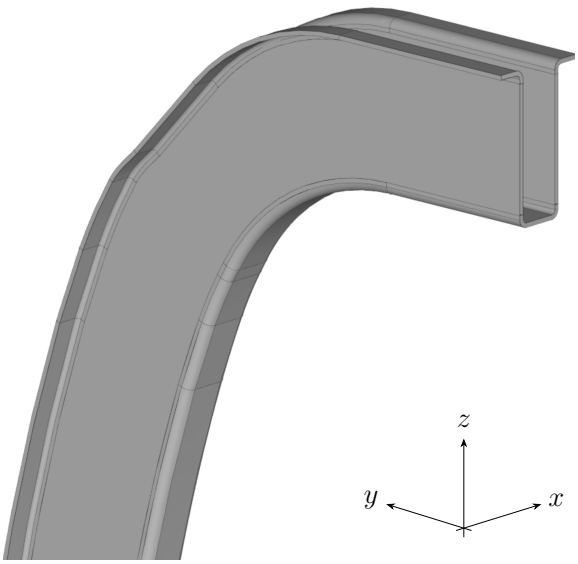
Für die Generierung der in Abbildung 5.17 abgebildeten Spanten wurden die Profilparameter weitestgehend identisch gewählt. In Abbildung 5.18 sind Detailansichten der verschiedenen Spantbauweisen abgebildet. Zur Hervorhebung der Profilform wurde im oberen Bereich ein Schnitt gesetzt, an dem sich die in Abbildung 5.16 dargestellten Profile im Querschnitt zeigen.



(a) „I“-Spantgeometrie



(b) „C“-Spantgeometrie



(c) „Ω“-Spantgeometrie

Abbildung 5.18: Detailansicht der verschiedenen, automatisiert generierten Spantvarianten[21].

[20] Eine explizite Verträglichkeitsprüfung durch Gleichungen bzw. Ungleichungen findet hierbei nicht statt. Kann der verwendete CAD-Kernel OpenCASCADE Technology® [Open Cascade, 2022] die Geometrie nicht ausprägen, wirft dieser eine Fehlermeldung und der Entwurfsvorgang der zugehörigen Variante wird daraufhin abgebrochen.

[20] Zur Kenntlichmachung der Profilformen wurde ein Schnitt kurz vor der linken oberen Kante der Spante gesetzt.

Der Einsatz des Entwurfsrahmenwerks für die Konstruktion des Rumpfspants beschleunigt durch die vollständige Automatisierung den Entwurf auf der einen Seite enorm, ermöglicht auf der anderen Seite eine sehr hohe Flexibilität und Anpassungsfähigkeit. Änderungen der Randbedingungen – beispielweise bezüglich der Position des Spants in der Helikopterzelle, des eingesetzten Materials, des Herstellungsverfahrens oder der Profilgeometrie – können völlig automatisiert in ein neues Produkt überführt werden. Darüber hinaus können verschiedene Konstruktionsvarianten im Sinne eines globalen Optimums bezüglich einer oder mehrerer Bewertungsgrößen leicht untersucht und gegeneinander abgewogen werden.

Die Gesamtdauer der Generierungssequenz des Spants liegt aufgrund der erhöhten Modellkomplexität und den benötigten umfassenden Geometriemanipulationsoperationen im Minutenbereich[22]. Obwohl in der vorgestellten Konstruktionssystematik bereits einige Grundregeln der Mechanik und Festigkeitsberechnung berücksichtigt werden, werden diese aber hier noch nicht umfassend nachgewiesen. Im Zusammenspiel mit einer nachgelagerten Simulation wie beispielsweise einer FEM-Analyse ergäbe sich jedoch die Möglichkeit, die Varianten konstruktiv zu validieren und potenziell darüber hinaus zu zertifizieren. Diese Untersuchungen gehen aber über den Umfang dieses Anwendungsbeispiels hinaus und stellen einen weiteren Anknüpfungspunkt für zukünftige Forschungsvorhaben dar.

5.3.2 Fenster- und Türausschnitte

Korrespondierend mit dem beschriebenen Vorgehen bei der Konstruktion des Rumpfspants müssen auch für die Ausschnitte zunächst die benötigten Randbedingungen festgelegt werden, bevor mithilfe des „elektronischen Messers“ die gewünschten Anforderungen umgesetzt werden können. Hier stellt die Außen- und Kabinengeometrie wiederholt die wichtigste Randbedingung dar. Für den in Flugrichtung links befindlichen Türausschnitt für den Zutritt zur hinteren Kabine müssen mehrere Schnitte hintereinander gesetzt werden, um die benötigte Konstruktionszone zu erhalten. Dies ist in Abbildung 5.19 in Analogie zu Abbildung 5.14 visuell aufbereitet.

Die für diesen Vorgang benötigten Schnittebenen sind in gelber Farbe visualisiert. Die aus den Schnitten abgeleitete Konstruktionszone ist durch eine Färbung in ebenfalls gelber Farbe gut sichtbar hervorgehoben. Im Unterschied zur Konstruktion der Rumpfspante handelt es sich in diesem Fall um eine mehrfach gekrümmte Fläche und nicht um einen Volumenkörper. Die Umrandungskurve dieser Fläche stellt eine spezielle Drahtgitterlinie des gesamten Drahtgitternetzwerks dar. Basierend auf der abgeleiteten Fläche wird in einem nachgelagerten Schritt durch regelbasierte Projektionen und weitere Schnittdefinitionen mithilfe des „elektronischen Messers“ das restliche Drahtgitternetzwerk mitsamt der Konstruktionshilfslinien für den Fensterausschnitt in die Konstruktionszone der Türe einbeschrieben. All diese Drahtgitterlinien werden zusammen für die weitere Auskonstruktion im Rahmen des im Entwurfsrahmenwerk a priori als graphenbasiertes Entwurfsmuster definierten Ausschnittmusters benötigt. Das resultierende Drahtgitternetzwerk für dieses Beispiel ist in Abbildung 5.20 dargestellt.

[22] Die Generierungsdauer beträgt durchschnittlich 12min 25s für den Spant in „I“-, 3min 46s in „C“- und 11min 37s in „Ω“-Profilbauweise auf einer mobilen Workstation aus dem Jahr 2020. Hierbei handelt es sich konkret um einen Dell Precision 7550 mit einer Intel®Core™i7-10875H CPU mit 2.3 GHz Arbeitstakt, 64 GB Arbeitsspeicher und einer dedizierten NVIDIA®Quadro T2000 Grafikkarte.

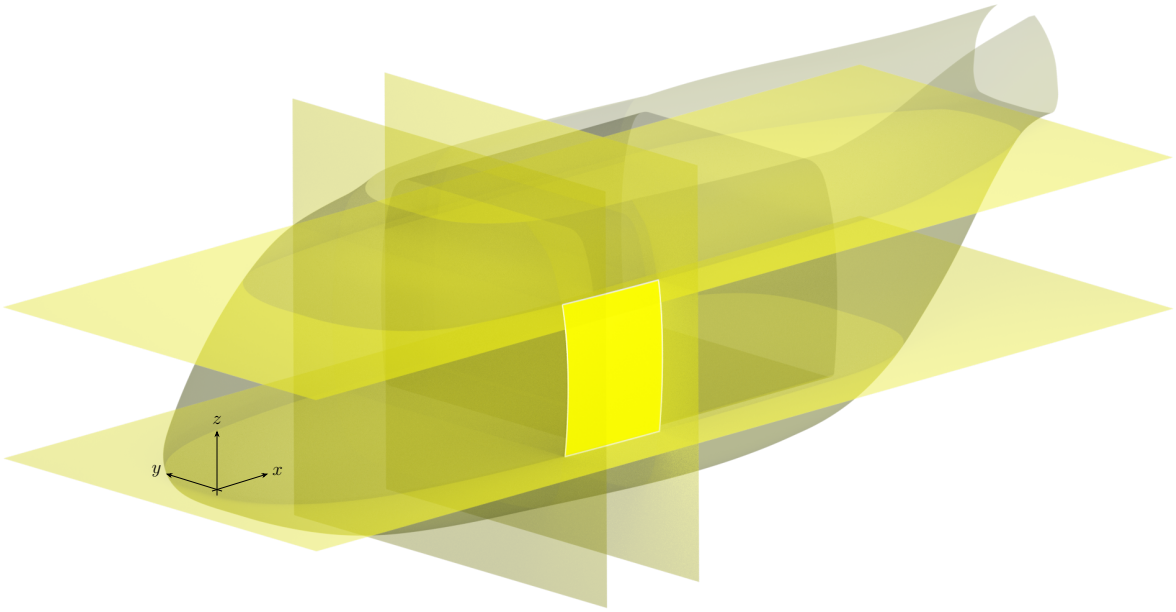


Abbildung 5.19: Ableitung der Konstruktionszone des Fensterausschnitts in der Rumpfstruktur durch multiple Anwendung des „elektronischen Messers“

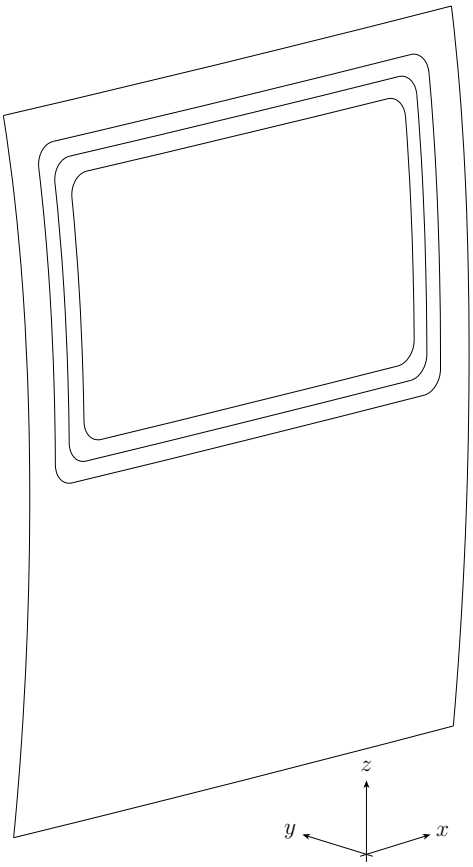


Abbildung 5.20: In die Konstruktionszone des Fensterausschnitts automatisiert einbeschriebenes Drahtgitternetzwerk

Analog zum Beispiel des Rumpfspants wurde im Entwurfsrahmenwerk ein graphenbasiertes Entwurfsmuster für die Konstruktionsaufgabe der Türgeometrie hinterlegt, wodurch auch die Ableitung des benötigten Drahtgitternetzwerks automatisiert geschieht. Die Lage und Dimensionierung der Türe wird maßgeblich durch die Position der Schnittebenen des „elektronischen Messers“ festgelegt. Daneben existiert eine Reihe von Konstruktionsparametern wie beispielsweise ein minimaler Krümmungsradius oder der minimale Abstand zwischen Türaußenkante und Fenster, die mit Default-Werten versehen sind. Diese können allerdings selbstredend in Form einer festen Randbedingung von dem/der Anwender/in noch weiter beeinflusst werden.

Basierend auf dem Drahtgitternetzwerk wird mit der Auskonstruktion fortgefahren und die Türgeometrie schrittweise als Vollkörper generiert. Im zugehörigen Entwurfsmuster ist die Berücksichtigung eines Fensters wie bereits beschrieben ebenfalls vorgesehen. Darüber hinaus kann in diesem Anwendungsbeispiel zwischen zwei verschiedenen Arten der Fensterbefestigung – geklebt oder genietet – variiert werden. In Abbildung 5.21 ist die auskonstruierte Seitenschiebetüre mit geklebt ausgeführter Fensterbefestigung zu sehen.

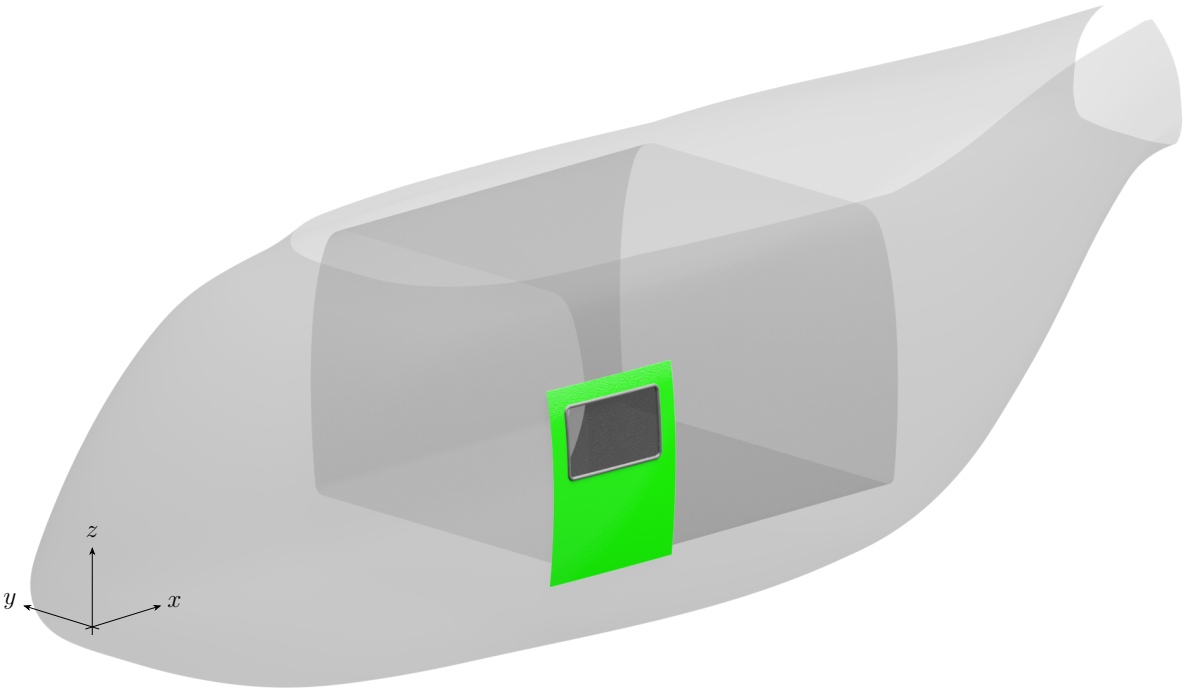


Abbildung 5.21: Automatisiert generierter Türausschnitt mit geklebtem Fenster in der Rumpfstruktur (Herausvergrößerter Detailausschnitt siehe Abbildung 5.22)

Der Türvollkörper entsteht durch das Aufdicken der abgeleiteten Ausgangsfläche (siehe Abbildung 5.19) im Rahmen einer Offset-Operation mit einem gewünschten Wert für den Wandstärkenparameter t_{door}[23]. In den dabei entstandenen Vollkörper wird anschließend der Fensterausschnitt durch eine BOOL’sche Schnittoperation mit einem passenden Schneidwerkzeug erzielt. Das Werkzeug wird auf Basis der Konstruktionshilfslinien automatisiert erzeugt. In diesen

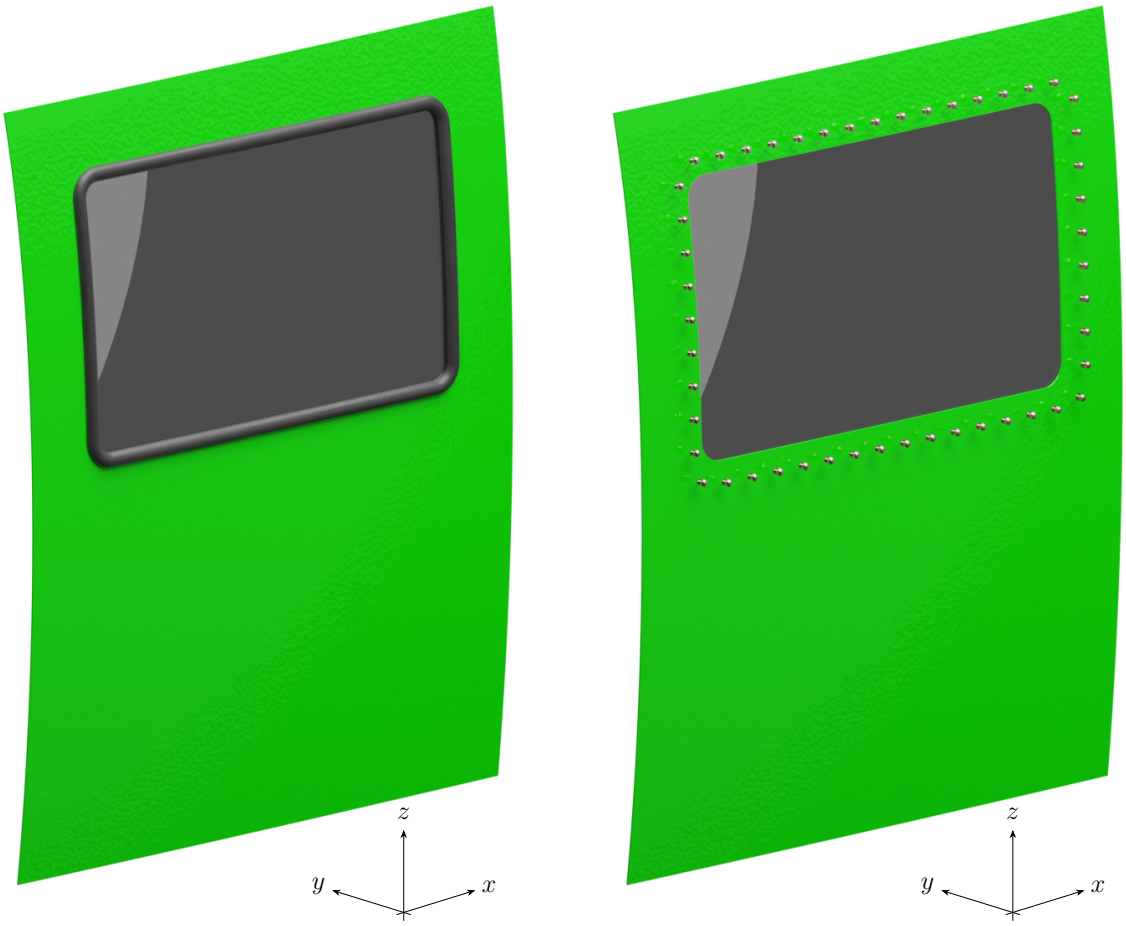
[23] Die Wandstärkenparameter der Tür kann theoretisch beliebig gewählt werden. Das Entwurfsrahmenwerk stellt dabei sicher, dass der gewählte Parameter mit der restlichen Anforderungen verträglich ist und die Tür beispielsweise nicht in den Innenraum hineinragt.

Ausschnitt wird der Volumenkörper des Fensters wiederum durch eine Aufdickung einer auf der Ausschnittsfläche basierenden Hilfsfläche hineinkonstruiert. Je nach gewählter Befestigungsart variiert die Größe des Volumenkörpers des Fensters bei konstanter Fensterausschnittsgröße.

Bei der geklebten Variante wird die Befestigung des Fensters durch zwei Sweep-Operationen entlang der inneren und äußeren Ausschnittkurvensegmentfolge erreicht. Als Profil wird an dieser Stelle standardmäßig ein Halbkreis sowohl auf der Innen- als auch auf der Außenseite angesetzt. Dieser kann von dem/der Anwender/in wieder mit komplexeren Profilgeometrien getauscht werden. Wichtig ist an dieser Stelle, die benötigte Mindestüberlappung sicherzustellen.

Bei der genieteten Fensterbefestigung wird das Fenster größer als der Fensterausschnitt dimensioniert und anschließend in dem Überlappungsbereich mit der Türgeometrie durch Nieten verbunden. Alternativ kann hier auch eine Laschenverbindung vorgesehen werden.

In Abbildung 5.22 sind zwei Türgeometrien für die identische Konstruktionszone dargestellt. In Abbildung 5.22a ist die Bauweise mit eingeklebtem Fenster gezeigt, in Abbildung 5.22b ist die Konstruktionsart mit durch Nieten befestigter Fensterscheibe abgebildet.



(a) Automatisiert generierte Türgeometrie mit geklebtem Fenster

(b) Automatisiert generierte Türgeometrie mit genietetem Fenster

Abbildung 5.22: Detailansicht der mit dem „elektronischen Messer“ abgeleiteten Türvarianten

Außer Türen mit integrierten Fenstern können auch nur Fenster oder verwandte Ausschnitte wie beispielsweise Klappen mit diesem Entwurfsmuster realisiert werden.

Durch die beiden in den vorangegangenen Abschnitten gezeigten, automatisierten Konstruktionen von Strukturbauteilen eines Helikopters konnte die Anwendung des Entwurfsrahmenwerks an einem komplexeren Beispiel demonstriert und die Funktionsfähigkeit validiert werden. Die automatisierte Konstruktion weiterer Strukturbauteile wie beispielsweise Stringern (Längsversteifungen der Zelle) kann dabei völlig analog über ein zusätzliches graphenbasiertes Entwurfsmuster beschrieben werden. Auch an dieser Stelle ergibt sich ein weiterer Anknüpfungspunkt für eine Weiterentwicklung im Rahmen zukünftige Forschungsvorhaben.

5.4 Diskussion

Direkte Implikationen, die sich aus den Definitionen, Modellvorstellungen und konkreten Umsetzungen, die im Rahmen dieser Arbeit entwickelt und implementiert wurden, ergeben, wurden in den entsprechenden Kapiteln 3 - 5.3 bereits eingehend besprochen. An dieser Stelle sollen darüber hinausreichende Fragestellungen aufgegriffen und kurz erörtert werden.

Vollständige Formalisierbarkeit des Konstruktionsprozesses

Diese Arbeit stützt sich unter anderem auf die Erkenntnisse vieler unterschiedlicher Autoren und dabei im Besonderen auf die Befunde zur Formalisierbarkeit bestimmter Aspekte der Produktentstehung. TJALVE hebt in diesem Zusammenhang hervor, dass die Formulierung systematischer Konstruktionsmethoden den Eindruck vermitteln würden, dass durch deren Anwendung automatisch optimale Produkte entstünden. Dies sei aber nicht der Fall, da diese nur in einem ausgewogenen Verhältnis zwischen Systematik und Intuition des/der Entwerfers/Entwerferin entstehen können. [Tjalve, 1979, S. i] Dieser Kritik kann sich auch die vorliegende Arbeit nicht vollständig entziehen, es sei jedoch angemerkt, dass der/die Ingenieur/in im beschriebenen Entwurfsrahmenwerk durch die Wahl der festen Randbedingungen, die Formulierung von Maximaltiefenparametern und weiteren Steuergrößen aktiv mit seiner Erfahrung und Intuition in den automatisierten Prozess eingreifen kann. Auch der passive menschliche Einfluss auf die automatisierten Prozesse über die im Vorfeld mandatorisch zu definierende Wissensbasis und die Modellierung der Regeln und Muster seien an dieser Stelle argumentativ entgegengehalten. Darüber hinaus ist das Stadium der vorliegenden Arbeit der Grundlagenforschung zuzurechnen, wodurch ein direkter Vergleich mit einem menschlich geprägten, manuellen Konstruktionsprozess zum jetzigen Zeitpunkt als verfrüht zu erachten ist, um wirklich aussagekräftig zu sein.

Aus der Sicht des Autors der vorliegenden Arbeit ist eine mit der Formalisierbarkeit eng verknüpfte Fragestellung, ob das Konstruieren an sich prinzipiell lehr- und lernbar ist. Eine Verneinung dieser Frage würde unweigerlich dazu führen, dass der Konstruktionsvorgang kaum formalisierbar – und somit auch nicht maschinell automatisierbar – sein könnte, da es sich um einen sich jeder Regelmäßigkeit entziehenden Prozess handeln müsste.

Durch den Erkenntnisgewinn, der im Rahmen der Konstruktionswissenschaft in den vergangenen Jahrzehnten auf dem Gebiet der Konstruktionslehre stattgefunden hat, wurden

Anschauungen und Modellierungsmöglichkeiten geschaffen, die es ermöglichen, zumindest Teile des Konstruktionsvorgangs zu formalisieren, einer Lehrbarkeit zugänglich zu machen und somit eine Basis für die Automatisierung zu legen. Verfechter der Lehr- und Lernbarkeit können somit im Umkehrschluss als Vertreter der prinzipiellen Automatisierbarkeit gedeutet werden.

HUBKA führt in diesem Zusammenhang bereits Mitte der 1980er Jahre aus, dass „*ohne weitere Beweise postuliert werden [kann], dass die Einführung der Konstruktionswissenschaft den Unterrichtsprozess effektiver gestaltet und die Chancen für die Konstruktionsausbildung erhöht. Dies noch stärker dann, wenn in allen Konstruktionsfächern ein nach der allgemeinen Konstruktionswissenschaft harmonisierter Unterricht eingeführt wird und auch in den ‚reineren' Wissenschaftsfächern die Anwendbarkeit und die problemlösenden Arbeitsweisen eingebaut sind.*" [Hubka, 1984, S. 35–36]. Die Formalisierung des Konstruktionsvorgangs führt somit zu einer einfacheren Anwendbarkeit in der Praxis. Auch EHRLENSPIEL argumentiert in diesem Sinne, indem er schreibt, dass „*durch die Konstruktionsmethodik das Konstruieren wenigstens lehrbar geworden [ist], und man es sich nicht, wie früher, nur durch Abschauen und Eigenerfahrungen mühsam und langsam aneignen [muss]*" [Ehrlenspiel und Meerkamm, 2013, S. 9].

Aus Sicht des Autors der vorliegenden Arbeit stellt die systematisierte Konstruktionslehre und die damit verbundene Formalisierung des Konstruktionsprozesses einen essenziellen Weichensteller für die zukünftige Akzeptanz und Verwendung automatisierter Konstruktionssysteme dar. Wie bereits ausgeführt sind die Denkweisen eng miteinander verknüpft und bedingen sich in gewisser Hinsicht gegenseitig. In einem Artikel der INCOSE über die Zukunft des Ingenieurwesens prognostizieren die Autoren in diesem Sinne folgerichtig, dass „*im Jahre 2035 die Praktiken des Systems Engineering auf einer Reihe theoretischer Grundlagen und anderer allgemeiner Prinzipien beruhen werden, die im Rahmen des Lehrplans für Systems Engineering konsequent gelehrt werden.*" [INCOSE, 2021, S. vii].

Kombinatorische Explosion

Die vorgestellte Automatisierungssystematik basiert unter anderem auf einem in Universalontologien ausgedrückten Basiselementmodell, das einem morphologischen Kasten [Zwicky, 1971] nicht unähnlich ist. Im Rahmen eines Re-Engineerings wird auf dieser Basis durch eine Kombinationsexploration eine großangelegte Entwurfsraumuntersuchung ermöglicht. Die bei einem solchen Ansatz exponentiell mit der Anzahl freier Randbedingungen steigende Variantenanzahl kann schnell zu einem Laufzeit- und Speicherproblem im Entwurfsrahmenwerk führen.

Die Verwendung von Heuristiken sowie eine Beschränkung der Suchtiefe [Rudolph, 1995, S. 7], aber noch wichtiger die Definition von Constraints[24], die a priori unverträgliche Varianten ausschließen können, helfen im vorliegenden Ansatz dabei, diese Auswirkungen abzumildern und bis zu einem gewissen Grad zu umgehen. Für eine industrielle Anwendung der Systematik reichen diese Mechanismen alleine aber nicht aus. Hierfür muss neben einer extensiven Ausweitung der Rechenkapazität durch Parallelisierung auf einer Vielzahl unterschiedlicher Rechenknoten auch die Speicherkapazität stark vergrößert werden. Darüber hinaus ist auch die

[24] Als Constraints werden hier Zwangsbedingungen in Form von Gleichungen, Regeln oder sonstigen Beschränkungen bezeichnet, die einen Ausschluss unverträglicher Kombinationen ermöglichen.

Verwendung dedizierter Hardware-Software-Kombinationen denkbar, die auch bei den ersten menschlichen Experten überlegenen Schachcomputern[25] erfolgreich eingesetzt wurden.

Auch der Einsatz von Methoden der Künstliche Intelligenz (KI) wird im industriellen Umfeld unumgänglich sein. Es sollte allerdings tunlichst von dem Gedanken Abstand genommen werden, dass mit nahezu unbegrenzter Hard- und Softwarepower alle potenziell auslegbaren Konstruktionen mit einer solchen Systematik evaluiert werden könnten. Im Rahmen der sogenannten finitistischen Erkenntnistheorie definiert GIERER eine aus natürlichen Gegebenheiten resultierende, „kosmische Obergrenze analytischer Operationen“ von 10^{120} [Gierer, 1985, S. 45]. Die Existenz dieser Grenze vorausgesetzt, ergibt sich daraus, dass Problemstellungen, deren Klärung mehr als 10^{120} Rechenoperationen benötigen, als *prinzipiell unentscheidbar* aufgefasst werden müssen. Dabei gilt es zu berücksichtigen, dass nach Auffassung mancher Forscher „*unentscheidbare Probleme auch für Quantencomputer unlösbar [bleiben]*“ [Mainzer, 2019, S. 219].

Um den Zusammenhang der kosmischen Obergrenze mit der explodierenden Variantenanzahl herzustellen, sei erneut auf das in Verlauf dieser Arbeit oft zitierte Beispiel des Schachs zurückgegriffen. Allein durch die ersten 40 Züge einer Schachpartie werden mögliche Spielpartien in der Größenordnung von 10^{120} eröffnet [Bonsdorf et al., 1978, S. 13]. Trotzdem sei an dieser Stelle noch einmal darauf hingewiesen, dass Schachcomputer heutzutage nicht mehr von einem menschlichen Experten geschlagen werden können. Diese beobachtete Charakteristik wird sich nach Meinung des Autors der vorliegenden Arbeit – ohne weitere Beweise hierfür vorweisen zu können – auch im automatisierten ingenieurwissenschaftlichen Entwurf zukünftig abzeichnen.

Innovationsfähigkeit automatisierter Konstruktionsrahmenwerke

Wie bereits im Einführungskapitel dieser Arbeit (siehe Kapitel 1) einschränkend dargelegt, sollten im Rahmen dieser Arbeit vorwiegend Konstruktionsprozesse betrachtet werden, über die bereits ein fundiertes Konstruktionswissen vorhanden ist. Diese Art von Konstruktionen ist einer Automatisierung leichter zugänglich. Doch was ist mit hochinnovativen Produkten, die auf wenig bekannten Lösungsprinzipien aufbauen und im Rahmen einer Neukonstruktion entworfen werden sollen? Dieser Entwurfskontext – der bewusst weitgehend ausgeklammert wurde – wirft unweigerlich die Frage nach der prinzipiellen Innovationsfähigkeit eines, wie in dieser Arbeit beschriebenen, automatisierten Konstruktionsframeworks auf. In [Riestenpatt gen. Richter und Rudolph, 2020] werden von den Autoren drei verschiedene Möglichkeiten genannt, wie derartige Innovationsmechanismen im Zusammenhang mit formalen Sprachen erreicht und umgesetzt werden können.

Die erste Option ist eine, auf „Trial-and-Error“ basierende, „unorthodoxe“ Ausprägung eines Objektes aus der Wissensbasis bezüglich seiner Eigenschaften [Riestenpatt gen. Richter und Rudolph, 2020, S. 11]. Als anschauliches Beispiel kann hier eine Schraube ins Feld geführt werden, der in ihrer Definition in der Wissensbasis beispielsweise ein Parameter „Gewindesteigung P“ zugeschrieben wurde. Bei der Instanziierung einer Schraube mit der willkürlich festgelegten Eigenschaft der Gewindesteigung zu null ($P = 0$), kann die Schraube fortan aus

[25] Als Beispiel kann in diesem Zusammenhang der im Jahr 1997 von IBM fertig entwickelte Schachcomputer Deep Blue angeführt werden.

Kategorisierungsgesichtspunkten nicht mehr von einem Bolzen oder einem Nagel unterschieden werden, die definitorisch kein Gewinde und damit keine Gewindesteigung aufweisen. Sollten in der aktuellen Wissensbasis des Entwurfsrahmenwerks weder die Klasse Bolzen noch die Klasse Nagel beinhaltet sein, kann diese unkonventionelle Ausprägung der Eigenschaft der Gewindesteigung als Innovation durch Emergenz gedeutet werden.

Aus einem philosophischen Blickwinkel heraus betrachtet müssen für eine solche Emergenz die Beschreibungsräume der zwei ineinander übergehenden Begrifflichkeiten eine Schnittmenge oder zumindest geteilte Berandungskurve aufweisen, da ansonsten die Überführung des einen auf das andere Element und vice versa nicht möglich sein könnte.

Eine weitere Möglichkeit der Innovation besteht nach RIESTENPATT GENANNT RICHTER und RUDOLPH in der manuellen Addition einer neuartigen Klasse mitsamt ihrer Eigenschaften in der Wissensbasis. Die anschließende Instanziierung erzeugt das bis dahin unbekannte, innovative Repräsentationsobjekt. Es ist hervorzuheben, dass bei diesem Vorgang die Innovation aktiv von außen durch den/die Anwender/in in das Entwurfsrahmenwerk eingebracht werden muss. [Riestenpatt gen. Richter und Rudolph, 2020, S. 11]

Die letzte Innovationsmöglichkeit stellt die Implementierung einer regelbasierten Optimierungsstrategie dar, die in einem automatisierten Konstruktionsframework ebenfalls zu innovativen Produkten führen kann [Riestenpatt gen. Richter und Rudolph, 2020, S. 11].

In Bezug auf die Ausgangsproblemstellung der Innovationsfähigkeit sei an dieser Stelle noch einmal vor Augen geführt, dass diese hauptsächlich im Bereich der Neukonstruktion anzusiedeln ist. Die Neukonstruktion an sich „*ist zwar sehr wichtig, macht i.A. aber höchstens 10 % der in der Praxis zu leistenden Konstruktionsarbeit aus.*“ [Ehrlenspiel und Meerkamm, 2013] und spielt damit eine eher untergeordnete Rolle im Gesamtkontext des Entwurfs. Auch KOLLER ist der Meinung, dass „*neue, unbekannte Produktarten, für welche noch keine Lösungen und Konstruktionsprozesse bekannt sind, [. . .] nur sehr selten zu konstruieren*“ sind [Koller, 1998, S. 463]. Angesichts dieser Feststellungen bleibt am Ende zu konstatieren, dass die Frage nach der Innovationsfähigkeit automatisierter Konstruktionssysteme eher von akademischer Natur ist und die Automatisierung in diesem Anwendungsbereich aus ökonomischer Sicht deutlich weniger Vorteile bietet als bei der Anpassungs- oder Variantenkonstruktion vor einer bekannten Wissensbasis. Dessen ungeachtet ist eine Innovationsfähigkeit gegeben, deren Nutzen sich aber möglicherweise erst in Kombination mit zukünftigen Technologien vollständig offenbart.

Wissensformulierung in Ontologien

Ein Großteil des im Rahmen der vorliegenden Arbeit definierten a priori Wissens wird wie bereits umfassend dargelegt in Form von Ontologien spezifiziert, die somit eine herausragende Rolle im beschriebenen Entwurfsrahmenwerk einnehmen. An dieser Stelle sollen prinzipielle Problemstellungen bei der Verwendung von Ontologien und mögliche Umgehungen diskutiert werden. ARP, SMITH und SPEAR sind drei prominente der zahlreichen Autoren, die sich intensiv mit der Verwendung von Ontologien im wissensbasierten Kontext beschäftigt haben. In [Arp et al., 2015] bringen sie einige Argumente an, warum Ontologien versagen können. Die wichtigsten Argumente sind in Tabelle 5.3 zusammengestellt.

Tabelle 5.3: Probleme bei der Verwendung von Ontologien[26]

1. Ontologien werden von Individuen mit individuellen Vorstellungen und Zielen für die eigene Gemeinschaft oder ein Unternehmen modelliert. Außerhalb dieser Community sind sie oft nicht verwendbar („Silo-Syndrom“). [Arp et al., 2015, S. xviii]
2. Ontologien werden oft vor dem Hintergrund lokal bereits vorhandener Daten erstellt. Ändert sich diese Datenbasis oder ein verwendetes Dateiformat, verliert die Ontologie schnell ihren Wert. Niemand hat somit einen Anreiz, sie weiter zu pflegen („Kurze Halbwertszeit-Syndrom“). [Arp et al., 2015, S. 63]
3. Ontologien werden kontextuell, losgelöst von anderen Ontologien, definiert. Häufige Begriffe wie beispielsweise Ereignis, System oder Eigenschaft haben oft unterschiedliche Bedeutungen („Das Rad neu erfinden Syndrom“) [Arp et al., 2015, S. 47]

Um all diesen Problemen zu begegnen, schlagen die Autoren die Verwendung eines gemeinsamen Architekturframeworks auf oberster Ebene, auf dem alle Ontologien aufsetzen, vor. Diese Metaebene bezeichnen sie als „Basic Formal Ontology (BFO)“ [Arp et al., 2015, S. 85 ff.]. Für zukünftige Überarbeitungen der in dieser Arbeit beschriebenen Implementierung sollte geklärt werden, ob die BFO und ihre Mechanismen hierfür angewendet werden kann.

Ebenfalls im Zusammenhang mit der Definition der Ontologien sollte die generelle Verwendung von Vererbungen in der Ontologiearchitektur kritisch diskutiert werden. Die Anwendung der FCA (siehe Abschnitt 2.2.3) zur Ableitung der Ontologie führt in der Regel auf eine häufige Verwendung der Generalisierung. Aus implementatorischer Sicht bringt deren Einsatz einige strukturelle Nachteile mit sich, die im Folgenden diskutiert werden sollen.

Eines der weitreichendsten Probleme ist, dass die erbende Klasse durch die Übernahme aller Eigenschaften und Methoden der Oberklasse ihre Schnittstelle nach außen nicht mehr unter ihrer alleinigen Kontrolle hat. Dies tritt im Besonderen bei einer neu hinzugefügten Methode in der Basisklasse zutage, da die zusätzliche Funktionalität sofort auch in allen Kindklassen auftaucht, ohne dass sich an deren Konstitution zwingend etwas geändert haben muss. Der gleiche Zielkonflikt ergibt sich bei der Änderung einer Methode in der Basisklasse oder deren Struktur in der Ontologie, die automatisch die Änderung an der Methode und Struktur in den erbenden Klassen nach sich zieht. Es gibt selbstredend Änderungen, bei denen die automatische Anpassung der abgeleiteten Klassen sinnvoll und erwünscht ist, aber dies trifft nicht auf alle Arten von Änderungen zu. Beispielsweise können vererbte Assoziationen in den Kindklassen nicht mehr konkretisiert beziehungsweise eingeschränkt werden. Das bedeutet, dass auf abstrakter Ebene der Basisklasse gezogene Assoziationen spätere Verlinkungen zwischen konkreten Objekten von Kindklassen erlauben, die eigentlich nicht erlaubt sein sollten, beziehungsweise nicht sinnvoll sind. Als Beispiel sei hier eine abstrakte Klasse `Funktion` angeführt, die im Klassendiagramm mit einer `Komponente` assoziiert sei. Nun werde die Klasse `Funktion` durch die Klasse `Bewegen` und die `Komponente` durch die Klasse `Rad` erweitert. Im späteren konkreten

[26] Die Argumente stammen aus einem Vortrag von Barry Smith, der unter [Smith, 2018] als Video mit zugehörigem Foliensatz zu finden ist. Da es sich dabei um eine volatile Quelle handelt, ist zusätzlich an den einzelnen Argumenten [Arp et al., 2015] mit angegeben, wo sich die beschriebenen Sachverhalte ebenfalls finden.

Anwendungsfall soll der Funktion `Bewegen` genau ein `Rad` zugeordnet werden. Nun werde im Klassendiagramm noch eine weitere Komponente `Schloss` hinzugefügt. Im konkreten Fall dürfte dann auf Objektebene die Funktion `Bewegen` ebenfalls mit einem `Schloss` verlinkt werden, was augenscheinlich einen Widerspruch darstellt. Die Lösung dieses Dilemmas könnte in einem Überschreibungsmechanismus für Assoziationen analog zu Methoden in der OOP liegen. Eine weitere Alternative könnte die verstärkte Definition von Wissen in Form von Regeln darstellen. Dies gilt es in zukünftigen Implementierungen abzuwägen und zu berücksichtigen.

Verifikation und Validierung

Beim in dieser Arbeit identifizierten und beschriebenen Reverse-Engineering konnte gezeigt werden, dass spezielle Konstruktionsaufgaben es erfordern, aus einem Domänenartefakt ein abstraktes Modell zurück zu folgern. Detailliert wurde dies anhand der Domäne der Geometrie über einen Mustererkennungsmechanismus basierend auf geometrischen Zentralmomenten dargelegt (siehe Abschnitt 3.1.2 und Abschnitt 5.2). Als Nebenprodukt des beschriebenen Vorgehens ergab sich dabei unter gewissen Voraussetzungen die rekonstruierte Stückliste des ursprünglichen Produkts. Durch die Möglichkeit der automatisierten Ableitung der Stückliste ergeben sich darüber hinaus bemerkenswerte weitere Nutzungsmöglichkeiten im Rahmen des automatisierten Produktentwurfs. Da in einem automatisierten Forward- bzw. Re-Engineering nur über Anforderungen die Generierung des Produkts gesteuert wird, muss am Ende unbedingt verifiziert und validiert werden, dass das gewünschte Produkt tatsächlich umgesetzt und generiert wurde. Diesen Beweis kann man mit den im Rahmen des Reverse-Engineering vorstellten Techniken führen, da man für Bauteile und Baugruppen Wissen in der Wissensbasis besitzt und deren Vorhandensein beweisen kann. Ohne eine solche Technik wäre ein derartiger Nachweis nur durch einen Menschen und seinen Erfahrungshorizont zu erbringen. Dieser Sachverhalt verdeutlicht nochmals eindringlich die Zusammenhänge des Forward-, Reverse- und Re-Engineering in Form des beschriebenen Round-Trip-Engineerings.

Alternative Regelsequenzen

Bei der Vorstellung der dynamischen Regelausführung (siehe Abschnitt 3.2.3) ist bereits angeklungen, dass es potenziell verschiedene Regelsequenzen geben kann, die unter demselben Entwurfskontext auf ein valides, möglicherweise individuelles Produkt führen. In der vorgestellten Implementierung im Rahmen dieser Arbeit wird der erste mögliche Pfad beschritten, potenzielle Alternativpfade werden nicht mehr weiter untersucht. Theoretisch könnte an dieser Stelle ein weiteres Re-Engineering ansetzen, dass alle möglichen Alternativpfade weiterverfolgt und als Produktvarianten in den Generierungslauf einbringt. Bei der Lösung eines unbestimmten Gleichungssystems mit mehreren Lösungssequenzmöglichkeiten führen die verschiedenen Pfade je nach Typ des Gleichungssystems auf dasselbe oder ein individuelles Ergebnis. Die Frage, die sich hier aufdrängt, ist, ob diese Aussage auf den Produktentwurf übertragen werden kann. An dieser Stelle sind weitere Untersuchungen notwendig, um zu überprüfen, ob sich solche Produktvarianten wirklich voneinander unterscheiden und wie möglicherweise a priori

entschieden werden kann, welcher Pfad auf das beste Ergebnis führt. Diese Untersuchungen sind zukünftigen Forschungsaktivitäten vorbehalten.

Industrielle Anwendbarkeit

In Bezug auf die industrielle Anwendbarkeit des im Rahmen der vorliegenden Arbeit vorgestellten und implementierten Ansatzes können mit dem einfachsten Zugang zunächst die Anwendungsbeispiele angeführt werden, die eine industrielle Reife im Stadium des Vorentwurfs oder frühen Detailentwurfs belegen (siehe z. B. Abschnitt 5.3). Bei der in diesem Kontext oftmals aufgerufenen klassischen Einordnung in sogenannte Technology Readyness Levels (TRL) nach [Mankins, 1995] ist der Ansatz der Stufe 3/9 „Analytical and experimental critical function and/or characteristic proof-of-concept“ zuzuordnen und somit noch weit von einem tatsächlichen, vollumfänglichen industriellen Einsatz entfernt. Dies ist auch nicht die Intention der vorliegenden Arbeit. Hier sollen vielmehr die Grundlagen gelegt und anhand einer prototypischen Implementierung demonstriert werden.

Einsatz Künstlicher Intelligenz

Von zentraler Bedeutung in dieser Arbeit ist die Frage, ob der Konstruktionsvorgang in all seinen Facetten vollständig automatisierbar ist (vergleiche FF2). In der oben bereits gegebenen Antwort, ist ein Mensch weiterhin bei der Definition der Wissensbasis, der Regeln und Muster explizit bei der Automatisierung beteiligt. Über diese Vorstellung hinaus, kann die Frage aber auch dergestalt verstanden werden, dass auch diese Prozesse automatisiert von einer Maschine ausgeführt werden könnten. Insbesondere im hochinnovativen Neukonstruktionsbereich ist der Sachverhalt daher eng mit der historisch umstrittenen Fragestellung nach der potenziellen Abbildbarkeit kognitiver Fähigkeiten durch eine Maschine verwandt (siehe z. B. [Penrose, 1989, S. 3–29]). Bis heute streiten sich die Vertreter der „starken“ und „schwachen“ künstlichen Intelligenz (KI) über diesen Kernpunkt und es wird wohl in absehbarer Zeit keine eindeutige Antwort darauf geben. Einen Beleg für die starke KI konnte bisher trotz aller Bemühungen nicht erbracht werden [Gethmann et al., 2022, S. 14]. Diese Feststellung sollte aber bei weitem nicht als Showstopper verstanden werden, denn „*künstliche Intelligenz muss keineswegs menschliche Intuition und Emotion nachahmen, um den Menschen bei Entscheidungen unter unvollständiger Information zu schlagen*“ [Mainzer, 2019, S. 257]. Die Verknüpfung des in dieser Arbeit beschriebenen Entwurfsrahmenwerks mit Methoden der KI könnte daher spannende und ungeahnte neue Möglichkeiten eröffnen.

6 Zusammenfassung

> *Innovation ist die Fähigkeit, Veränderung als Chance und nicht als Bedrohung anzusehen.“*
>
> **— Steve Jobs**
> (US-amerikanischer Unternehmer)

In diesem Kapitel werden die wichtigsten Erkenntnisse aus der vorliegenden Arbeit noch einmal zusammengefasst und die im Einführungskapitel aufgestellten Forschungsfragen beantwortet. Das Kapitel schließt mit einem Ausblick, wie die in dieser Arbeit gewonnenen Erkenntnisse und Methoden in Zukunft angewendet und noch weiter ausgebaut werden können.

Kapitelübersicht

6.1 Ergebnisse

In Rahmen der vorliegenden Arbeit wurden sowohl theoretische als auch praktische Resultate im Zusammenhang mit der Automatisierung des ingenieurwissenschaftlichen Produktentwurfs erzielt. Der präskriptive Charakter des theoretischen Kapitels (siehe Kapitel 3) konnte durch die anschließenden implementatorischen (siehe Kapitel 4) und anwendungsorientierten (siehe Kapitel 5) Kapitel weitergehend konkretisiert, verifiziert und schließlich validiert werden.

Als Erstes wurde in der vorliegenden Arbeit dargelegt, dass sich die drei aus dem Softwareentwurf bekannten Vorgänge des Forward-, Reverse- und Re-Engineering (siehe Abschnitt 2.1.4) im Ingenieurentwurf in ähnlicher Form identifizieren lassen und dieser somit einer Übertragung softwaretechnischer Automatisierungstechniken grundsätzlich zugänglich ist.

Das Forward-Engineering (siehe Abschnitt 3.1.1) tritt im Ingenieurentwurf bei der klassischen Betrachtungsweise nach PAHL und BEITZ in Entwurfsaufgaben wie der Neu- oder Variantenkonstruktion zutage. In dieser Arbeit wurde eine Vorgehensweise zur Automatisierung des Forward-Engineering auf Basis einer domänenübergreifenden Wissensbasis konzipiert. Dabei wird aus eingangs definierten Anforderungen durch eine auf den aktuellen Entwurfskontext abgestimmte, *individuelle* Modelltransformationssequenz, ein konkretes Modell generiert, das das zu konstruierende Produkt repräsentiert. Für eine einheitliche Definition und ein optimales Zusammenspiel mit den weiteren entwickelten Werkzeugen wurde in der Arbeit des Weiteren gezeigt, wie die Formal Concept Analysis (FCA)[1] genutzt werden kann, um *objektorientierte*

[1] Zu deutsch formale Begriffsanalyse.

Universalontologien als Repräsentation der angesprochenen Wissensbasen abzuleiten. Die Implementierung wurde exemplarisch für die fünf aus Sicht des Autors der vorliegenden Arbeit im mechanischen Produktentwurf wesentlichen Domänen der Funktion, der Form, des Materials, der Herstellung und der physikalischen Lasten durchgeführt (siehe Abschnitt 3.2.2 für die theoretische Beschreibung und Abschnitt 4.2 für die Implementierung der Domänen).

Das Reverse-Engineering (siehe Abschnitt 3.1.2) wurde in der vorliegenden Arbeit als weiterer möglicher Einstiegspunkt in den Iterationszyklus des Round-Trip-Engineering präsentiert. Konkret handelt es sich hierbei um eine Konstruktionsaufgabe, deren Zielprodukt in Form eines Entwurfsartefakts – wie beispielsweise eines CAD-Modells – gegeben ist, zu dem aber keine weiteren Modelle im Sinne eines modellbasierten Systementwurfs bekannt sind. Es wurde dargelegt, dass im Rahmen des Reverse-Engineering eine Rekonstruktion dieser fehlenden, abstrakten Modelle aus den vorhandenen Modellartefakten umgesetzt werden muss. Von besonderer Bedeutung ist die Verknüpfung dieser Rekonstruktion mit der Lösung eines zugrundeliegenden Mustererkennungsproblems. Hierfür war es erforderlich, eine in der Model-driven Architecture (MDA)[2] (siehe Abschnitt 2.1.2) bisher nicht definierte Text-zu-Modell (T2M)-Transformation einzuführen. In der vorliegenden Arbeit wurde am Beispiel der Geometriedomäne detailliert dargelegt, wie ein solches Reverse-Engineering im Rahmen eines automatisierten Round-Trip-Engineerings konzeptuell umgesetzt werden kann (siehe Abschnitt 3.1.2). Am Beispiel eines Satellitengehäuses konnte die im Rahmen dieser Arbeit vorgenommene konkrete Implementierung erfolgreich validiert werden (siehe Abschnitt 5.2). Dabei wurde ebenfalls gezeigt, wie Modelle und Parameter aus einem geometrischen Entwurfsartefakt rekonstruiert und als Nebenprodukt auch die Stückliste der Baugruppe abgeleitet und bewiesen werden kann.

Zuletzt wurde in dieser Arbeit das Re-Engineering aus der Softwaretechnik als „Umkonstruktion“ eines bestehenden Entwurfs im weitesten Sinne auf das Ingenieurwesen übertragen (siehe Abschnitt 3.1.3). Unter den Geltungsbereich des Re-Engineerings fallen somit sowohl parametrische – z. B. die Anpassung eines gewünschten Maximalgewichts – als auch topologische Änderungen, wie beispielsweise der Wechsel auf eine alternative Stoßgeometrie bei einem Zusammenbau. In der klassischen Konstruktionsphilosophie nach PAHL und BEITZ würde diesem Vorgang die Anpassungs- oder auch die Variantenkonstruktion am nächsten kommen. Als Ergebnis dieser Arbeit ist in diesem Zusammenhang die Einführung zweier spezifischer Modellierungswerkzeuge zu nennen. Erstens die Definition sogenannter graphenbasierter *Spezifikations- und Abstraktionscasts* als Übertragung und Erweiterung des klassischen Casts aus der Informatik (siehe Abschnitt 3.1.3), mit deren Hilfe insbesondere topologische Änderungen einfach modelliert werden können. Zweitens die Definition einer Sammlung fertigungstechnischer und bauteiltheoretischer Prinziplösungen in einer Bibliothek generischer Graphentransformationen als Übertragung eines objektorientierten Entwurfsmusters auf das Ingenieurwesen, die Formulierung notwendiger Änderungssequenzen vereinfachen. In der vorliegenden Arbeit wurden diese als sogenannte *graphenbasierte Entwurfsmuster* beschrieben und am Beispiel der Geometriedomäne in Form einer generischen Stoßdatenbank für Übergangsgeometrien konkret implementiert (siehe Abschnitt 4.2.3). Im Rahmen der Anwendungsbeispiele „Verbindungen

[2] Zu deutsch modellgetriebene Architektur.

dünnwandiger Strukturbauteile“ (siehe Abschnitt 5.1) sowie „Hubschrauberstrukturelemente“ (siehe Abschnitt 5.3) konnte die korrekte Funktionsweise verifiziert und validiert werden.

Insgesamt wurde in dieser Arbeit mit der Konzeption des RTE ein möglicher Weg aufgezeigt, wie basierend auf der Vorstellung des modellbasierten Systementwurfs, alle identifizierten Teilprozesse in ein und demselben Iterationsprozess automatisiert werden können. Als Fundament für das Zusammenspiel der verschiedenen Teilprozesse des RTE wurden in dieser Arbeit verschiedene Arten von Randbedingungen einer Konstruktionsaufgabe identifiziert und in ihrer Konsequenz für die Ableitung der automatisierten Entwurfssequenz diskutiert (siehe Abschnitt 3.2.1). Der dynamischen Reaktion des Entwurfsrahmenwerks auf geänderte Anforderungen wurde durch die Definition eines generischen Aktivitätenschemas Rechnung getragen, das die geforderte Flexibilität bei der Ausführung sicherstellen konnte (siehe Abschnitt 3.2.3). Dafür wurde in der Implementierung die sogenannte *dynamische Regelausführung* entwickelt, die im Wesentlichen auf der Anwendung von Graphenalgorithmen zur Bestimmung der Entwurfssequenz zur Laufzeit der Entwurfssprache basiert (siehe Abschnitt 4.4).

Darüber hinaus wurde im Rahmen dieser Arbeit unabhängig von der Prozessart für die Einbringung sogenannter geometriewirksamer Randbedingungen bei einer Konstruktionsaufgabe mit vorgegebener oder aus anderen Gründen fester äußerer Form das sogenannte *elektronische Messer* entwickelt und implementiert (siehe Abschnitt 3.2.2). Mithilfe dieses innovativen Werkzeugs wurde die Einbringung sich auf die äußere Grundform auswirkender Randbedingungen durch eine BOOL’sche Schnittdefinition an der gewünschten Position in den aktuellen Entwurfskontext ermöglicht. Einer oder mehrere solcher Schnitte wurden dabei in sogenannte Konstruktionszonen überführt und im weiteren Fortgang einer musterbasierten Geometriemanipulation zugänglich gemacht. Die Anwendung des elektronischen Messers konnte am Beispiel der automatisierten Definition von Übergangs- oder Verstärkungsgeometrien (siehe Abschnitt 5.1) und auch anhand der Detailkonstruktion von Hubschrauberstrukturelementen (siehe Abschnitt 5.3) an einem industriell relevanten Anwendungsbeispiel demonstriert werden.

Alles in allem konnte in dieser Arbeit dargelegt werden, wie für ein und dieselbe Konstruktionsaufgabe automatisiert mit den gewünschten Randbedingungen verträgliche Varianten bezüglich des Materials, der Herstellungsprozesse, der Sekundärgeometrien und physikalischen Lasten erzeugt, gegeneinander abgewägt und in der Konsequenz die Annäherung an ein globales Optimum über die verschiedenen Domänen hinweg erzielt werden kann. Die vollständige Automatisierung des zugrundeliegenden RTE löst dabei das im Einführungskapitel beschriebene „Paradoxon der Konstruktion“ (siehe Abbildung 1.1) durch eine große Detailtiefe der Varianten, die in kurzer Zeit generiert werden können. Überdies wurde festgestellt, dass die drei Prozesse des Forward-, Reverse- und Re-Engineering je nach Aufgabenstellung beliebig untereinander kombiniert auftreten können und diese somit in einem gemeinsamen Zusammenhang stehen. Die Wechselbeziehung der verschiedenen Teilprozesse als RTE konnte in allen in Kapitel 5 gezeigten Beispielen beobachtet werden. Für die Implementierung wurden die graphenbasierten Entwurfssprachen um die genannten Modellierungskonzepte und Werkzeuge erweitert.

Im Verlauf dieser Arbeit, wurden die in der Einführung motivierten und in Tabelle 1.1 formu-

lierten Forschungsfragen bereits umfassend beantwortet. An dieser Stelle sollen sie nochmals aufgegriffen und die ausführlichen Antworten prägnant zusammengefasst werden.

FF 1 *Welche Modelle und Vorgehensweisen aus dem Softwareentwicklungsprozess lassen sich im Hinblick auf eine Automatisierung auf den ingenieurwissenschaftlichen Entwurf übertragen?*

Im Rahmen der vorliegenden Arbeit konnte gezeigt werden, dass das Round-Trip-Engineering bestehend aus Forward-, Reverse- und Re-Engineering sowie modellbasierte Automatisierungstechniken aus der Model-driven Architecture (MDA) auf den Ingenieurentwurf übertragen werden kann. Darüber hinaus wurde dargelegt, wie speziell definierte, objektorientierte Ontologien, erweiterte Casts und graphenbasierte Entwurfsmuster für die Wissensmodellierung und Produktgenerierung verwendet werden können.

FF 2 *Welche Prozesse der Produktentstehung eignen sich für eine Automatisierung?*

Prinzipiell lassen sich die allermeisten Teilprozesse der Produktentstehung automatisieren. Insbesondere, wenn das benötigte Konstruktionswissen theoretisch und methodisch bereits durchdrungen ist, ist eine Automatisierung mit den in dieser Arbeit beschriebenen Methoden einfach umsetzbar (Re- und Reverse-Engineering). Neuartige und innovative Konstruktionsvorgänge lassen sich theoretisch ebenfalls im Rahmen eines Forward-Engineerings automatisieren, eröffnen aber initial weniger Potenzial zur Zeitersparnis, da die Wissensbasis, die benötigten Regeln und Muster parallel zur Produktentwicklung abgeleitet und definiert werden müssen.

FF 3 *Wie kann der verschiedene Abstraktionsebenen überschreitende, domänenübergreifende, mehrfach iterative Prozess des Ingenieurentwurfs in einem automatisierten Framework abgebildet werden?*

Die Definition von Universalontologien mit der FCA sowie die in dieser Arbeit beschriebene Formulierung von Regeln und Aktivitäten ermöglichen durch graphenbasierte Entwurfsmuster eine automatisierte Entwurfsraumexploration, die alle Ebenen der Produktentstehung überstreicht. Die Produktgenerierung wird somit vollständig formalisiert, modularisiert und maschinell ausführbar gemacht. Die Grundlage für die Implementierung bilden graphenbasierte Entwurfssprachen.

FF 4 *Wie können Anforderungen an ein Produkt variabel definiert und in einem automatisierten Verarbeitungsprozess dynamisch zur Laufzeit interpretiert werden, sodass geänderte Anforderungen auf ein geändertes Produkt führen?*

In dieser Arbeit wurde durch die Auffassung des Produktentwurfs als Analogon zu einem nicht linearen System in der Mathematik eine Herangehensweise eröffnet, die den Aufbau der Produktentstehungssequenz durch die Aufteilung in feste und freie Randbedingungen ermöglicht. Da die Randbedingungen direkt mit den objektorientierten Ontologien korrelieren, ist eine automatisierte Typbestimmung – fest oder variabel – aller Randbedingungen im aktuellen Entwurfskontext möglich. Der iterative Aufbau des Gesamtaktivitätsdiagramms zusammen mit

in speziell definierten Regeln hinterlegten Entwurfsmustern ermöglicht eine sich zur Laufzeit selbst assemblierende Regelsequenz, entlang derer die mit den spezifizierten Randbedingungen verträglichen Produktvarianten konkretisiert und generiert werden. Als Grundvoraussetzung für diese Vorgehensweise muss sichergestellt sein, dass die durch die Ontologien beschriebene *Einhüllende* das gewünschte Produkt zu umfassen in der Lage ist (closed world assumption).

FF 5 *Wie lassen sich Anforderungen, die sich auf die Form eines Produktes auswirken, im ingenieurwissenschaftlichen Entwurf allgemeingültig abbilden und einbringen?*

Durch die im Rahmen dieser Arbeit entwickelte Definition eines sogenannten elektronischen Messers kann eine vorgegebene äußere Hüllgeometrie in sogenannte Konstruktionszonen unterteilt, an den Rändern aufgetrennt und auf Basis eines in die Konstruktionszone einbeschriebenen, manipulierbaren Drahtgitters der Berücksichtigung geometriewirksamer Randbedingungen zugänglich gemacht werden. Die konkrete Ausgestaltung der in die Konstruktionszone einzubeschreibenden Sekundärgeometrien – wie beispielsweise Stoß- oder Verstärkungsgeometrien – wird in einer Sammlung generischer Geometriemanipulationsvorschriften allgemeingültig in einer zum Entwurfsrahmenwerk gehörenden Bibliothek vorgehalten. Diese geometrischen Entwurfsmuster lassen sich dadurch auf beliebige Eingangsgeometrien anwenden.

6.2 Ausblick

Im Rahmen der vorliegenden Arbeit wurden im wissens- und modellbasierten Ingenieurentwurf neuartige Modellierungsansätze und Werkzeuge erarbeitet, die eine Automatisierung diverser Konstruktionsaufgaben wie Varianten-, Anpassungs- oder – mit gewissen Einschränkungen – auch Neukonstruktionen ermöglichen. Aufgrund des sehr weitgefassten Themenspektrums und den vielfältigen Anwendungsfeldern werden entsprechend viele Anknüpfungs-, Erweiterungs- und Verbesserungspunkte für zukünftige Forschungsarbeiten eröffnet. Einige dieser Punkte sind bereits im vorangegangen diskursiven Abschnitt angeklungen.

Das Reverse-Engineering bietet eine erste Erweiterungsmöglichkeit durch die Identifikation, Untersuchung und Lösung des Mustererkennungsproblems in weiteren Domänen über die Geometrie hinaus. Darunter könnten beispielsweise Strömungsmechanik-, Finite-Elemente-, Mehrkörper- und weitere Simulationsmodelle fallen und viele nützliche zusätzliche Informationen in das abstrakte Modell mit einbringen. Ebenfalls würde sich die Erprobung des Mechanismus in einer Erweiterung auf die zugehörige digitale Fabrik anbieten, sofern hier ein entsprechendes Entwurfsartefakt vorliegt. Prinzipiell sollte das digitale Modell der Herstellung im besten Fall bereits im Rahmen des Forward- oder Re-Engineerings integriert mit generiert werden. Ein erster Ansatz in diese Richtung, auf dem an dieser Stelle aufgesetzt werden könnte, wurde in [Kübler et al., 2020] und [Schopper et al., 2021] bereits dargestellt.

Eine weitere Anknüpfung beziehungsweise Erweiterung des in dieser Arbeit vorgestellten RTE kann bei der Ableitung und Definition der Ontologien ansetzen. Ontologien sind wissenschaftlich mittlerweile gut untersucht und bereits in vielen Bereichen eingesetzt worden. In der

Zwischenzeit wurden sogar Standards[3] formuliert, die den Übergang eines Forschungsbereichs in einen Anwendungsbereich andeuten. In diesem Zusammenhang könnte untersucht werden, inwieweit durch eine Verknüpfung des Entwurfsrahmenwerks mit aktuellen ontologischen Technologien wie dem sogenannten Semantic Web[4], der Web Ontology Language (OWL) und Werkzeugen wie beispielsweise Protégé[5] Synergien erzielt werden könnten.

Das vorgestellte Entwurfsrahmenwerk ermöglicht das Aufspannen der parametrischen und topologischen Einhüllenden eines bekannten Konstruktionsproblems. In diesem Raum befinden sich alle möglichen (und auch unmöglichen) Konstruktionen, die theoretisch mit dem vorhandenen Wissen und den zur Verfügung stehenden Elementen und Wirkprinzipien überhaupt entwickelt werden könnten. Ob der schier unendlichen Größe dieses Raumes ist es in endlicher Zeit nicht möglich, eine vollständige Exploration zu erreichen (siehe „Kombinatorische Explosion“ im vorangegangenen Abschnitt). In diesem Zusammenhang könnte durch den Einsatz von Mechanismen der mathematischen oder heuristischen Optimierung sowie der künstlichen Intelligenz eine zielorientiertere Einschränkung des Lösungsraum maschinell sichergestellt werden, ohne wichtige Bereiche des Entwurfsraums auszulassen.

Die Gefahr besteht, dass in einem vollständig automatisierten Prozess, in den lediglich am Anfang durch die Vorgabe von Anforderungen oder am Ende durch die Auswahl eines Lösungskonzepts durch den Menschen eingegriffen werden kann, eine Lösung entspringt, die der Mensch nicht mehr in der Lage ist zu fassen und einzuordnen. Dies erinnert an den Zustand im Buch *Per Anhalter durch die Galaxis*, als ein Supercomputer auf die Frage „nach dem Leben, dem Universum und dem ganzen Rest“ (englisch „life, the universe and everything“), mit „42“ antwortet und niemand etwas mit dieser Antwort anzufangen weiß. Bis es allerdings so weit kommen könnte, werden noch viele Arbeiten auf diesem Gebiet verfasst werden müssen.

Ob KOLLER[6] Recht behalten wird, dass zukünftig nur bekannte Konstruktionsaufgaben automatisiert werden können, wird sich zeigen. Die hier vorgestellte Systematik basiert jedenfalls in großen Teilen darauf. Trotzdem können durch Emergenz auch neuartige Dinge entstehen, ohne dass diese vorher bekannt gewesen sein müssten (siehe Paragraf „Innovationsfähigkeit automatisierter Konstruktionsrahmenwerke“ in Abschnitt 5.4). Wir dürfen gespannt sein, zu welchen neuen Erkenntnissen diese Tatsache zukünftig noch führen wird.

[3] An dieser Stelle seien die beiden Standards [ISO 21838-1] und [ISO 21838-2] angeführt.

[4] „*Das ‚Semantic Web‘ ist eine Vision, bei der Daten im Web so definiert und verknüpft werden, dass sie von Maschinen nicht nur zur Anzeige, sondern auch zur Automatisierung, Integration und Wiederverwendung von Daten in verschiedenen Anwendungen genutzt werden können.*“ [Kashyap et al., 2008, S. 1]

[5] Protégé ist nach eigener Darstellung ein freier, quelloffener Ontologie-Editor und ein Framework für den Aufbau intelligenter Systeme. Für weitere Informationen sei auf `https://protege.stanford.edu/` verwiesen.

[6] Hier ist die von KOLLER vertretene Position in seinem Buch [Koller, 1998] gemeint.

Literaturverzeichnis

Dieses Literaturverzeichnis umfasst insgesamt 269 alphabetisch sortierte Einträge. Verschiedene Werke des selben Autors sind nach Erscheinungsjahr aufsteigend sortiert. Für eine bessere Übersicht sind die Normen und Richtlinien von den restlichen Quellen separiert aufgeführt.

[Alber und Rudolph, 2002]
Rolf Alber und Stephan Rudolph (2002). „On a Grammar-Based Design Language That Supports Automated Design Generation and Creativity“. In: Hrsg. von Jonathan C. Borg, Philip J. Farrugia und Kenneth P. Camilleri. Boston, MA: Springer US, S. 19–35. ISBN: 978-0-387-35708-9. DOI: 10.1007/978-0-387-35708-9_2 (zitiert auf Seite 34).

[Alber und Rudolph, 2003]
Rolf Alber und Stephan Rudolph (2003). „‚43‘ – A Generic Approach for Engineering Design Grammars“. In: *AAAI Spring Symposium Technical Report SS-03-02* (zitiert auf Seite 43).

[Albers et al., 2021]
Albert Albers, Joshua Fahl, Tobias Hirschter und Simon Rapp (Mai 2021). „Application of the generic variation operator in the model of PGE – product generation engineering onto the element types of properties and functions of technical systems“. In: *Procedia CIRP* 100, S. 870–875. DOI: 10.1016/j.procir.2021.05.029 (zitiert auf Seite 54).

[Aldanondo und Vareilles, 2008]
Michel Aldanondo und Élise Vareilles (Okt. 2008). „Configuration for mass customization: how to extend product configuration towards requirements and process configuration“. In: *Journal of Intelligent Manufacturing* 19.5, p. 521–535. DOI: 10.1007/s10845-008-0135-z. URL: https://hal.archives-ouvertes.fr/hal-01599442 (zitiert auf den Seiten 50, 51).

[Alexander et al., 1977]
Christopher Alexander, Sara Ishikawa und Murray Silverstein (1977). *A Pattern Language: Towns, Buildings, Construction*. Center for Environmental Structure Berkeley, California. OUP USA. ISBN: 9780195019193. URL: https://books.google.de/books?id=hwAHmktpk5IC (zitiert auf Seite 19).

[Alfaris, 2009]
Anas Alfaris (Nov. 2009). „Emergence through conflict: the Multi-Disciplinary Design System (MDDS)“. Dissertation. Massachusetts Institute of Technology (MIT) (zitiert auf den Seiten 48, 49).

[Anacker et al., 2020]
Harald Anacker, Roman Dumitrescu, Aschot Kharatyan und Andre Lipsmeier (Mai 2020). „Pattern based systems engineering -– application of solution patterns in the design of intelligent technical systems“. In: *Proceedings of the Design Society: DESIGN Conference* 1, S. 1195–1204. DOI: 10.1017/dsd.2020.107 (zitiert auf den Seiten 52, 53).

[Anderl und Trippner, 2000]
Reiner Anderl und Dietmar Trippner, Hrsg. (2000). *STEP – Standard for the Exchange of Product Model Data: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*. Stuttgart: Teubner. ISBN: 3-519-06377-8 (zitiert auf Seite 109).

[Angele et al., 1992]
Jürgen Angele, Dieter Fensel, Dieter Landes, Susanne Neubert und Rudi Studer (1992). „Model-Based and Incremental Knowledge Engineering: The MIKE Approach“. In: *Extended Papers from the IFIP TC12 Workshop on Artificial Intelligence from the Information Processing Perspective: Knowledge Oriented Software Design*. AIFIPP '92, S. 139–168 (zitiert auf den Seiten 48, 49).

[Antonsson und Cagan, 2001]
Erik K. Antonsson und Jonathan Cagan, Hrsg. (2001). *Formal Engineering Design Synthesis*. Cambridge University Press. DOI: 10.1017/CBO9780511529627 (zitiert auf den Seiten 33, 45).

[Anwer und Mathieu, 2016]
Nabil Anwer und Luc Mathieu (2016). „From reverse engineering to shape engineering in mechanical design“. In: *CIRP Annals* 65.1, S. 165–168. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2016.04.052 (zitiert auf den Seiten 52, 53).

[Arana et al., 2001]
Inés Arana, Hatem Ahriz und Pat Fothergill (2001). „Redesign Knowledge Analysis, Representation and Reuse“. In: Hrsg. von Rajkumar Roy. London: Springer London, S. 139–146. ISBN: 978-1-4471-0351-6. DOI: 10.1007/978-1-4471-0351-6_9 (zitiert auf den Seiten 52, 53).

[Arnold und Rudolph, 2012]
Peter Arnold und Stephan Rudolph (2012). „Bridging the Gap between Product Design and Product Manufacturing by Means of Graph-based Design Languages“. In: *Tools and methods of competitive engineering: Proceedings of the Ninth International Symposium on Tools and Methods of Competitive Engineering (TMCE 2012), Karlsruhe, Germany, May 7-11, 2012. Ed.: I. Horváth* (zitiert auf Seite 43).

[Arp et al., 2015]
Robert Arp, Barry Smith und Andrew D. Spear (2015). *Building ontologies with basic formal ontology*. Cambridge, Massachusetts: MIT Press. ISBN: 9780262527811. URL: https://ieeexplore.ieee.org/book/7275982 (zitiert auf den Seiten 130, 131).

[Ashby, 2005]
Michael F. Ashby (2005). *Materials selection in mechanical design*. 3. ed. Amsterdam: Elsevier, XIV, 603 Seiten. ISBN: 0-7506-6168-2 (zitiert auf den Seiten 26, 46, 47, 56, 71, 77).

[Ashino, 2010]
Toshihiro Ashino (Juli 2010). „Materials Ontology: An Infrastructure for Exchanging Ma-

terials Information and Knowledge“. In: *Data Science Journal - DATASCIENCE* 9. DOI: 10.2481/dsj.008-041 (zitiert auf Seite 77).

[Aßmann, 2003]
Uwe Aßmann (2003). „Automatic Roundtrip Engineering“. In: *Electronic Notes in Theoretical Computer Science* 82.5. SC 2003, Workshop on Software Composition (Satellite Event for ETAPS 2003), S. 33–41. ISSN: 1571-0661. DOI: https://doi.org/10.1016/S1571-0661(04)80732-1 (zitiert auf Seite 52).

[Bajzek et al., 2021]
Matthias Bajzek, Johannes Fritz und Hannes Hick (2021). *Systems Engineering Principles*. Hrsg. von Hannes Hick, Klaus Küpper und Helfried Sorger. Cham: Springer International Publishing, S. 149–194. ISBN: 978-3-319-99629-5. DOI: 10.1007/978-3-319-99629-5_7. URL: https://doi.org/10.1007/978-3-319-99629-5_7 (zitiert auf Seite 45).

[Baxter et al., 2008]
David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane und Shilpa Dani (2008). „A framework to integrate design knowledge reuse and requirements management in engineering design“. In: *Robotics and Computer-Integrated Manufacturing* 24.4. ICMR2005: Third International Conference on Manufacturing Research, S. 585–593. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2007.07.010 (zitiert auf den Seiten 52, 53).

[Beilstein, 2011]
Laura Beilstein (2011). „Eine Methodik zur analytischen Gewichtsabschätzung und Bewertung von Strukturverbindungen im Flugzeugvorentwurf“. Dissertation. Institut für Flugzeugbau, Universität Stuttgart (zitiert auf den Seiten 46, 47).

[Blessing und Chakrabarti, 2009]
Luciënne T. M. Blessing und Amaresh Chakrabarti (2009). *DRM, a Design Research Methodology*. Springer, London. ISBN: 978-1-84882-587-1. DOI: https://doi.org/10.1007/978-1-84882-587-1 (zitiert auf den Seiten 13, 14).

[Bonsdorf et al., 1978]
Eero Bonsdorf, Karl Fabel und Olavi Riihimaa (1978). *Schach und Zahl*. 3. Aufl. Walter Rau Verlag. ISBN: 978-3925691317 (zitiert auf den Seiten 56, 129).

[Brambilla et al., 2017]
Marco Brambilla, Jordi Cabot und Manuel Wimmer (2017). *Model-driven software engineering in practice*. Second edition. Synthesis lectures on software engineering 4. San Rafael, Calif.: Morgan & Claypool Publishers. ISBN: 9781627057080 (zitiert auf den Seiten 20, 21).

[Brechet et al., 2001]
Yves Brechet, David Bassetti, Didier Landru und Luc Salvo (2001). „Challenges in materials and process selection“. In: *Progress in Materials Science* 46.3, S. 407–428. ISSN: 0079-6425. DOI: https://doi.org/10.1016/S0079-6425(00)00019-0 (zitiert auf den Seiten 46, 47).

[Broy, 2019]
Manfred Broy (2019). *Logische und Methodische Grundlagen der Programm- und Systementwicklung: Datenstrukturen, funktionale, sequenzielle und objektorientierte Programmierung*. Deutsch. Springer eBooks. Computer Science and Engineering. Wiesbaden: Springer Vieweg. ISBN: 9783658263027. URL: https://doi.org/10.1007/978-3-658-26302-7 (zitiert auf Seite 18).

[Bschorer et al., 2021]
Sabine Bschorer, Konrad Költzsch und Thomas Buck (2021). *Technische Strömungslehre: mit 262 Aufgaben und 31 Beispielen : weiterentwickelt aus dem Lehrbuch von Leopold Böswirth*. 12., überarbeitete und ergänzte Auflage. Springer eBook Collection. Wiesbaden: Springer Vieweg. ISBN: 9783658304072. URL: https://doi.org/10.1007/978-3-658-30407-2 (zitiert auf Seite 71).

[Camba et al., 2016]
Jorge D. Camba, Manuel Contero und Pedro Company (2016). „Parametric CAD modeling: An analysis of strategies for design reusability“. In: *Computer-Aided Design* 74, S. 18–31. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2016.01.003 (zitiert auf den Seiten 52, 53).

[Chakrabarti und Blessing, 2015]
Amaresh Chakrabarti und Luciënne T. M. Blessing (Jan. 2015). „A review of theories and models of design“. In: *Journal of the Indian Institute of Science* 95, S. 325–340. URL: http://journal.library.iisc.ernet.in/index.php/iisc/article/view/4582/4878 (zitiert auf Seite 25).

[Chikofsky und Cross, 1990]
E. J. Chikofsky und J. H. Cross (1990). „Reverse engineering and design recovery: a taxonomy“. In: *IEEE Software* 7.1, S. 13–17. DOI: 10.1109/52.43044 (zitiert auf den Seiten 23, 52).

[Choudry, 2019]
Saphir A. Choudry (2019). „Multidimensionale Bewertung von Fügetechnologien“. Dissertation. Universität Chemnitz. URL: https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-353763 (zitiert auf den Seiten 46, 47).

[Curran et al., 2010]
Richard Curran, Wim J.C. Verhagen, Michel J.L. van Tooren und Ton. H. van der Laan (2010). „A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD“. In: *Expert Systems with Applications* 37.11. Advances in Aligning Knowledge Systems, Improving Business Logistics, Driving Innovation and Adapting Customer Centric Services, S. 7336–7350. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.04.027 (zitiert auf Seite 48).

[Cybenko et al., 1996]
George Cybenko, Aditya Bhasin und Kurt D. Cohen (1996). „Pattern Recognition of 3D CAD Objects: Towards an Electronic Yellow Pages of Mechanical Parts“. In: *Smart Engineering Systems Design, Volume 1* (zitiert auf den Seiten 62, 112).

[Das und Swain, 2016]
Shantanu Kumar Das und Abinash Kumar Swain (2016). „Knowledge-Based Application of Liaison for Variant Design“. In: *Product Lifecycle Management for Digital Transformation of Industries*. Hrsg. von Ramy Harik, Louis Rivest, Alain Bernard, Benoit Eynard und Abdelaziz Bouras, S. 365–374 (zitiert auf den Seiten 46, 47).

[Das und Swain, 2020]
Shantanu Kumar Das und Abinash Kumar Swain (2020). „An Ontology-Based Framework for Decision Support in Assembly Variant Design“. In: *Journal of Computing and Information Science in Engineering*. DOI: https://doi.org/10.1115/1.4048127 (zitiert auf Seite 51).

[de Baas, 2017]
Anne F. de Baas (2017). *What makes a material function? Let me compute the ways: modelling in H2020 LEIT-NMBP programme materials and nanotechnology projects*. ISBN: 978-92-79-63185-6. DOI: 10.2777/417118. URL: https://op.europa.eu/en/publication-detail/-/publication/ec1455c3-d7ca-11e6-ad7c-01aa75ed71a1 (zitiert auf Seite 77).

[Du et al., 2018]
Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama und Wojciech Matusik (Dez. 2018). „InverseCSG: Automatic Conversion of 3D Models to CSG Trees“. In: *ACM Trans. Graph.* 37.6. ISSN: 0730-0301. DOI: 10.1145/3272127.3275006 (zitiert auf den Seiten 52, 53).

[Durhuus und Eilers, 2014]
Bergfinnur Durhuus und Søren Eilers (2014). „On the entropy of LEGO®“. In: *Journal of Applied Mathematics and Computing*. DOI: https://doi.org/10.1007/s12190-013-0730-9 (zitiert auf Seite 12).

[Dym und Brown, 2012]
Clive L. Dym und David C. Brown (2012). *Engineering Design: Representation and Reasoning*. 2. Aufl. Cambridge University Press. DOI: 10.1017/CBO9781139031813 (zitiert auf den Seiten 7, 44, 45).

[Eckert et al., 2003]
Claudia M. Eckert, P. John Clarkson und Martin Stacey (2003). „The Spiral of Applied Research: a Methodological View on Integrated Design Research“. In: *14th International Conference on Engineering Design (ICED03)*, S. 245–246 (zitiert auf Seite 13).

[Eden et al., 2018]
Amnon H. Eden, Epameinondas Gasparis, John Nicholson und Rick Kazman (2018). „Round-trip engineering with the Two-Tier Programming Toolkit“. In: *Software Quality Journal*. DOI: 10.1007/s11219-017-9363-9 (zitiert auf Seite 22).

[Ehrlenspiel und Meerkamm, 2013]
Klaus Ehrlenspiel und Harald Meerkamm (2013). *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit*. 5., überarb. und erw. Aufl., [elektronische

Ressource]. Hanser eLibrary. München: Hanser Verlag. ISBN: 9783446436275 (zitiert auf den Seiten 7, 8, 30, 56, 78, 79, 128, 130).

[Ehrlenspiel und Meerkamm, 2017]
Klaus Ehrlenspiel und Harald Meerkamm (2017). *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit*. 6., vollständig überarbeitete und erweiterte Auflage. Müchen ; Wien: Hanser, XXVIII, 991 Seiten. ISBN: 9783446440890 (zitiert auf den Seiten 29, 45).

[Eigner et al., 2017]
Martin Eigner, Christian Muggeo und Walter Koch (2017). *Modellbasierter Entwicklungsprozess cybertronischer Systeme: Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme*. SpringerLink : Bücher. Berlin, Heidelberg: Springer Vieweg. ISBN: 978-3-662-55124-0. URL: http://dx.doi.org/10.1007/978-3-662-55124-0 (zitiert auf den Seiten 34, 45).

[Eigner et al., 2014]
Martin Eigner, Daniil Roubanov und Radoslav Zafirov (2014). *Modellbasierte virtuelle Produktentwicklung*. Springer eBook Collection. Berlin, Heidelberg: Springer Vieweg. ISBN: 9783662438169. URL: http://dx.doi.org/10.1007/978-3-662-43816-9 (zitiert auf den Seiten 25, 31).

[Eilebrecht und Starke, 2019]
Karl Eilebrecht und Gernot Starke (2019). *Patterns kompakt: Entwurfsmuster für effektive Softwareentwicklung*. 5., aktualisierte und erweiterte Auflage. Springer eBook Collection. Berlin: Springer. ISBN: 9783662579374. URL: https://doi.org/10.1007/978-3-662-57937-4 (zitiert auf Seite 19).

[Esawi und Ashby, 2004]
Amal M.K. Esawi und Michael F. Ashby (2004). „Computer-based selection of joining processes: Methods, software and case studies“. In: *Materials & Design* 25.7, S. 555–564. ISSN: 0261-3069. DOI: http://dx.doi.org/10.1016/j.matdes.2004.03.002 (zitiert auf den Seiten 46, 47).

[Eversheim und Schuh, 2005]
Walter Eversheim und Günther Schuh (2005). *Integrierte Produkt- und Prozessgestaltung*. VDI-Buch : SpringerLink : Bücher. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-26946-5. URL: http://dx.doi.org/10.1007/b138348 (zitiert auf den Seiten 44, 45).

[Farin, 1993]
Gerald Farin (1993). „Chapter 20 - Coons Patches“. In: Hrsg. von Gerald Farin. Third Edition. Boston: Academic Press, S. 363–373. ISBN: 978-0-12-249052-1. DOI: https://doi.org/10.1016/B978-0-12-249052-1.50025-8 (zitiert auf Seite 72).

[Farin, 2006]
Gerald Farin (2006). „Class A Bézier curves“. In: *Computer Aided Geometric Design* 23.7, S. 573–581. ISSN: 0167-8396. DOI: https://doi.org/10.1016/j.cagd.2006.03.004 (zitiert auf Seite 71).

[Fink, 2017]
Andreas Fink (2017). „Untersuchungen zur integrierten Missions- und Fahrzeugbetrachtung mit graphenbasierten Entwurfssprachen am Beispiel eines Space Tug“. Dissertation. Institut für Raumfahrtsysteme, Universität Stuttgart. ISBN: 9783843932059 (zitiert auf Seite 86).

[Fitch und Cooper, 2005]
Peder Fitch und Joyce Cooper (Jan. 2005). „Life-cycle modeling for adaptive and variant design. Part 1: Methodology“. In: *Research in Engineering Design* 15, S. 216–228. DOI: 10.1007/s00163-004-0055-7 (zitiert auf den Seiten 49, 51).

[Forster et al., 1996]
Jörg Forster, Inés Arana und Pat Fothergill (1996). „Re-design knowledge representation with DEKLARE - A domain model oriented, industrial KA framework applied to VT“. In: *Proceedings of KEML 1996 6th workshop on knowledge engineering: methods & languages, Paris, France* (zitiert auf den Seiten 48, 49).

[Forster et al., 1997]
Jörg Forster, Pat Fothergill und Inés Arana (1997). „Enabling intelligent variant design using constraints“. In: *Colloquium on Intelligent Design Systems (Digest No. 1997/016)* (zitiert auf den Seiten 49, 51).

[Franke, 1976]
Hans-Joachim Franke (1976). „Untersuchungen zur Algorithmisierbarkeit des Konstruktionsprozesses“. Dissertation. Düsseldorf, III, 205 S. ISBN: 3181447013 (zitiert auf Seite 10).

[Franke, 2002]
Hans-Joachim Franke (2002). *Variantenmanagement in der Einzel- und Kleinserienfertigung: mit 33 Tabellen*. Deutsch. Hanser, XXIII, 240 Seiten. ISBN: 9783446217300 (zitiert auf Seite 31).

[Fritz, 2018]
A. Herbert Fritz (2018). *Fertigungstechnik*. 12., neu bearbeitete und ergänzte Auflage. Springer-Lehrbuch. Berlin, Heidelberg: Springer Vieweg. ISBN: 978-3-662-56535-3. URL: http://dx.doi.org/10.1007/978-3-662-56535-3 (zitiert auf Seite 72).

[Fritzsche und Oks, 2018]
Albrecht Fritzsche und Sascha Julian Oks (2018). *The Future of Engineering: Philosophical Foundations, Ethical Problems and Application Cases*. Philosophy of engineering and technology ; v. 31. Springer. ISBN: 3319910280. DOI: https://doi.org/10.1007/978-3-319-91029-1 (zitiert auf Seite 56).

[Gamma, 1992]
Erich Gamma (1992). „Objektorientierte Software-Entwicklung am Beispiel von ET++: Design-Muster, Klassenbibliothek, Werkzeuge“. Deutsch. Dissertation. Berlin; Heidelberg [u.a.], XII, 193 Seiten. ISBN: 3540560068 (zitiert auf Seite 19).

[Gamma et al., 1995]
Erich Gamma, Richard Helm, Ralph Johnson und John M. Vlissides (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley professional computing series. Reading, Mass. [u.a.]: Addison-Wesley. ISBN: 9780201633610 (zitiert auf Seite 19).

[Ganter et al., 2000]
Bernhard Ganter, erd Stumme und Rudolf Wille (2000). *Begriffliche Wissensverarbeitung: Methoden und Anwendungen*. 1. Aufl. Springer-Verlag Berlin Heidelberg. ISBN: 9783540663911. URL: https://doi.org/10.1007/978-3-642-57217-3 (zitiert auf Seite 38).

[Ganter und Wille, 1996]
Bernhard Ganter und Rudolph Wille (1996). *Formale Begriffsanalyse*. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-642-61450-7. DOI: 10.1007/978-3-642-61450-7 (zitiert auf Seite 38).

[Gembarski, 2018]
Paul Gembarski (Dez. 2018). „Komplexitätsmanagement mittels wissensbasiertem CAD - Ein Ansatz zum unternehmenstypologischen Management konstruktiver Lösungsräume“. Dissertation. Institut für Produktentwicklung & Gerätebau (iPeG), Leibniz Universität Hannover. ISBN: 978-3-95900-247-9 (zitiert auf den Seiten 34, 48, 49).

[Gembarski et al., 2016]
Paul Gembarski, Mehdi Bibani und Roland Lachmayer (Mai 2016). „Design Catalogues: Knowledge Repositories for Knowledge-Based-Engineering Applications“. In: *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference* (zitiert auf Seite 112).

[Gembarski und Lachmayer, 2018]
Paul Gembarski und Roland Lachmayer (Sep. 2018). „Complexity Management of Solution Spaces in Mass Customization“. In: *8th International Conference on Mass Customization and Personalization – Community of Europe (MCP - CE 2018)* (zitiert auf Seite 33).

[Gembarski et al., 2017]
Paul Gembarski, Haibing Li und R. Lachmayer (Sep. 2017). „Template-based modelling of structural components“. In: *International Journal of Mechanical Engineering and Robotics Research* 6, S. 336–342. DOI: 10.18178/ijmerr.6.5.336-342 (zitiert auf Seite 51).

[Gericke und Eisenbart, 2017]
Kilian Gericke und Boris Eisenbart (Nov. 2017). „The integrated function modeling framework and its relation to function structures“. In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 31, S. 436–457. DOI: 10.1017/S089006041700049X (zitiert auf Seite 54).

[Gero, 1990]
John S. Gero (Dez. 1990). „Design Prototypes: A Knowledge Representation Schema for Design“. In: *AI Magazine* 11.4, S. 26. DOI: 10.1609/aimag.v11i4.854. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/854 (zitiert auf den Seiten 32, 34).

[Gethmann et al., 2022]
Carl Friedrich Gethmann, Peter Buxmann und Julia Distelrath (2022). *Künstliche Intelligenz in der Forschung: neue Möglichkeiten und Herausforderungen für die Wissenschaft*. Ethics of science and technology assessment; Band 48. Berlin: Springer. ISBN: 9783662634493. URL: https://doi.org/10.1007/978-3-662-63449-3 (zitiert auf Seite 133).

[Gierer, 1985]
Alfred Gierer (1985). *Die Physik, Das Leben Und Die Seele*. Piper, München. URL: http://edoc.bbaw.de/volltexte/2006/105/pdf/20rGXUFzQvc.pdf (zitiert auf Seite 129).

[Glock, 1997]
Friedrich Glock (1997). *Zur Soziologie des Konstruierens*. Technische Universität Wien, Institut Technik und Gesellschaft. - Serie: WZB Discussion Paper., S. 97–114 (zitiert auf Seite 80).

[Goll, 2014]
Joachim Goll (2014). *Architektur- und Entwurfsmuster der Softwaretechnik: Mit lauffähigen Beispielen in Java*. Vieweg + Teubner Verlag. ISBN: 9783658055318. DOI: 10.1007/978-3-658-05532-5 (zitiert auf den Seiten 18, 19).

[Gordon, 1969]
William J. Gordon (1969). „Spline-Blended Surface Interpolation Through Curve Networks". In: *Journal of Mathematics and Mechanics* 18.10, S. 931–952. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24902095 (zitiert auf Seite 72).

[Gori et al., 2017]
Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr und Tao Ju (2017). „FlowRep: Descriptive Curve Networks for Free-Form Design Shapes". In: *ACM Transaction on Graphics* 36.4. DOI: http://dx.doi.org/10.1145/3072959.3073639 (zitiert auf den Seiten 52, 53).

[Grabowski et al., 1996]
Hans Grabowski, Stefan Rude und G. Langlotz (1996). „Die Funktionsmodellierung auf dem Weg zur Wissensverarbeitung in CAD-Systemen". In: *CAD '96: Verteilte und intelligente CAD-Systeme, GI-Fachtagung, Kaiserslautern, 7./8. März 1996*. Hrsg. von Detlev Ruland, S. 3–16. URL: https://dblp.org/rec/conf/cad/GrabowskiRL96.bib (zitiert auf Seite 8).

[Groß, 2014]
Johannes Groß (2014). „Aufbau und Einsatz von Entwurfssprachen zur Auslegung von Satelliten". Dissertation. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart. URL: http://dx.doi.org/10.18419/opus-3938 (zitiert auf den Seiten 32, 53, 54, 86).

[Groß und Rudolph, 2015]
Johannes Groß und Stephan Rudolph (Nov. 2015). „Modeling graph-based satellite design languages". In: *Aerospace Science and Technology* 49. DOI: 10.1016/j.ast.2015.11.026 (zitiert auf Seite 43).

[Gruber, 1993]
Thomas R. Gruber (1993). „A translation approach to portable ontology specifications". In: *Knowledge Acquisition* 5.2, S. 199–220. ISSN: 1042-8143. DOI: https://doi.org/10.1006/knac.1993.1008 (zitiert auf Seite 37).

[Haberfellner et al., 2019]
Reinhard Haberfellner, Olivier de Weck, Ernst Fricke und Siegfried Voessner (Jan. 2019). *Systems Engineering, Fundamentals and Applications*. ISBN: 978-3-030-13430-3. DOI: 10.1007/978-3-030-13431-0 (zitiert auf den Seiten 32, 40, 45).

[Heina, 2013]
Jürgen Heina (2013). *Variantenmanagement: Kosten-Nutzen-Bewertung zur Optimierung der Variantenvielfalt*. Gabler Edition Wissenschaft. Deutscher Universitätsverlag. ISBN: 978-3-663-09093-9. DOI: https://doi.org/10.1007/978-3-663-09093-9 (zitiert auf Seite 30).

[Helbig, 2020]
Hermann Helbig (2020). *Welträtsel aus Sicht der modernen Wissenschaften: Emergenz in Natur, Gesellschaft, Psychologie, Technik und Religion*. 2. Auflage. Sachbuch. Berlin: Springer. ISBN: 9783662607626. DOI: http://dx.doi.org/10.1007/978-3-662-60762-6 (zitiert auf Seite 61).

[Helms, 2013]
Bergen Helms (2013). „Object-oriented graph grammars for computational design synthesis“. Dissertation. Technische Universität München (zitiert auf den Seiten 48, 49).

[Henriksson und Larsson, 2003]
Anders Henriksson und Henrik Larsson (2003). „A Definition of Round-Trip Engineering“. In: *Department of Computer and Information Science, Linköpings Universitet, SE-581 83 Linköping, Sweden* (zitiert auf Seite 24).

[Hesse, 1975]
Hermann Hesse (1975). *Siddhartha*. 3. Aufl. Frankfurt: Suhrkamp Verlag. ISBN: 978-3-51801-227-7 (zitiert auf Seite 55).

[Hooshmand, 2014]
Amir Hooshmand (März 2014). „Solving Engineering Design Problems through a Combination of Generative Grammars and Simulations“. Dissertation. Technische Universität München (zitiert auf den Seiten 53, 54).

[Hoschek, 1994]
Josef Hoschek (1994). „Parametric and variational design“. In: *Teubner*, 175 S (zitiert auf Seite 51).

[Houldcroft, 1990]
Peter Houldcroft (1990). *Which Process?* Woodhead Publishing Series in Welding and Other Joining Technologies. Woodhead Publishing. ISBN: 978-1-85573-008-3. DOI: https://doi.org/10.1533/9781845698959.frontmatter (zitiert auf den Seiten 45, 47, 76).

[Hromkovič, 2011]
Juraj Hromkovič (2011). *Theoretische Informatik: Formale Sprachen, Berechenbarkeit, Komplexitätstheorie,Algorithmik, Kommunikation und Kryptographie*. 4., aktualisierte Auflage. SpringerLink. Bücher. Vieweg+Teubner. ISBN: 9783834898531. URL: http://dx.doi.org/10.1007/978-3-8348-9853-1 (zitiert auf Seite 12).

[Hu, 1962]
Ming-Kuei Hu (1962). „Visual pattern recognition by moment invariants“. In: *IRE Transactions on Information Theory* 8.2, S. 179–187. DOI: 10.1109/TIT.1962.1057692 (zitiert auf Seite 62).

[Hubka, 1984]
Vladimir Hubka (1984). *Theorie technischer Systeme: Grundlagen einer wissenschaftlichen Konstruktionslehre*. Hochschultext. Berlin ; Heidelberg ; New York: Springer-Verlag GmbH, XIV, 211 Seiten. ISBN: 9783540129530 (zitiert auf den Seiten 25, 29, 44, 45, 128).

[Huckle und S. Schneider, 2006]
Thomas Huckle und Stefan Schneider, Hrsg. (2006). *Numerische Methoden: Eine Einführung für Informatiker, Naturwissenschaftler, Ingenieure und Mathematiker*. 2. Auflage. SpringerLink. Bücher. Online-Ressource (XIV, 385 Seiten 103 Abb, digital). Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 3540303162 (Druck-Ausgabe). URL: http://dx.doi.org/10.1007/3-540-30318-9 (zitiert auf Seite 63).

[Isaksson et al., 2019]
Ola Isaksson, Claudia Eckert, Olivia Borgue, Sophie Hallstedt, Andreas M. Hein, Kilian Gericke, Massimo Panarotto, Yoram Reich und Anna Rönnbäck (Juli 2019). „Perspectives on Innovation: The Role of Engineering Design“. In: *Proceedings of the Design Society: International Conference on Engineering Design* 1, S. 1235–1244. DOI: 10.1017/dsi.2019.129 (zitiert auf Seite 45).

[Johansson, 2019]
Christian Johansson (Juli 2019). „Reverse Engineered Design Automation: Applying Knowledge Based Engineering Techniques to a Case of Automotive Fixtures Design Configuration“. In: *Proceedings of the Design Society: International Conference on Engineering Design* 1, S. 1583–1592. DOI: 10.1017/dsi.2019.164 (zitiert auf Seite 54).

[Kaiser und Rudolph, 2018]
Dennis Kaiser und Stephan Rudolph (2018). „Automated retrieval of arbitrary complex similar CAD-parts based on dimensionless invariants.“ In: *Tools and Methods of Competitive Engineering TMCE 2018, Las Palmas de Gran Canaria* (zitiert auf den Seiten 62, 111, 113, 114).

[Kak, 2003]
Avinash C. Kak (2003). *Programming with objects: a comparative presentation of object-oriented programming with C++ and Java*. Online-Ausg. IEEE Xplore Digital Library. Hoboken, NJ: John Wiley. ISBN: 978-0-470-54714-4. URL: http://ieeexplore.ieee.org/xpl/bkabstractplus.jsp?bkn=5271080 (zitiert auf den Seiten 18, 19).

[Kalibatiene und Vasilecas, 2011]
Diana Kalibatiene und Olegas Vasilecas (Jan. 2011). „Survey on Ontology Languages“. In: *Lecture Notes in Business Information Processing* 90, S. 124–141. DOI: 10.1007/978-3-642-24511-4_10 (zitiert auf Seite 38).

[Kang et al., 2011]
Eunsuk Kang, Ethan Jackson und Wolfram Schulte (2011). „An Approach for Effective

Design Space Exploration“. In: *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*. Hrsg. von Radu Calinescu und Ethan Jackson (zitiert auf den Seiten 9, 34, 50, 51).

[Kashyap et al., 2008]
Vipul Kashyap, Christoph Bussler und Matthew Moran (2008). *The Semantic Web: Semantics for Data and Services on the Web*. 1. Aufl. Springer Publishing Company, Incorporated. ISBN: 9783540764519 (zitiert auf Seite 140).

[Kaspar und Vielhaber, 2018]
Jerome Kaspar und Michael Vielhaber (2018). „Integrated Cross-Component Lightweight and Material-Oriented Development Methodology – The Embodiment Design Cycle“. In: *Procedia CIRP, 28th CIRP Design Conference, Nantes, Frankreich* 70. 28th CIRP Design Conference 2018, 23-25 May 2018, Nantes, France, S. 481–486. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2018.03.001 (zitiert auf Seite 47).

[Kecher et al., 2021]
Christoph Kecher, Ralf Elbern und Torsten T. Will (2021). *UML 2.5: das umfassende Handbuch*. 7., aktualisierte Auflage. Bonn: Rheinwerk Verlag. ISBN: 9783836284493 (zitiert auf Seite 21).

[Kemper und Reimers, 2022]
Gregor Kemper und Fabian Reimers (2022). *Lineare Algebra: Mit einer Einführung in diskrete Mathematik und Mengenlehre*. Springer eBook Collection. 1 Online-Ressource(IX, 375 Seiten 54 Abb., 6 Abb. in Farbe.) Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 9783662637241. URL: http://dx.doi.org/10.1007/978-3-662-63724-1 (zitiert auf Seite 81).

[Kesper, 2012]
Heiner Kesper (2012). „Gestaltung von Produktvariantenspektren mittels matrixbasierter Methoden“. Dissertation. Lehrstuhl für Produktentwicklung der Technischen Universität München (zitiert auf den Seiten 50, 51).

[Kingston, 2007]
John Kingston (Jan. 2007). „Multi-perspective modelling for knowledge management and knowledge engineering“. Dissertation. University of Edinburgh (zitiert auf den Seiten 50, 51).

[Knöös Franzén et al., 2019]
Ludvig Knöös Franzén, Ingo Staack, Christopher Jouannet und Petter Krus (Okt. 2019). „An Ontological Approach to System of Systems Engineering in Product Development“. In: *Linköping Electronic Conference Proceedings*, S. 35–44. DOI: 10.3384/ecp19162004 (zitiert auf Seite 45).

[Knublauch, 2002]
Holger Knublauch (2002). „An Agile Development Methodology for Knowledge-Based Systems Including a Java Framework for Knowledge Modeling and Appropriate Tool Support“. Dissertation. Universität Ulm, Fakultät für Informatik, Abteilung Programmiermethodik

und Compilerbau. URL: http://dx.doi.org/10.18725/OPARU-29 (zitiert auf Seite 39).

[Knublauch und Rose, 2000]
Holger Knublauch und Thomas Rose (2000). „Round-Trip Engineering of Ontologies for Knowledge-based Systems“. In: *Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE), Chicago, IL* (zitiert auf Seite 52).

[Kohli und Krishnamurti, 1989]
Rajeev Kohli und Ramesh Krishnamurti (1989). „Optimal product design using conjoint analysis : Computational complexity and algorithms*“. In: *European Journal of Operational Research 40*, S. 186–195 (zitiert auf Seite 12).

[Koller, 1998]
Rudolf Koller (1998). *Konstruktionslehre für den Maschinenbau: Grundlagen zur Neu- und Weiterentwicklung technischer Produkte mit Beispielen*. 4., neubearb. und erw. Aufl. Springer, XXVI, 692 Seiten. ISBN: 9783642804182 (zitiert auf den Seiten 10, 44, 45, 130, 140).

[Korf, 1985]
Richard E. Korf (1985). „Depth-first iterative-deepening: An optimal admissible tree search“. In: *Artificial Intelligence* 27.1, S. 97–109. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(85)90084-0 (zitiert auf Seite 67).

[Kormeier, 2010]
Thomas Kormeier (2010). „Graphenbasierte Entwurfssprachen zur konsistenten Modellierung und musterbasierten Topologiemodifikation von Faserverbundstrukturen“. Dissertation. Düsseldorf: Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart. ISBN: 9783183430208 (zitiert auf Seite 43).

[Kosiol, 1973]
Erich Kosiol (1973). *Bausteine der Betriebswirtschaftslehre: eine Sammlung ausgewählter Abhandlungen, Aufsätze und Vorträge*. Betriebswirtschaftliche Forschungsergebnisse ; 62. Erschienen: Bd.1 (1973) bis Bd. 2 (1973). Berlin: Duncker & Humblot. ISBN: 3-428-02926-7 (zitiert auf Seite 66).

[Kothari, 2004]
C.R. Kothari (2004). *Research Methodology: Methods and Techniques*. New Age International. ISBN: 9788122415223. URL: https://books.google.de/books?id=hZ9wSHysQDYC (zitiert auf Seite 13).

[Kratzer, 2015]
Martin Kratzer (2015). „Anwendungsspezifische Entwicklung eines proaktiven Konstruktionssystems auf Basis von Softwareagenten“. Dissertation. Institut für Konstruktionstechnik und Technisches Design, Universität Stuttgart. DOI: http://dx.doi.org/10.18419/opus-4593 (zitiert auf den Seiten 48, 49).

[Krish, 2011]
Sivam Krish (2011). „A practical generative design method“. In: *Computer-Aided Design* 43.1, S. 88–100. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2010.09.009 (zitiert auf den Seiten 50, 51).

[Kübler et al., 2020]
Karl Kübler, Dominik Schopper, Oliver Riedel und Stephan Rudolph (2020). „Towards an Automated Product-Production System Design – Combining Simulation-based Engineering and Graph-based Design Languages“. In: *5th International Conference on System-Integrated Intelligence* (zitiert auf Seite 139).

[Kulfan, 2008]
Brenda M. Kulfan (2008). „Universal Parametric Geometry Representation Method“. In: *Journal of Aircraft* 45.1, S. 142–158. URL: https://doi.org/10.2514/1.29958 (zitiert auf Seite 73).

[Kurtoglu et al., 2010]
Tolga Kurtoglu, Albert Swantner und Matthew I. Campbell (2010). „Automating the conceptual design process: “From black box to component selection”“. In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 24.1, S. 49–62. DOI: 10.1017/S0890060409990163 (zitiert auf den Seiten 53, 54).

[L'Eglise et al., 2001]
Thomas L'Eglise, Brahim Rekiek, Pièrre De Lit, Benoît Raucent, Patrick Fouda und Alain Delchambre (2001). „A multicriteria decision-aid system for joining process selection“. In: *Proceedings of the IEEE International Symposium on Assembly and Task Planning*. DOI: 10.1109/ISATP.2001.929043 (zitiert auf den Seiten 46, 47).

[LaRocca, 2011]
Gianfranco LaRocca (2011). „Knowledge based engineering techniques to support aircraft design and optimization“. Dissertation. TU Delft (zitiert auf Seite 39).

[Le Masson et al., 2017]
Pascal Le Masson, Benoit Weil und Armand Hatchuel (2017). *Design Theory: Methods and Organization for Innovation*. Springer. ISBN: 9783319502779. URL: http://dx.doi.org/10.1007/978-3-319-50277-9 (zitiert auf den Seiten 44, 45).

[LeBacq et al., 2002]
Christophe LeBacq, Yves Brechet, Hugh R. Shercliff, Thierry Jeggy und Luc Salvo (2002). „Selection of joining methods in mechanical design“. In: *Materials & Design* 23.4, S. 405–416. DOI: http://dx.doi.org/10.1016/S0261-3069(01)00093-0 (zitiert auf den Seiten 46, 47).

[Leemhuis, 2005]
Helen Leemhuis (2005). „Funktionsgetriebene Konstruktion als Grundlage verbesserter Produktentwicklung“. Dissertation. Berlin: Technische Universität Berlin, Fakultät V - Verkehrs- und Maschinensysteme. DOI: 10.14279/depositonce-932 (zitiert auf Seite 8).

[Lemburg, 2009]
Johannes Peter Lemburg (2009). „Methodik der schrittweisen Gestaltsynthese“. Dissertation. Aachen, 109 S. : Ill., graph. Darst. ISBN: 978-3-8322-8117-5. URL: https://publications.rwth-aachen.de/record/50794 (zitiert auf Seite 33).

[Lenders, 2009]
Michael Lenders (2009). „Beschleunigung der Produktentwicklung durch Lösungsraum-

Management“. Hochschulschrift. Aachen: Werkzeugmaschinenlabor (WZL), Rheinisch-Westfälische-Technische Hochschule Aachen (RWTH Aachen), XIV, 271 Seiten. ISBN: 978-3-940565-26-6 (zitiert auf Seite 51).

[Haibing Li, 2020]

Haibing Li (Apr. 2020). „Generative Design Approach for Robust Solution Development“. Dissertation. Institut für Produktentwicklung & Gerätebau (iPeG), Leibniz Universität Hannover (zitiert auf den Seiten 33, 48, 54, 57).

[Huanyu Li et al., 2020]

Huanyu Li, Rickard Armiento und Patrick Lambrix (2020). „An Ontology for the Materials Design Domain“. In: *CoRR* abs/2006.07712. arXiv: 2006.07712. URL: https://arxiv.org/abs/2006.07712 (zitiert auf Seite 77).

[T. Li et al., 2020]

Tao Li, Helen Lockett und Craig Lawson (2020). „Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration“. In: *Journal of Manufacturing Systems* 54, S. 242–257. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2020.01.001 (zitiert auf Seite 54).

[Lindemann, 2016]

Udo Lindemann (2016). *Handbuch Produktentwicklung*. Hanser eLibrary. Online-Ressource (XXXIII, 1036 Seiten). München: Hanser. ISBN: 9783446445185 (Druck-Ausgabe). URL: http://dx.doi.org/10.3139/9783446445819 (zitiert auf den Seiten 44, 45).

[Lo et al., 2010]

Cheng-Hung Lo, K. Tseng und C. Chu (2010). „One-Step QFD based 3D morphological charts for concept generation of product variant design“. In: *Expert Syst. Appl.* 37, S. 7351–7363 (zitiert auf den Seiten 50, 51).

[Lovatt und Shercliff, 1998a]

Andrew M. Lovatt und Hugh R. Shercliff (1998a). „Manufacturing process selection in engineering design. Part 1: the role of process selection“. In: *Materials & Design* 19.5–6, S. 205–215. ISSN: 0261-3069. DOI: https://doi.org/10.1016/S0261-3069(98)00038-7 (zitiert auf den Seiten 46, 47).

[Lovatt und Shercliff, 1998b]

Andrew M. Lovatt und Hugh R. Shercliff (1998b). „Manufacturing process selection in engineering design. Part 2: a methodology for creating task-based process selection procedures“. In: *Materials & Design* 19.5–6, S. 217–230. ISSN: 0261-3069. DOI: https://doi.org/10.1016/S0261-3069(98)00039-9 (zitiert auf den Seiten 46, 47).

[Lüdtke, 2016]

Bastian Lüdtke (2016). „Lösungsraum-Steuerung in der Produktentwicklung; 1. Auflage“. Deutsche und englische Zusammenfassung. - Weitere Reihe: Produktionssystematik; Dissertation, RWTH Aachen University, 2016. Dissertation. Aachen: RWTH Aachen University, XXIV, 245 Seiten : Diagramme. ISBN: 978-3-86359-443-5. URL: https://publications.rwth-aachen.de/record/672707 (zitiert auf den Seiten 9, 51).

[Maier und Störrle, 2011]
Anja Maier und Harald Störrle (Jan. 2011). „What are the Characteristics of Engineering Desing Processes?“ In: *ICED 11 - 18th International Conference on Engineering Design - Impacting Society Through Engineering Design* 1, S. 188–198 (zitiert auf Seite 29).

[Mainzer, 2019]
Klaus Mainzer (2019). *Künstliche Intelligenz – Wann übernehmen die Maschinen?* 2., erweiterte Auflage. Berlin: Springer. ISBN: 9783662580462. URL: https://doi.org/10.1007/978-3-662-58046-2 (zitiert auf den Seiten 39, 40, 129, 133).

[Malmqvist, 1997]
Johan Malmqvist (1997). „Improved Function-means Trees by Inclusion of Design History Information“. In: *Journal of Engineering Design* 8.2, S. 107–117. DOI: 10.1080/09544829708907955 (zitiert auf Seite 33).

[Mankins, 1995]
John C. Mankins (1995). „Technology Readiness Level – A White Paper“. In: *Advanced Concepts Office – Office of Space Access and Technology NASA*. URL: http://www.artemisinnovation.com/images/TRL_White_Paper_2004-Edited.pdf (zitiert auf Seite 133).

[Mäntylä, 1987]
Martti Mäntylä (1987). *An Introduction to Solid Modeling*. USA: Computer Science Press, Inc. ISBN: 0881751081 (zitiert auf Seite 72).

[Matthiesen, 2002]
Sven Matthiesen (2002). „Ein Beitrag zur Basisdefinition des Elementmodells ‚Wirkflächenpaare & Leitstützstrukturen‘ zum Zusammenhang von Funktion und Gestalt technischer Systeme“. Dissertation. Institut für Maschinenkonstruktionslehre und Kraftfahrzeugbau, Universität Karlsruhe (zitiert auf Seite 33).

[Mavris et al., 1998]
Dimitri Mavris, Daniel DeLaurentis, Oliver Bandte und Mark Hale (1998). *A Stochastic Approach to Multi-disciplinary Aircraft Analysis and Design*. DOI: 10.2514/6.1998-912 (zitiert auf den Seiten 8, 9).

[McMahon, 2021]
Chris McMahon (2021). „Situation, Patterns, Exploration, and Exploitation in Engineering Design“. In: *She Ji: The Journal of Design, Economics, and Innovation* 7.1, S. 71–94. ISSN: 2405-8726. DOI: https://doi.org/10.1016/j.sheji.2020.08.009 (zitiert auf Seite 51).

[Melan, 2003]
Albrecht Melan (2003). „N-dimensionale Merkmalsgewinnung durch vektorielle Dimensionen am Beispiel von Farbbildern und weiteren Anwendungen“. Dissertation. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart (zitiert auf den Seiten 62, 114).

[Mesa et al., 2018]
Jaime Alberto Mesa, Danny Illera, Iván Esparragoza, Heriberto Maury und Humberto

Gómez (2018). „Functional characterisation of mechanical joints to facilitate its selection during the design of open architecture products“. In: *International Journal of Production Research* 56.24, S. 7390–7404. DOI: 10.1080/00207543.2017.1412530 (zitiert auf den Seiten 46, 47).

[Mesa et al., 2015]
Jaime Alberto Mesa, Heriberto Maury, René Arrieta, Antonio Bula und Carles Riba (2015). „Characterization of modular architecture principles towards reconfiguration: a first approach in its selection process“. In: *The International Journal of Advanced Manufacturing Technology* 80, S. 221–232. DOI: https://doi.org/10.1007/s00170-015-6951-3 (zitiert auf den Seiten 52, 53).

[Michalos et al., 2010]
George Michalos, Sotiris Makris, Nikolas Papakostas, Dimitris Mourtzis und G. Chryssolouris (2010). „Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach“. In: *CIRP Journal of Manufacturing Science and Technology* 2.2, S. 81–91. ISSN: 1755-5817. DOI: http://dx.doi.org/10.1016/j.cirpj.2009.12.001 (zitiert auf den Seiten 46, 47).

[Motzer, 2015]
Martin Motzer (2015). „Integrierte Flugzeugrumpf- und Kabinenentwicklung mit graphenbasierten Entwurfssprachen“. Dissertation. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart. DOI: https://doi.org/10.18419/OPUS-3938 (zitiert auf Seite 86).

[Müller, 2018]
Jakob Ramon Müller (2018). „Towards automated conceptual design space exploration: An investigation into the design process of aerospace components“. Litentiate Thesis. Chalmers University of Technology (zitiert auf Seite 51).

[Müller, 2020]
Jakob Ramon Müller (2020). „Does form follow function? – Connectin Function Modelling and Geometry Modelling for Design Space Exploration“. Dissertation. Department of Industrial and Materials Science Chalmers University of Technology (zitiert auf Seite 51).

[Münzer, 2015]
Clemens Münzer (2015). „Constraint-Based Methods for Automated Computational Design Synthesis of Solution Spaces“. Diss. Zürich: ETH Zurich. DOI: 10.3929/ethz-a-010603411 (zitiert auf den Seiten 53, 54).

[Naefe, 2018]
Paul Naefe (2018). *Methodisches Konstruieren: Auf den Punkt gebracht*. 3. Aufl. 2018. SpringerLink : Bücher. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-22636-7. URL: http://dx.doi.org/10.1007/978-3-658-22636-7 (zitiert auf den Seiten 29, 45).

[Navinchandra, 1991]
Dundee J. Navinchandra (1991). *Exploration and Innovation in Design: Towards a Computational Model*. 1. Aufl. Symbolic Computation. Springer. ISBN: 978-0-387-97481-1. URL: https://doi.org/10.1007/978-1-4612-3114-1 (zitiert auf Seite 9).

[North, 2011]
Klaus North (2011). *Wissensorientierte Unternehmensführung: Wertschöpfung durch Wissen*. 5., aktualisierte und erweiterte Auflage. SpringerLink : Bücher. Wiesbaden: Gabler. ISBN: 978-3-8349-6427-4. URL: http://dx.doi.org/10.1007/978-3-8349-6427-4 (zitiert auf Seite 36).

[Noy und McGuinness, 2001]
Natasha F. Noy und Deborah L. McGuinness (Jan. 2001). „Ontology Development 101: A Guide to Creating Your First Ontology“. In: *Knowledge Systems Laboratory* 32. URL: http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf (zitiert auf Seite 37).

[Omar und Soltan, 2020]
Mohamed Omar und Hassan Soltan (2020). „A framework for welding process selection“. In: *SN Applied Sciences*. DOI: https://doi.org/10.1007/s42452-020-2144-2 (zitiert auf den Seiten 46, 47).

[Open Cascade, 2022]
Open Cascade SAS (2022). *Open CASCADE Technology (OCCT) – 3D modeling kernel, freely available in open source.* Zuletzt aufgerufen am 11.02.2022. URL: https://www.opencascade.com/open-cascade-technology/ (zitiert auf den Seiten 72, 105, 122).

[Pahl und Beitz, 2007]
Gerhard Pahl und Wolfgang Beitz (2007). *Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung Methoden und Anwendung*. Hrsg. von Wolfgang, Jörg Feldhusen und Karl-Heinrich Grote. 7. Auflage. SpringerLink. Bücher. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 9783540340614. URL: http://dx.doi.org/10.1007/978-3-540-34061-4 (zitiert auf den Seiten 44, 45, 70).

[Pahl und Beitz, 2013]
Gerhard Pahl und Wolfgang Beitz (2013). *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung*. Hrsg. von Jörg Feldhusen und Karl-Heinrich Grote. Berlin, Heidelberg: Springer. ISBN: 978-3-642-29568-3. DOI: 10.1007/978-3-642-29569-0 (zitiert auf den Seiten 8–10, 27–33, 43, 56, 63, 69, 70).

[Pasini, 2002]
Damiano Pasini (2002). „A new theory for modelling the mass efficiency of material, shape and form“. PhD Thesis. University of Bristol (zitiert auf den Seiten 46, 47).

[Penrose, 1989]
Roger Penrose (1989). *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press. ISBN: 9780198519737. DOI: 10.1093/oso/9780198519737.001.0001 (zitiert auf Seite 133).

[Pfenning, 2017]
Michael Pfenning (2017). „Durchgängiges Engineering durch die Integration von PLM und MBSE“. Dissertation. Kaiserslautern: Technische Universität Kaiserslautern, Lehrstuhl für Virtuelle Produktentwicklung. ISBN: 9783959740609 (zitiert auf Seite 34).

[Ponn und Lindemann, 2011]
Josef Ponn und Udo Lindemann, Hrsg. (2011). *Konzeptentwicklung und Gestaltung technischer Produkte: Systematisch von Anforderungen zu Konzepten und Gestaltlösungen*. SpringerLink : Bücher. Online-Ressource (XIV, 466S, digital). Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-20580-4. URL: http://dx.doi.org/10.1007/978-3-642-20580-4 (zitiert auf den Seiten 33, 44, 45).

[Poorkiany, 2017]
Morteza Poorkiany (2017). „Managing design rationale in the development of product families and related design automation systems“. Diss. School of Engineering, Jönköping University, JTH. Research area Product Development - Computer supported engineering design, S. 100. ISBN: 978-91-87289-32-3 (zitiert auf den Seiten 53, 54).

[Reichwein, 2011]
Axel Reichwein (2011). „Application-specific UML Profiles for Multidisciplinary Product Data Integration“. Dissertation. Institut für Statik- und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, S. 2, 5, 6, 10 (zitiert auf den Seiten 21, 44, 68).

[Reuter, 2021]
Martin Reuter (2021). *Methodik der Werkstoffauswahl: der systematische Weg zum richtigen Material*. 3., aktualisierte Auflage. Hanser eLibrary. 1 Online-Ressource (XI, 287 Seiten). München: Hanser. ISBN: 9783446468542. URL: http://dx.doi.org/10.3139/9783446468542 (zitiert auf Seite 77).

[Riestenpatt gen. Richter, 2021]
Marius Riestenpatt genannt Richter (2021). „Automatische Fehlerbaumerstellung und -analyse zeitinvarianter Netzwerke“. 1 Online-Ressource (x, 155 Seiten). Hochschulschrift. Stuttgart. URL: http://nbn-resolving.de/urn:nbn:de:bsz:93-opus-ds-114966 (zitiert auf Seite 43).

[Riestenpatt gen. Richter und Rudolph, 2020]
Marius Riestenpatt genannt Richter und Stephan Rudolph (Okt. 2020). „A scientific discourse on creativity and innovation in the formal context of graph-based design languages“. In: *Thirtieth anniversary "heron island" conference workshop on computational and cognitive models of creative design (hi'19)* (zitiert auf den Seiten 61, 129, 130).

[Rigger und Vosgien, 2018]
Eugen Rigger und Thomas Vosgien (Jan. 2018). „Design automation state of practice - potential and opportunities“. In: *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, S. 441–452. DOI: 10.21278/idc.2018.0537 (zitiert auf Seite 54).

[Rittel und Webber, 1973]
Horst W. J. Rittel und Melvin M. Webber (1973). „Dilemmas in a general theory of planning“. In: *Policy Sciences*. DOI: https://doi.org/10.1007/BF01405730 (zitiert auf Seite 7).

[Roth, 1996]
Karlheinz Roth (1996). *Konstruieren mit Konstruktionskatalogen*. 2. Aufl., wesentl. erw. und neu gestaltet. Bd. 3. 3. Verbindungen und Verschlüsse, Lösungsfindung : mit 48

Konstruktionskatalogen und 52 Lösungsammlungen. Berlin ; Heidelberg: Springer. ISBN: 9783642872204 (zitiert auf den Seiten 46, 47).

[Rudolph, 1995]
Stephan Rudolph (1995). „Eine Methodik zur systematischen Bewertung von Konstruktionen“. Dissertation. Institut für Statik- und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart (zitiert auf den Seiten 9, 128).

[Rudolph, 2002]
Stephan Rudolph (2002). „Übertragung von Ähnlichkeitsbegriffen“. Habilitationsschrift. Institut für Statik- und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart (zitiert auf den Seiten 25, 35, 41, 42, 82).

[Rudolph, 2006]
Stephan Rudolph (2006). „A semantic validation scheme for graph-based engineering design grammars“. In: *Design Computing and Cognition DCC'6*. Hrsg. von John S. Gero (zitiert auf den Seiten 63, 64).

[Rudolph, 2013]
Stephan Rudolph (2013). „Seiteneffekte – Ursachen, Wirkungen und Konsequenzen für ein ganzheitliches Systems Engineering“. In: *Tag des Systems Engineering*, S. 59–69. DOI: 10.3139/9783446439467.007 (zitiert auf Seite 63).

[Rudolph und Alber, 2002]
Stephan Rudolph und Rolf Alber (2002). „An Evolutionary Approach to the Inverse Problem in Rule-Based Design Representations“. In: Hrsg. von John S. Gero. Springer Netherlands. URL: https://doi.org/10.1007/978-94-017-0795-4_16 (zitiert auf Seite 43).

[Rudolph et al., 2013]
Stephan Rudolph, Johannes Beichter, Marc Eheim, Stefan Hess, Martin Motzer und Roland Weil (2013). „On Multi-Disciplinary Architectural Synthesis and Analysis of Complex Systems with Graph-based Design Languages“. In: *62. Deutscher Luft- und Raumfahrtkongress (DGLR 2013)*. URL: https://idealism.ifb.uni-stuttgart.de/wp-content/uploads/2016/06/On-Multi-Disciplinary-Architectural-Synthesis-and-Analysis-of-COmplex-Systems-with-Graph-based-Design-Languages.pdf (zitiert auf Seite 43).

[Rudolph und Kröplin, 2005]
Stephan Rudolph und Bernd Kröplin (2005). „Entwurfsgrammatiken – Ein Paradigmenwechsel?“ In: *Der Prüfingenieur*, S. 43 (zitiert auf den Seiten 41, 43, 71).

[Rumpe, 2011]
Bernhard Rumpe (2011). *Modellierung mit UML: Sprache, Konzepte und Methodik*. Berlin, Heidelberg: Springer Berlin Heidelberg, Online–Ressource (X, 294S, digital). ISBN: 978-3-642-22413-3. DOI: http://dx.doi.org/10.1007/978-3-642-22413-3 (zitiert auf den Seiten 21, 22).

[Rusitschka, 2017]
Fabian Rusitschka (2017). „Methodik zur Auswahl von lösbaren Verbindungen in der variantenreichen Serienfertigung“. 1 Online-Ressource (VII, 135 Seiten). Hochschulschrift.

Stuttgart: Institut für Konstruktionstechnik und Technisches Design (IKTD), Universität Stuttgart. ISBN: 978-3-946924-00-5 (Druck-Ausgabe). URL: http://d-nb.info/1153769859/34 (zitiert auf den Seiten 46, 47).

[Saake und Sattler, 2021]
Gunter Saake und Kai-Uwe Sattler (2021). *Algorithmen und Datenstrukturen: eine Einführung mit Java*. 6., überarbeitete und erweiterte Auflage. dpunkt.verlag. ISBN: 3969100666. URL: http://www.content-select.com/index.php?id=bib_view&ean=9783969100660 (zitiert auf Seite 81).

[Salvador et al., 2009]
Fabrizio Salvador, Pablo Martin de Holan und Frank Piller (Juli 2009). „Cracking the Code of Mass Customization". In: *MIT Sloan Management Review* 50 (zitiert auf Seite 34).

[Sauthoff, 2017]
Bastian Sauthoff (2017). „Generative parametrische Modellierung von Strukturkomponenten für die technische Vererbung". Dissertation. Institut für Produktentwicklung & Gerätebau (iPeG), Leibniz Universität Hannover. DOI: 978-3-95900-144-1 (zitiert auf den Seiten 53, 54).

[Saxena und Karsai, 2010]
Tripti Saxena und Gabor Karsai (2010). „MDE-Based Approach for Generalizing Design Space Exploration". In: *Model Driven Engineering Languages and Systems*. Hrsg. von Nicolas Petriu Dorina C.and Rouquette und Øystein Haugen, S. 46–60 (zitiert auf den Seiten 33, 50, 51).

[J. Schmidt und Rudolph, 2014]
Jens Schmidt und Stephan Rudolph (2014). „Gaining System Design Knowledge by Systematic Design Space Exploration with Graph-based Design Languages". In: *AIP Conference Proceedings* 1618.1, S. 390–393. DOI: 10.1063/1.4897755. URL: https://aip.scitation.org/doi/abs/10.1063/1.4897755 (zitiert auf den Seiten 50, 51).

[J. Schmidt und Rudolph, 2016]
Jens Schmidt und Stephan Rudolph (2016). „Graph-Based Design Languages: A Lingua Franca for Product Design Including Abstract Geometry". In: *IEEE Computer Graphics and Applications* 36, S. 88–93. DOI: 10.1109/MCG.2016.89 (zitiert auf Seite 43).

[L. C. Schmidt und Cagan, 1997]
Linda C. Schmidt und Jonathan Cagan (1997). „GGREADA: A graph grammar-based machine design algorithm". In: *Research in Engineering Design* 9 (4). DOI: 10.1007/BF01589682 (zitiert auf den Seiten 53, 54).

[E. Schneider, 1934]
Erich Schneider (1934). *Theorie der Produktion*. Wien: J. Springer. URL: https://books.google.de/books?id=2rk5AQAAIAAJ (zitiert auf Seite 66).

[Schöne, 2020]
Philip Jonathan Schöne (2020). „Generatives Design von Hubschrauberzellen mit graphenbasierten Entwurfssprachen". Bachelorarbeit. Institut für Flugzeugbau, Universität Stuttgart (zitiert auf Seite 73).

[Schopper et al., 2021]
Dominik Schopper, Karl Kübler, Stephan Rudolph und Oliver Riedel (2021). „EIPPM - The Executable Integrative Product-Production Model“. In: *Computers* 10.6. ISSN: 2073-431X. DOI: 10.3390/computers10060072. URL: https://www.mdpi.com/2073-431X/10/6/72 (zitiert auf den Seiten 43, 47, 139).

[Schopper und Rudolph, 2018]
Dominik Schopper und Stephan Rudolph (2018). „From Model-Driven Architecture and Model-Based Systems Engineering via Formal Concept Analysis to Graph-Based Design Languages and Back: A Scientific Discourse“. In: *Volume 1B: 38th Computers and Information in Engineering Conference* 51739, V01BT02A043–. DOI: http://dx.doi.org/10.1115/DETC2018-86392 (zitiert auf den Seiten 21, 26, 27, 37, 38, 41, 59–63, 108).

[Schreiber et al., 2001]
Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert Hoog, Nigel Shadbolt, Walter Velde und Bob Wielinga (Jan. 2001). *Knowledge Engineering and Management - The CommonKADS Methodology*. Bd. 24. MIT Press. ISBN: 9780262283236. DOI: 10.7551/mitpress/4073.001.0001 (zitiert auf den Seiten 48, 49).

[Schreiber et al., 1994]
Guus Schreiber, Bob Wielinga, Robert de Hoog, Hans Akkermans und Walter Van de Velde (1994). „CommonKADS: a comprehensive methodology for KBS development“. In: *IEEE Expert* 9.6, S. 28–37. DOI: 10.1109/64.363263 (zitiert auf den Seiten 48, 49).

[Sedchaicharn, 2010]
Korkiat Sedchaicharn (2010). „Eine rechnergestützte Methode zur Festlegung der Produktarchitektur mit integrierter Berücksichtigung von Funktion und Gestalt“. Dissertation. Institut für Produktentwicklung (IPEK) am Karlsruhe Institut für Technologie (KIT). DOI: 10.5445/IR/1000019072 (zitiert auf den Seiten 48, 49).

[Seifert, 2011]
Mirko Seifert (2011). „Designing Round-Trip Systems by Change Propagation and Model Partitioning“. Dissertation. Technische Universität Dresden (zitiert auf Seite 22).

[Sendall und Küster, 2004]
Shane Sendall und Jochen Küster (2004). „Taming Model Round-Trip Engineering“. In: *Proceedings of Workshop on Best Practices for Model-Driven Software Development (satellite event of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)), Vancouver (Canada)* (zitiert auf Seite 22).

[Shah, 2001]
Jami J. Shah (2001). „Designing with Parametric CAD: Classification and comparison of construction techniques“. In: Hrsg. von Fumihiko Kimura. Boston, MA, S. 53–68. ISBN: 978-0-387-35490-3. DOI: 10.1007/978-0-387-35490-3_4. URL: https://doi.org/10.1007/978-0-387-35490-3_4 (zitiert auf den Seiten 49, 51).

[Sharan und Davis, 2022]
Kishori Sharan und Adam L. Davis (2022). *Beginning Java 17 Fundamentals: Object-Oriented Programming in Java 17*. 3rd ed. 2022. Springer eBook Collection. Berkeley, CA: Apress.

ISBN: 9781484273074. URL: http://dx.doi.org/10.1007/978-1-4842-7307-4 (zitiert auf den Seiten 18, 19).

[Shercliff und Lovatt, 2001]
Hugh R. Shercliff und Andrew M. Lovatt (2001). „Selection of manufacturing processes in design and the role of process modelling“. In: *Progress in Materials Science* 46.3, S. 429–459. ISSN: 0079-6425. DOI: https://doi.org/10.1016/S0079-6425(00)00013-X (zitiert auf den Seiten 46, 47).

[Siggel et al., 2019]
Martin Siggel, Jan Kleinert, Tobias Stollenwerk und Reinhold Maierl (2019). „TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design“. In: *Mathematics in Computer Science* 13, S. 367–389. DOI: https://doi.org/10.1007/s11786-019-00401-y (zitiert auf Seite 72).

[Smith, 2018]
Barry Smith (2018). *Part 1: Three ways ontologies fail – with lessons learned for the Navy Systems Engineering Transformation*. Zuletzt geprüft am 20.04.2022. URL: http://ncorwiki.buffalo.edu/index.php/Ontology_and_the_Navy_SYSCOMs_Systems_Engineering_Transformation_Process (zitiert auf Seite 131).

[Smith und Grenon, 2021]
Barry Smith und Pierre Grenon (2021). *Basic Formal Ontology (FBO)*. Zuletzt geprüft am 20.04.2022. URL: https://basic-formal-ontology.org/ (zitiert auf Seite 68).

[Sobek et al., 1999]
Durward Sobek, Allen C. Ward und Jeffrey Liker (Dez. 1999). „Toyota's Principles of Set-Based Concurrent Engineering“. In: *Sloan Management Review* 40. URL: https://sloanreview.mit.edu/article/toyotas-principles-of-setbased-concurrent-engineering/ (zitiert auf Seite 9).

[Sobieszczanski-Sobieski et al., 2015]
Jaroslaw Sobieszczanski-Sobieski, Alan Morris und Michel J.L. Van Tooren (2015). *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons (zitiert auf Seite 49).

[Soddu, 2006]
Celestino Soddu (2006). „Generative designer, a swimmer in a natural sea frame.“ In: *GA2006, the 9th Generative Art Conference*, S. 1–11 (zitiert auf den Seiten 53, 54).

[Stachowiak, 1973]
Herbert Stachowiak (1973). *Allgemeine Modelltheorie*. Wien: Springer (zitiert auf Seite 26).

[J. Stahl, 2008]
Jan Stahl (2008). „Development of a Methodology for Joining Technology Selection based on Cost Information in the Preliminary Automotive Body-in-White Product Development Process“. PhD Thesis. Cranfield University (zitiert auf den Seiten 46, 47).

[T. Stahl und Bettin, 2007]
Thomas Stahl und Jorn Bettin (2007). *Modellgetriebene Softwareentwicklung: Techniken,*

Engineering, Management. 2., aktualisierte und erw. Aufl. Heidelberg: dpunkt-Verl. ISBN: 9783898644488 (zitiert auf Seite 19).

[Stoffels, 2017]
Pascal Dominik Stoffels (2017). „Integrierte Definition von Produkt, Produktion und Material zur Steigerung der Ressourceneffizienz“. Dissertation. Saarbrücken: Lehrstuhl für Konstruktionstechnik, Universität des Saarlandes. DOI: http://dx.doi.org/10.22028/D291-26985 (zitiert auf Seite 47).

[Stokes, 2001]
Melody Stokes (2001). *Managing Engineering Knowledge – Moka: Methodology for Knowledge Based Engineering Applications*. London: Professional Engineering Publishing. ISBN: 978-1-860-58295-0 (zitiert auf den Seiten 39, 48, 49).

[Styczynski et al., 2017]
Zbigniew A. Styczynski, Krzysztof Rudion und André Naumann (2017). *Einführung in Expertensysteme: Grundlagen, Anwendungen und Beispiele aus der elektrischen Energieversorgung*. Berlin, Heidelberg. DOI: http://dx.doi.org/10.1007/978-3-662-53172-3 (zitiert auf den Seiten 37, 39).

[Suh, 1990]
Nam P Suh (1990). *The principles of design*. 6. Oxford University Press on Demand. ISBN: 978-0-1950-4345-7 (zitiert auf Seite 35).

[Suh, 2005]
Nam P Suh (2005). *Complexity: theory and applications*. Oxford University Press on Demand. ISBN: 978-0-1951-7876-0 (zitiert auf den Seiten 49, 51).

[Sullivan, 1896]
Louis H Sullivan (1896). „The Tall Office Building Artistically Considered [1896]“. In: *Kindergarten Chats and Other Writings. New York: Dover Publications* (zitiert auf den Seiten 59, 71).

[Sung et al., 2002]
Raymond C. W. Sung, Heather J. Rea, Jonathan R. Corney, Douglas E. R. Clark und John Pritchard (2002). „Shapesifter: A retrieval system for databases of 3D engineering data“. In: *New Review of Information Networking* 8, S. 33–53 (zitiert auf Seite 112).

[Swain et al., 2014]
Abinash Kumar Swain, Dibakar Sen und B. Gurumoorthy (2014). „Extended liaison as an interface between product and process model in assembly“. In: *Robotics and Computer-Integrated Manufacturing* 30.5, S. 527–545. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2014.02.005 (zitiert auf den Seiten 46, 47).

[Swift und Booker, 1997]
Kenneth G. Swift und Julian D. Booker, Hrsg. (1997). *Process selection: from design to manufacture*. London: Wiley, 214 Seiten. ISBN: 0-470-23774-0 (zitiert auf den Seiten 46, 47, 56).

[Tarkian, 2012]
Mehdi Tarkian (2012). „Design Automation for Multidisciplinary Optimization : A High

Level CAD Template Approach“. Dissertation. Linköping University, Division of Machine Design, Department of Management und Engineering (zitiert auf den Seiten 53, 54).

[Tjalve, 1979]
Eskild Tjalve (1979). *A short course in industrial design*. London: Newnes, 207 Seiten. ISBN: 9781483104928. URL: https://www.elsevier.com/books/a-short-course-in-industrial-design/tjalve/978-0-408-00388-9 (zitiert auf Seite 127).

[Tonhäuser und Rudolph, 2017]
Claudia Tonhäuser und Stephan Rudolph (2017). „Individual Coffee Maker Design Using Graph-Based Design Languages“. In: *Design Computing and Cognition '16*. Hrsg. von John. S Gero, S. 513–533 (zitiert auf Seite 43).

[Tromp, 2016]
John Tromp (2016). „The Number of Legal Go Positions“. In: *Computers and Games*. Hrsg. von Aske Plaat, Walter Kosters und Jaap van den Herik, S. 183–190 (zitiert auf Seite 56).

[Ullman, 2003]
David G. Ullman (2003). *The Mechanical Design Process*. McGraw-Hill series in mechanical engineering. McGraw-Hill. ISBN: 9780072373387 (zitiert auf den Seiten 56, 57).

[Ulrich et al., 2020]
Karl T. Ulrich, Steven D. Eppinger und Maria C. Yang (2020). *Product design and development*. Seventh edition, International student edition. New York, NY: McGraw-Hill. ISBN: 9781260569544 (zitiert auf den Seiten 29, 45).

[Um, 2018]
Dugan Um (2018). *Solid Modeling and Applications: Rapid Prototyping, CAD and CAE Theory*. 2nd ed. 2018. Springer eBook Collection. Engineering. Cham: Springer. ISBN: 9783319745947. URL: http://dx.doi.org/10.1007/978-3-319-74594-7 (zitiert auf Seite 71).

[Vajna et al., 2010]
Sándor Vajna, Konstantin Kittel und Tibor Bercsey (2010). „Designing the solution space for the autogenetic design theory (ADT)“. In: *Proceedings of the Design Society: DESIGN Conference 2010* (zitiert auf den Seiten 50, 51).

[Valente et al., 1998]
André Valente, Joost Breuker und Walter Van De Velde (1998). „TheCommonKADS library in perspective“. In: *International Journal of Human-Computer Studies* 49.4, S. 391–416. ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1998.0212 (zitiert auf den Seiten 48, 49).

[Venn, 1880]
John Venn (1880). „I. On the diagrammatic and mechanical representation of propositions and reasonings“. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 10.59, S. 1–18. URL: https://doi.org/10.1080/14786448008626877 (zitiert auf Seite 11).

[Vogel, 2016]
Samuel Vogel (2016). „Über Ordnungsmechanismen im wissensbasierten Entwurf von

SCR-Systemen“. Dissertation. Stuttgart: Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart (zitiert auf den Seiten 33, 86).

[Vogel und Arnold, 2020]
Samuel Vogel und Peter Arnold (2020). „Towards a more complete object-orientation in graph-based design languages“. In: *SN Applied Sciences*. DOI: 10.1007/s42452-020-2959-x (zitiert auf Seite 18).

[Vogel und Rudolph, 2015]
Samuel Vogel und Stephan Rudolph (2015). „Mathematische Dimension im Entwurf komplexer Systeme“. In: *Tag des Systems Engineering 2015* (zitiert auf Seite 58).

[von Lindemann, 1882]
Carl Louis Ferdinand von Lindemann (1882). *Über die Zahl π*. DigiZeitschriften e. V. URL: http://resolver.sub.uni-goettingen.de/purl?PPN235181684_0020 (zitiert auf Seite 9).

[Voss et al., 2021]
Christopher Voss, Frank Petzold und Stephan Rudolph (2021). „Connecting Building Design with the Digital Factory by Design Languages to Explore Different Solutions“. In: *Journal of Integrated Design and Process Science*. DOI: 10.3233/JID200019 (zitiert auf Seite 43).

[Walter, 2021]
Benedikt Walter (2021). „Formal knowledge representations for textual automotive system requirements and tests“. Dissertation. Institut für Flugzeugbau, Universität Stuttgart. URL: http://nbn-resolving.de/urn:nbn:de:bsz:93-opus-ds-113790 (zitiert auf Seite 28).

[Walter et al., 2018]
Benedikt Walter, Dennis Kaiser und Stephan Rudolph (2018). „Machine-executable Model-based Systems Engineering with Graph-based Design Languages“. In: *Complex System Design and Management (CSD&M)* (zitiert auf Seite 43).

[J. Wang, 2002]
Juite Wang (2002). „Improved engineering design concept selection using fuzzy sets“. In: *International Journal of Computer Integrated Manufacturing* 15.1, S. 18–27. DOI: 10.1080/09511920110034996 (zitiert auf Seite 57).

[R. Wang et al., 2018]
Ru Wang, Anand Balu Nellippallil, Guoxin Wang, Yan Yan, Janet K. Allen und Farrokh Mistree (2018). „Systematic design space exploration using a template-based ontological method“. In: *Adv. Eng. Informatics* 36, S. 163–177 (zitiert auf den Seiten 50, 51).

[Wei et al., 2010]
Piguang Wei, Wenhua Zhu und Tianpeng Wang (2010). „Research of 3D general parts library based on Knowledge Template“. In: *2010 IEEE International Conference on Mechatronics and Automation*, S. 1659–1664. DOI: 10.1109/ICMA.2010.5588875 (zitiert auf Seite 112).

[Weinbrenner, 1993]
Volker Weinbrenner (1993). „Produktlogik als Hilfsmittel zum Automatisieren von Varianten-

und Anpassungskonstruktionen“. Dissertation. München: TU München (zitiert auf den Seiten 30, 33).

[Welch und Dixon, 1990]
Richard V. Welch und John R. Dixon (1990). „Configuration Design by Iterative Redesign: Sheet Metal Bracket Design as an Example“. In: *Journal of Engineering Design* 1.3, S. 221–237. DOI: 10.1080/09544829008901654 (zitiert auf den Seiten 52, 53).

[Wittel et al., 2019]
Herbert Wittel, Dieter Jannasch, Joachim Voßiek und Christian Spura (2019). *Roloff/Matek Maschinenelemente: Normung, Berechnung, Gestaltung : mit 731 Abbildungen, 79 vollständig durchgerechneten Beispielen und einem Tabellenbuch mit 296 Tabellen*. Hrsg. von Hermann Roloff und Wilhelm Matek. 24., überarbeitete und erweiterte Auflage. Springer eBooks. Computer Science and Engineering. Wiesbaden: Springer Vieweg. ISBN: 9783658262808. URL: https://doi.org/10.1007/978-3-658-26280-8 (zitiert auf den Seiten 34, 45, 72, 76).

[Yang et al., 2008]
Dong Yang, Ming Dong und Rui Miao (2008). „Development of a product configuration system with an ontology-based approach“. In: *Computer-Aided Design* 40.8, S. 863–878. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2008.05.004 (zitiert auf den Seiten 50, 51).

[Zafirov, 2017]
Radoslav Zafirov (2017). „Model-based systems engineering methods for integrated product design, process planning, and production systems design“. Dissertation. Kaiserslautern: Technische Universität Kaiserslautern, XVI, 206 Seiten. ISBN: 978-3-95974-068-5 (zitiert auf Seite 47).

[Zech et al., 2021]
Andreas Zech, Ralf Stetter, Markus Till und Stephan Rudolph (2021). „Automated Generation of Clamping Concepts and Assembly Cells for Car Body Parts for the Digitalization of Automobile Production“. In: *Advances in Automotive Production Technology – Theory and Application*. Hrsg. von Philipp Weißgraeber, Frieder Heieck und Clemens Ackermann, S. 293–301 (zitiert auf Seite 43).

[Zimmermann und von Hoessle, 2013]
Markus Zimmermann und Johannes Edler von Hoessle (2013). „Computing solution spaces for robust design“. In: *International Journal for Numerical Methods in Engineering* 94, S. 290–307 (zitiert auf Seite 51).

[Zindani et al., 2020]
Divya Zindani, Saikat Ranjan Maity und Sumit Bhowmik (2020). „Decision making tools for optimal material selection: A review“. In: *Journal of Central South University* 27.3, S. 629–673. ISSN: 2095-2899. DOI: 10.1007/s11771-020-4322-1 (zitiert auf den Seiten 46, 47).

[Zwicky, 1971]
Fritz Zwicky (1971). *Entdecken - Erfinden - Forschen im morphologischen Weltbild*. Vollst.

Taschenbuchausg. Knaur-Taschenbücher; 264. Historisches Institut. München; Zürich: Droemer Knaur, 206 S. ISBN: 3426002647 (zitiert auf den Seiten 34, 49, 51, 128).

Normen und Richtlinien

[Cephas, 2006]
Cephas Consulting Corp (2006). *The Fast Guide to Model Driven Architecture – The Basics of Model Driven Architecture*. Cephas Cons. Corp. URL: https://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf (zitiert auf Seite 20).

[DIN 199-1]
Deutsche Institut für Normung e. V. (DIN), Hrsg. (2021). *Technische Produktdokumentation (TPD) – Teil 1: Begriffe im Dokumentationswesen*. DIN 199-1. Beuth Verlag (zitiert auf Seite 30).

[DIN 2342]
Deutsche Institut für Normung e. V. (DIN), Hrsg. (2011). *Begriffe der Terminologielehre*. DIN 2342. Beuth Verlag (zitiert auf Seite 37).

[DIN 699501-5]
Deutsche Institut für Normung e. V. (DIN), Hrsg. (2009). *Projektmanagement — Projektmanagementsysteme — Teil 5 Begriffe*. DIN 699501-5. Berlin: Beuth Verlag (zitiert auf Seite 31).

[DIN 8580]
Deutsche Institut für Normung e. V. (DIN), Hrsg. (2003). *Fertigungsverfahren – Begriffe, Einteilung*. DIN 8580. Beuth Verlag (zitiert auf Seite 78).

[ECSS-M-ST-10C]
ECSS European Cooperation for Space Standardization, Hrsg. (2009). *Project planning and implementation*. ECSS-M-ST-10C. Berlin (zitiert auf Seite 29).

[IEEE, 1990]
Institute of Electrical and Electronics Engineers (IEEE), Hrsg. (1990). *IEEE standard glossary of software engineering terminology: Approved September 28, 1990, IEEE Standards Board*. Revision and redesignation of IEEE Std 792-1983. New York, NY. ISBN: 1-55937-067-X (zitiert auf Seite 31).

[INCOSE, 2007]
The International Council on Systems Engineering (INCOSE), Hrsg. (2007). *INCOSE Systems Engineering Vision 2020, INCOSE-TP-2004-004-02*. The International Council on Systems Engineering (INCOSE) (zitiert auf Seite 40).

[INCOSE, 2021]
The International Council on Systems Engineering (INCOSE), Hrsg. (2021). *Systems Engineering Vision 2035 – Engineering Solutions for a Better World*. The International Council on Systems Engineering (INCOSE). URL: https://www.incose.org/about-systems-engineering/se-vision-2035 (zitiert auf den Seiten 7, 40, 128).

[ISO 24748]
International Organization for Standardization (ISO), International Electrotechnical Commission (IEC) und Institute of Electrical and Electronics Engineers (IEEE), Hrsg. (2018). *Systems and software engineering – Life cycle management – Part 1: Guidelines for life cycle management*. ISO/IEC/IEEE 24748: 2018 (zitiert auf den Seiten 27, 40).

[ISO 21838-1]
International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Hrsg. (2021a). *Information technology. Top-level ontologies (TLO). Requirements*. ISO/IEC 21838-1: 2021 (zitiert auf Seite 140).

[ISO 21838-2]
International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Hrsg. (2021b). *Information technology. Top-level ontologies (TLO). Basic Formal Ontology (BFO)*. ISO/IEC 21838-2: 2021 (zitiert auf Seite 140).

[Java, 2021]
James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith und Gavin Bierman (2021). *The Java® Language Specification, Document JSR-392 Java SE 17*. Hrsg. von Inc. Oracle America. URL: https://docs.oracle.com/javase/specs/ (zitiert auf Seite 72).

[MDA, 2003]
Object Management Group (OMG), Hrsg. (2003). *OMG MDA Guide 1.0.1, Document omg/2003-06-01* (zitiert auf Seite 20).

[MDA, 2014]
Object Management Group (OMG), Hrsg. (2014a). *MDA Guide rev. 2.0, Document ormsc/2014-06-01* (zitiert auf Seite 20).

[MOFM2T, 2008]
Object Management Group (OMG), Hrsg. (2008). *MOF Model to Text Transformation Language, v1.0, formal/2008-01-16* (zitiert auf Seite 21).

[NDIA, 2011]
National Defense Industrial Association (NDIA), Hrsg. (2011). *Final Report of the Model-based Engineering Subcommittee*. URL: https://content.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/modeling-and-simulation/reports/model-based-engineering.ashx (zitiert auf Seite 40).

[UML, 2017]
Object Management Group (OMG), Hrsg. (2017). *OMG Unified Modeling Language™ (OMG UML), Version 2.5.1*. Letzter Zugriff am 15.11.2021. URL: https://www.omg.org/spec/UML/2.5.1/PDF (zitiert auf den Seiten 21, 22).

[ODM, 2014]
Object Management Group (OMG), Hrsg. (2014b). *Ontology Definition Metamodel 1.1, Formal formal/14-09-02*. Zuletzt geprüft am 20.04.2022. URL: https://www.omg.org/spec/ODM (zitiert auf Seite 68).

[QVT, 2016]
Object Management Group (OMG), Hrsg. (2016). *Meta Object Facility (MOF) 2.0 Query/-View/Transformation Specification, Document formal/2016-06-03* (zitiert auf Seite 21).

[VDI 2206]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (2004a). *Entwicklungsmethodik für mechatronische Systeme*. VDI 2206. Berlin: Beuth Verlag (zitiert auf Seite 28).

[VDI 2221]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (2019). *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. VDI 2221. Beuth Verlag (zitiert auf den Seiten 7, 25–27, 29, 31, 45).

[VDI 2222]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (1997). *Konstruktionsmethodik – Methodisches Entwickeln von Lösungsprinzipien*. VDI 2222. Beuth Verlag (zitiert auf Seite 25).

[VDI 2223]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (2004b). *Methodisches Entwerfen technischer Produkte*. VDI 2223. Beuth Verlag (zitiert auf den Seiten 25, 27, 30).

[VDI 2235]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (1987). *Wirtschaftliche Entscheidungen beim Konstruieren – Methoden und Hilfen*. VDI 2235. Beuth Verlag (zitiert auf Seite 8).

[VDI 5610-1]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (2009). *Wissensmanagement im Ingenieurwesen – Grundlagen, Konzepte, Vorgehen*. VDI 5610-1. Berlin: Beuth Verlag (zitiert auf Seite 36).

[VDI 5610-2]
Verband Deutscher Ingenieure e.V. (VDI), Hrsg. (2017). *Wissensmanagement im Ingenieurwesen – Wissensbasierte Konstruktion (KBE)*. VDI 5610-2. Berlin: Beuth Verlag (zitiert auf den Seiten 7, 36, 37, 39, 57).

Abbildungsverzeichnis

Tabellenverzeichnis