Michael Reeßing

Softwarewerkzeuge für den phasen- und domänenübergreifenden Entwurf

Berichte aus dem
INSTITUT FÜR MASCHINEN- UND
GERÄTEKONSTRUKTION (IMGK)

Herausgegeben von
Univ.-Prof. Dr.-Ing. Ulf Kletzin (Maschinenelemente),
Univ.-Prof. Dr.-Ing. René Theska (Feinwerktechnik) und
Univ.-Prof. Dr.-Ing. Christian Weber (Konstruktionstechnik)
aus dem Institut für Maschinen- und Gerätekonstruktion (IMGK) an
der TU Ilmenau.

Band 29

Diese Reihe setzt die „Berichte aus dem Institut für
Maschinenelemente und Konstruktion“ fort.

Softwarewerkzeuge für den phasen- und domänenübergreifenden Entwurf

Michael Reeßing

Universitätsverlag Ilmenau
2018

Impressum

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

Diese Arbeit hat der Fakultät für Maschinenbau der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	28. August 2013
1. Gutachter:	Univ.-Prof. Dr.-Ing. Christian Weber (Technische Universität Ilmenau)
2. Gutachter:	Prof. Dr. sc. techn. Beat Brüderlin (Technische Universität Ilmenau)
3. Gutachter:	Prof. Dr. rer. nat. habil. Karl-Heinz Modler (Technische Universität Dresden)
Tag der Verteidigung:	16.09.2016

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
http://www.tu-ilmenau.de/universitaetsverlag

readbox unipress
in der readbox publishing GmbH
Am Hawerkamp 31
48155 Münster
http://unipress.readbox.net

ISSN 2191-8082 (Druckausgabe)
ISBN 978-3-86360-169-0 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2016000892

Geleitwort der Herausgeber

Die Konstruktion von Maschinen und Geräten sowie die zugehörigen Methoden und Werkzeuge sind seit den frühen 1950er Jahren ein profilbildender Schwerpunkt an der Technischen Universität Ilmenau und ihren Vorgängerinstitutionen. Es war daher ein nahe liegender Schritt, dass die drei konstruktiv orientierten Fachgebiete der Fakultät für Maschinenbau – Maschinenelemente, Feinwerktechnik/Precision Engineering, Konstruktionstechnik – im Mai 2008 das Institut für Maschinen- und Gerätekonstruktion (IMGK) neu gegründet haben. Das IMGK steht in der Tradition einer Kette ähnlicher Vorgängerinstitute, deren wechselnde Zusammensetzung hauptsächlich durch sich über der Zeit ändernde Universitätsstrukturen bedingt war.

Zweck des Institutes ist es, die Kompetenzen und Ressourcen der beteiligten Fachgebiete zu bündeln, um Forschung und Lehre zu verbessern und erzielte wissenschaftliche Ergebnisse gemeinsam in die Fachöffentlichkeit zu tragen.

Ein wesentliches Instrument hierzu ist die Schriftenreihe des Instituts für Maschinen- und Gerätekonstruktion. Sie führt eine erfolgreiche Schriftenreihe des im Jahr 1991 gegründeten unmittelbaren Vorgängerinstitutes IMK (Institut für Maschinenelemente und Konstruktion) fort.

In der Schriftenreihe erscheinen in erster Linie die am Institut entstandenen Dissertationen, daneben werden aber auch andere Forschungsberichte, die in den thematischen Rahmen passen und von allgemeinem Interesse sind, in die Schriftenreihe aufgenommen.

Der vorliegende Band 29 ist als Dissertation am Fachgebiet für Konstruktionstechnik unter der wissenschaftlichen Betreuung von Univ.-Prof. Dr.-Ing. Christian Weber entstanden. Die Herausgeber wünschen sich reges Interesse an der Schriftenreihe und würden sich freuen, wenn sie zum fruchtbaren Dialog in Wissenschaft und Praxis beitragen würde.

Ilmenau, im November 2017

Univ. Prof. Dr. Ing. Ulf Kletzin (Maschinenelemente)
Univ. Prof. Dr. Ing. René Theska (Feinwerktechnik)
Univ. Prof. Dr. Ing. Christian Weber (Konstruktionstechnik)

Vorwort

Die vorliegende Disseration entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an den Fachgebieten Konstruktionstechnik und Grafische Datenverarbeitung der Technischen Universität Ilmenau.

Mein besonderer Dank gilt meinem Doktorvater, Herrn Univ.-Prof. Dr.-Ing. Christian Weber, für die fachliche Betreuung und Unterstützung meiner Promotion. Herrn Prof. Dr. sc. techn. Beat Brüderlin und Herrn Prof. Dr.-Ing. habil. Dr. h. c. Dr. h. c. Günter Höhne danke ich für die Förderung bereits in der Anfangsphase meines Promotionsvorhabens, durch die meine Arbeit zu einem interdisziplinären Thema zwischen der Informatik und dem Maschinenbau wurde. Herrn Prof. Dr. rer. nat. habil. Karl-Heinz Modler danke ich für die Übernahme des Korreferats.

Ich danke Herrn Dr.-Ing. Ulf Döring und Herrn Dr.-Ing. Torsten Brix, die mir den Einstieg in die Welt der Constraint-Technologie bzw. der Konstruktionstechnik erleichterten und deren Arbeiten an und um die Software MASP einen der Ausgangspunkte meiner Arbeit bildeten.

Ich möchte mich ebenfalls bei allen Kollegen und studentischen Hilfskräften bedanken, die zum Gelingen meiner Arbeit beitrugen. Insbesondere danke ich Frau Dr.-Ing. Gunhild Chilian, Herrn Dr.-Ing. Stephan Husung, Herrn Dipl.-Inf. Ronny Krüger, Herrn Dr.-Ing. Markus Färber, Herrn Dipl.-Inf. Martin Weidner, Herrn Dipl.-Inf. Hilmar Dietrich und Herrn Dipl.-Inf. Sven Jäger.

Für ihre Unterstützung und Motivation, die sehr zum Gelingen meiner Arbeit beitrugen, danke ich meiner Familie.

Ilmenau, im August 2017

Michael Reeßing

Softwarewerkzeuge für den phasen- und domänenübergreifenden Entwurf

Die vorliegende Arbeit leistet einen Beitrag zur Verbesserung der Softwareunterstützung in frühen Phasen der Produktentwicklung. Dazu wurden insbesondere zwei Problemfelder identifiziert und bearbeitet. Dabei handelt es sich zum einen um die Realisierung eines phasenübergreifenden Entwurfskonzepts für Bewegungssysteme auf der Basis einer bidirektionalen Verknüpfung von Produktmerkmalen in Modellen unterschiedlicher Abstraktionsniveaus. Ausgehend von dieser Lösung wurde außerdem ein Verfahren entwickelt, das die Manipulation von CAD-Modellen in VR-Umgebungen erlaubt.

Zum anderen wurde ein Softwarewerkzeug konzipiert, das eine domänenübergreifende Modellierung und Simulation heterogener Systeme erlaubt. Als Plattform ist es dank der komponentenbasierten Softwarearchitektur um beliebige Domänen und zugehörige Berechnungsverfahren erweiterbar. Im Unterschied zu existierenden domänenübergreifenden Werkzeugen bietet es einen nutzerzentrierten Zugang zum Produktmodell, der domänenspezifische Darstellungen und Begriffe berücksichtigt.

Die Entwicklung beider Konzepte ging mit einer softwaretechnischen Umsetzung einher, die deren Realisierbarkeit belegt und eine Demonstration ihrer Arbeitsweise anhand verschiedener Anwendungsbeispiele ermöglicht. Mit dieser Arbeit wird ein umfangreiches Spektrum an Erfahrungen bei der Konzeption und Implementierung solcher Softwarewerkzeuge präsentiert und weitergegeben.

Software Tools for Phase-Spanning and Cross-Domain Design

This thesis proposes two major improvements of software tools for the early phases of product design. The first one concerns phase-spanning design of motion systems based on bi-directional interconnections of product features modelled on different levels of abstraction. In addition, this approach also led to a method for manipulating CAD models directly in VR environments.
Addressing the second issue, the thesis develops a concept for a software tool that allows cross-domain modeling and simulation of heterogeneous systems. Designed as an extensible platform, it provides means of incorporating arbitrary domains and related simulation methods. In contrast to previously existing software tools it allows the user to access the product model using domain-specific concepts and visual representations.
Software implementations of both concepts as well as different application examples prove the feasibility and power of the approaches. In addition, the thesis shares a broad range of experiences in devising and implementing such software tools.

Inhaltsverzeichnis

Vorwort vii

Zusammenfassung ix

Inhaltsverzeichnis xi

Abkürzungsverzeichnis xvii

1 Einführung 1
1.1 Einleitung 1
1.2 Zielstellung 2
1.2.1 Phasenübergreifende Modellierung 2
1.2.2 Domänenübergreifende Modellierung und Simulation 3
1.3 Struktur der Arbeit 5

2 Stand der Technik 7
2.1 Vorgehensmodelle für phasen- und domänenübergreifende Produktentwicklung 7
2.2 Software-Werkzeuge 10
2.2.1 Domänenspezifische Werkzeuge 10
2.2.2 Multidisziplinäre Werkzeuge 15
2.2.3 Strategien der domänenübergreifenden Modellierung . 17
2.2.4 Zugänglichkeit für den Nutzer 18
2.2.5 Zusammenfassung 21

3 Phasenübergreifende Modellierung 23
3.1 Ausgangslage und Zielsetzung 23
3.2 Vorgehensmodell für einen phasenübergreifenden Entwurf . . . 25

3.2.1 Einordnung in den konstruktiven Entwicklungsprozess 25
3.2.2 Möglichkeiten der Kopplung phasenspezifischer Modelle 25
3.2.3 Basiskonzept für den phasenübergreifenden Entwurf mittels verknüpfter Modelle unterschiedlicher Abstraktionsniveaus . . . 27
3.3 Modellierung des technischen Prinzips mit MASP . . . 30
3.3.1 Philosophie und Aufbau von MASP . . . 30
3.3.2 Constraint-basierte Modellierung . . . 32
3.3.3 Repräsentation des technischen Prinzips in MASP . . . 34
3.4 Konzept eines phasenübergreifenden Entwurfswerkzeugs . . . 35
3.4.1 Überblick . . . 35
3.4.2 Erzeugung einer Grobgestalt aus dem technischen Prinzip 37
3.4.3 Verknüpfung von Prinzip und Gestalt durch Zuordnungselemente . . . 39
3.5 Prototypische Implementierung des Werkzeugs . . . 42
3.6 CAD-Schnittstelle . . . 42
3.6.1 Kopplungsmöglichkeiten mit CAD-Systemen . . . 43
3.6.2 Architektur der CAD-Schnittstelle . . . 44
3.6.3 Methoden der Erzeugung von Volumenkörpern . . . 46
3.6.4 Implementierung der CAD-Schnittstelle . . . 51
3.6.5 Anwendungsbeispiel . . . 52
3.6.6 Einschätzung und Bemerkungen . . . 53
3.7 Kopplung von MASP, CAD und VR-Systemen . . . 57
3.7.1 Problemstellung und Grundidee der Lösung . . . 57
3.7.2 Notwendige Teillösungen . . . 59
3.7.3 Kopplung von MASP mit einer VR-Umgebung (FASP) 59
3.7.4 Übertragung tesselierter Modelle aus einem CAD-System in eine VR-Umgebung . . . 61
3.7.5 Kombination der Teillösungen . . . 62
3.7.6 Anwendungsbeispiel . . . 63
3.7.7 Erfahrungen und Ausblick . . . 64

4 Werkzeug für die integrierte und nutzerzentrierte Modellierung heterogener Systeme 67
4.1 Ausgangslage und Vorgehen . . . 68
4.2 Ansätze auf der Basis existierender Softwarewerkzeuge . . . 69
4.2.1 Konzept einer integrierten Lösung mit homogenem Modell und homogener Simulationsumgebung . . . 69
4.2.2 Erfahrungen mit dem Prototyp MASP/H . . . 70

4.2.3 Weitere Ansätze auf der Grundlage existierender Werkzeuge 73
4.3 Konzept einer Software-Plattform für Entwurf und Simulation heterogener Systeme 74
4.3.1 Vorteilhafte Merkmale domänenspezifischer Entwurfs- und Simulationswerkzeuge aus Sicht des Anwenders . . 74
4.3.2 Architektur der Plattform für Entwurf und Simulation 76
4.3.3 Anforderungen und Arbeitsschwerpunkte bei der Konzipierung eines Softwarewerkzeugs für den Entwurf heterogener Systeme 78
4.4 Modellierung heterogener Systeme 80
4.4.1 Begriff „Heterogenes System“ 80
4.4.2 Anforderungen und Problemstellungen 80
4.4.3 Domänenübergreifendes Gesamtmodell 81
4.4.4 Modellelemente und Transformationselemente 82
4.4.5 Beispiel für ein Modellelement mit Transformationselementen 84
4.5 Modellbildung in ausgewählten Domänen 86
4.5.1 Mechanische Modellelemente 86
4.5.2 Modellelemente der geometrischen Optik 92
4.5.3 Regelungstechnische Modellelemente 94
4.6 Integration von Berechnungsverfahren 97
4.6.1 Begriffe Simulation, Berechnungsmodul und Berechnungsverfahren 97
4.6.2 Auswahl von Simulationsverfahren zum Einsatz in frühen Phasen der Produktentwicklung 97
4.6.3 Das domänenübergreifende Gesamtmodell und solverspezifische Modelle 98
4.6.4 Transformation des solverunabhängigen Modells . . . 100
4.6.5 Koordination von Berechnungsmodulen 102
4.7 Visualisierung und Interaktion 104
4.7.1 Ziel und Bedeutung eines Interaktionskonzeptes 105
4.7.2 Anforderungen an das Interaktionssystem des Entwurfswerkzeugs für räumliche heterogene Systeme 105
4.7.3 Interaktionskonzept von Hesym 107
4.7.4 Interaktorkonzept im Detail 114
4.7.5 Visualisierung von Modellelementen 117
4.7.6 Visualisierungselemente 118
4.7.7 Interaktionsbeispiel 119
4.8 Softwaretechnische Aspekte 126

4.8.1 Allgemeine Anforderungen an die Softwarearchitektur von Hesym . . . 126
4.8.2 Infrastruktur und Produktmodell . . . 127
4.8.3 Integration von Berechnungsverfahren und Echtzeitfähigkeit . . . 130
4.8.4 Interaktionssystem aus softwaretechnischer Sicht . . . 131
4.8.5 Einbindung von Funktionalität zur Laufzeit (Plug-Ins) 135

5 Implementierung des Entwurfswerkzeugs für heterogene Systeme 139
5.1 Überblick . . . 139
5.2 Modulstruktur . . . 140
5.2.1 Basismodule . . . 142
5.2.2 Domänenspezifische Module . . . 144
5.3 Interaktoren . . . 147
5.4 Einschätzung . . . 148

6 Anwendungsbeispiele 149
6.1 Planplattenmikrometer . . . 149
6.2 Autokollimator . . . 153
6.2.1 Interaktiver Modelliervorgang . . . 153
6.2.2 Berechnungsvorgang . . . 156
6.3 Balanceregelung für ein inverses Pendel . . . 157
6.3.1 Aufbau und Modellierung in Hesym . . . 157
6.3.2 Modellvisualisierung über Domänengrenzen hinweg . . . 158
6.3.3 Erweiterung um einen Algorithmus zum Aufschwingen des Pendels . . . 159
6.4 Fehlerschätzung Nanopositionier- und Nanomessmaschine . . . 160
6.5 Anwendung im Rahmen der Digitalen Mechanismen- und Getriebebibliothek . . . 164
6.5.1 Beschreibung der Digitalen Mechanismen- und Getriebebibliothek DMG-Lib . . . 164
6.5.2 Problembeschreibung und Anwendungsfälle . . . 165
6.5.3 Arbeitsablauf . . . 166
6.5.4 Eingabedaten . . . 166
6.5.5 Extraktion der Struktur der abgebildeten Mechanismen 167
6.5.6 Strukturbeschreibung . . . 170
6.5.7 Visualisierungsstile . . . 171
6.5.8 Bewegungssimulation . . . 173
6.5.9 Ausgabedaten und Beispiele . . . 173

7 Zusammenfassung und Ausblick 175
7.1 Zusammenfassung 175
7.1.1 Phasenübergreifender Entwurf 175
7.1.2 Domänenübergreifender Entwurf 176
7.2 Ausblick 177
7.2.1 Integrierte Produktmodelle 177
7.2.2 Erweiterte CAD-Schnittstelle für MASP 177
7.2.3 Integration zusätzlicher Analyse- und Syntheseverfahren in MASP 177
7.2.4 Verallgemeinertes Verfahrens zur Manipulation von CAD-Modellen in VR-Umgebungen 178
7.2.5 Einbindung und verbesserte Koordination von Berechnungsverfahren und -werkzeugen 178
7.2.6 Aufbau und Nutzung von Lösungsrepositorien 179
7.2.7 Integration des phasen- und domänenübergreifenden Entwurfs 179

Literaturverzeichnis 181

Abbildungsverzeichnis 193

Abkürzungsverzeichnis

AIS-Player	Augmented Image Sequence Player, webbasierte Abspielsoftware für interaktive Animationen
API	Application Programming Interface, Programmierschnittstelle
C++	Objektorientierte Programmiersprache
CAD	Computer Aided Design
CAVE	Cave Automatic Virtual Environment, Projektionseinrichtung für Virtuelle Realität
CCD	Charge-coupled Device, lichtempfindlicher Sensor
COM	Component Object Model, Technik zur komponentenorientierten Softwareentwicklung
COSMOS3000	COnceptual Solid MOdeling System 3000, Forschungsplattform für Grafik, Interaktion und geometrisches Modellieren
CPM/PDD	Characteristics-Properties-Modeling / Property-driven Design
CSG	Constructive Solid Geometry
DLL	Dynamic Link Library, dynamische Programmbibliothek
DMG-Lib	Digitale Mechanismen- und Getriebebibliothek
FASP	Flexible Audiovisuelle Stereoprojektion
FEM	Finite-Elemente-Methode
Ficucs	Fast Iterative ConstrUctive Constraint Solver
GNU	GNU's Not Unix, freies unix-ähnliches Betriebssystem
GUI	Graphical User Interface, grafische Benutzeroberfläche
Hesym	Heterogeneous Systems Modeler, Name des im Rahmen dieser Arbeit entwickelten Entwurfswerkzeugs für heterogene Systeme
ID	Identifikator, eindeutiges Merkmal
KBE	Knowledge Based Engineering
KEP	Konstruktiver Entwicklungsprozess
MASP	Modeling and Analysis of Solution Principles, Software-

	werkzeug zur Modellierung und Analyse von Bewegungssystemen
MASP/H	Entwurfswerkzeug MASP mit Erweiterungen zur Modellierung heterogener Systeme
MKS	Mehrkörpersimulation, Mehrkörpersystem
MSVC	Model-Simulator-View-Controller, Erweiterung von MVC
MVC	Model-View-Controller, Architekturmuster der Softwareentwicklung
NPMM	Nanopositionier- und Nanomessmaschine
ODE	Open Dynamics Engine, Softwarebibliothek für Mehrkörpersimulation
OpenGL	Open Graphics Library
PID-Regler	Regler mit proportionalen, integrierenden und differenzierenden Anteilen
SDK	Software Development Kit
SPICE	Simulation Program with Integrated Circuit Emphasis, Simulationssoftware für elektrische Schaltungen
SPS/PLC	Speicherprogrammierbare Steuerung / Programmable Logic Controller
STL	Standard Template Library
VD2	Virtual Design 2, VR-Software
VR	Virtuelle Realität
VRML	Virtual Reality Modeling Language, Dateiformat für 3D-Modelle
XML	eXtensible Markup Language, Auszeichnungssprache für Dateiformate

Kapitel 1

Einführung

1.1 Einleitung

Die frühen Phasen des konstruktiven Entwicklungsprozesses üben maßgeblichen Einfluss auf die ihnen folgenden Entwicklungsschritte aus. Hier getroffene Entscheidungen legen den Grundstein für viele der späteren Eigenschaften des fertigen Produktes. Auch sind zu Beginn des Entwicklungsprozesses Änderungen am Produkt zu vergleichsweise geringen Kosten möglich.
Es steht daher außer Frage, dass in den frühen Phasen eine systematische Arbeit voller Sorgfalt und Umsicht für eine erfolgreiche Produktentwicklung unerlässlich ist. Dies ist insofern eine besondere Herausforderung, da der Ingenieur zu diesem Zeitpunkt einer außerordentlich großen Vielfalt an Lösungsmöglichkeiten gegenübersteht, aus denen er eine geeignete Auswahl treffen muss. So überrascht es kaum, dass in Forschung und Industrie seit langem an einer Unterstützung dieser Phasen durch Softwarewerkzeuge gearbeitet wird.
Mittlerweile existiert eine breite Palette von Werkzeugen zur Findung und Dokumentation von Ideen und Konzepten, zur Abschätzung des Produktverhaltens durch Simulation, zur Bildung und zum Vergleich von Lösungsvarianten, zur Unterstützung der Zusammenarbeit und zur Sammlung aller im Entwicklungsprozess entstandenen Produktinformationen. Obgleich es große Fortschritte bei der Schaffung dieser Unterstützung gibt, besteht bezüglich vieler Aspekte Verbesserungspotenzial. Im Rahmen dieser Arbeit werden zwei Problemfelder solcher Softwarewerkzeuge identifiziert, Konzepte zu deren Lösung erarbeitet und anhand einer Implementierung untersucht.

1.2 Zielstellung

1.2.1 Phasenübergreifende Modellierung

Das erste der erwähnten Problemfelder betrifft die bisher verfügbaren, unzulänglichen Möglichkeiten zur Integration von Softwarewerkzeugen und Produktmodellen, die in den einzelnen Phasen des konstruktiven Entwicklungsprozesses eingesetzt werden. Existierende Werkzeuge modellieren Produkte auf bestimmten Abstraktionsniveaus. In der Regel sind sie allerdings nicht oder nur eingeschränkt in der Lage, Informationen aus ihren Modellen an Werkzeuge späterer Phasen weiterzugeben. Dies hat zur Folge, dass der Vorgang der Übertragung solcher Informationen meist von manuellen Arbeitsschritten geprägt ist und somit zusätzlicher ein Aufwand für den Anwender entsteht. Gleichzeitig birgt diese Vorgehensweise zusätzliche Fehlerrisiken. Auch die Rückführung geänderter Informationen in die Modelle vorangegangener Phasen, beispielsweise im Rahmen von Entwurfsiterationen, ist nicht zufriedenstellend gelöst. Werden diese Modelle nicht nachträglich aktualisiert, entsteht eine inkonsistente Dokumentation des Entwicklungsprozesses.
Die vorliegende Arbeit setzt sich zum Ziel, ein Konzept zur phasenübergreifenden Kopplung von Modellen unterschiedlicher Entwurfsphasen und Softwareprodukte zu erarbeiten. Dies soll auf der Grundlage von Produktmerkmalen geschehen, die in den Modellen auf den einzelnen Abstraktionsebenen identifiziert und miteinander in Beziehung gesetzt werden. Um die Modelle inhaltlich konsistent zu erhalten, ist deren bidirektionale, parametrische Kopplung notwendig. Die Realisierung dieser Kopplung erfolgt unter Nutzung von Constraint-Techniken. Da in den einzelnen Entwurfsphasen unterschiedliche Softwarewerkzeuge zum Einsatz kommen, müssen zwischen ihnen geeignete Schnittstellen geschaffen werden.
Das Konzept zur phasenübergreifenden Modellierung soll am Beispiel der Kopplung einer getriebetechnischen Spezialsoftware mit einem kommerziellen CAD-System konkretisiert und erprobt werden. Um eine bidirektionale Zuordnung zwischen den Features des CAD-Datensatzes und den Merkmalen des Modells auf der Prinzipebene zu ermöglichen, ist eine geeignete Strukturierung der CAD-Modelle im Rahmen eines rechnerunterstützten Gestaltfindungsprozesses vorzunehmen. Außerdem soll eine adapterbasierte CAD-Schnittstelle umgesetzt werden, die mit Hilfe einer Menge von Grundfunktionalitäten die Verwendung einer Vielzahl am Markt befindlicher CAD-Systeme erlaubt.
In einem weiteren Schritt wird die phasenübergreifende Modellkopplung zu einem Verfahren zur Manipulation von CAD-Modellen in der virtuellen Realität weiterentwickelt. Auf der Grundlage der verknüpften Modellinhalte erfolgt eine aufgabenabhängige Auswahl von Parametern des CAD-Modells. Die so reduzierte Para-

meteranzahl erlaubt die Interaktion mit dem CAD-Modell in VR-Umgebungen mit den dort verfügbaren Ein- und Ausgabegeräten.

1.2.2 Domänenübergreifende Modellierung und Simulation

Das zweite in dieser Arbeit behandelte Problemfeld ist im Bereich der domänenübergreifenden Modellierung und Simulation angesiedelt. Existierende Softwarewerkzeuge lassen sich diesbezüglich in zwei Gruppen aufteilen. Auf der einen Seite finden sich Entwurfswerkzeuge, die auf die Behandlung einer bestimmten Anwendungsdomäne spezialisiert sind. Aspekte anderer Domänen lassen sich mit ihnen nicht oder nur umständlich, beispielsweise über Analogiebetrachtungen, abbilden. Einigen dieser Werkzeuge gelingt es durchaus, ihre Modelle mittels domänenspezifischer Darstellungsweisen und Begriffe anwenderfreundlich zu visualisieren. Auf der anderen Seite existieren domänenübergreifend arbeitende Werkzeuge, die allerdings mit ihrer abstrakten Modelldarstellung, z. B. durch Gleichungen oder Blockdiagramme, die Begriffswelt des Ingenieurs verlassen.
Das zweite Ziel dieser Arbeit besteht daher in der Schaffung eines Softwarewerkzeugs, welches nicht nur die domänenübergreifende Modellierung und Simulation heterogener Systeme erlaubt, sondern gleichzeitig dem Anwender einen gewohnten, domänenspezifischen Zugang zu den einzelnen Teildisziplinen des modellierten Gesamtsystems bietet. Auf diese Weise soll ein Hilfsmittel entstehen, das dem Ingenieur ein schnelles, nahezu intuitives Dokumentieren einer konkreten Lösungsidee auf der Konzeptphase ermöglicht. Vorangegangene Erfahrungen verdeutlichen, dass ein solches Vorhaben nicht durch einzelne Maßnahmen zur Erweiterung eines bestehenden Werkzeugs realisiert werden kann. Vielmehr ist eine ganze Reihe von Techniken und Teilkonzepten erforderlich, die neu entwickelt oder verbessert und im Rahmen eines Gesamtkonzepts aufeinander abgestimmt werden müssen.

Domänenübergreifendes Produktmodell

Von zentraler Bedeutung für ein solches Entwurfswerkzeug ist die Schaffung eines domänenübergreifenden Gesamtmodells, das technische Systeme unabhängig von den Datenstrukturen der Berechnungsverfahren abbildet. Hierzu wird ein Graph aus Modellelementen ausgearbeitet, die ihre Semantik und funktionsrelevante Parameter umfassen. Zur Abschätzung der Produkteigenschaften wird eine enge Kopplung der Modellierung an Simulationsverfahren angestrebt. Dabei sollen bevorzugt echtzeitfähige Verfahren zum Einsatz kommen, um eine interaktive Simulation zu ermöglichen. Das solverunabhängig formulierte, domänenübergreifende Gesamtmodell muss auf geeignete Weise in berechnungsspezifische Modelle umgewandelt werden. Im Gegenzug sollen Ergebnisse der Berechnungsverfahren

in das Gesamtmodell zurückgeführt und somit der Datenaustausch zwischen den domänenspezifischen Teilmodellen ermöglicht werden.

Zugänglichkeit für den Anwender

Zweifellos gehört auch eine durchdachte, der Aufgabe angepasste Mensch-Maschine-Kommunikation zu den wesentlichen Voraussetzungen für die effektive Arbeit mit einem Entwurfswerkzeug. Im Rahmen dieser Arbeit wird daher auf eine anschauliche, an der Darstellungs- und Begriffswelt der betrachteten Domänen orientierte Visualisierung der Modelle gesetzt. Diese dient gleichzeitig zur direkten Interaktion mit dem modellierten System. Eine solche problemangepasste Interaktionsschnittstelle erlaubt dem Anwender, seine gesamte Aufmerksamkeit der zu lösenden Aufgabe zu widmen.

Integration der Teilkonzepte und softwaretechnische Realisierung

Zur Realisierung der gefundenen Konzepte muss eine geeignete Softwarearchitektur geschaffen werden. Auf deren Grundlage soll eine Softwareplattform entstehen, die sowohl um Modellbestandteile unterschiedlicher Domänen als auch um Verfahren zu deren Berechnung und Visualisierung erweitert werden kann.
Die aufgeführten Teilaspekte zur Modellbildung, Berechnung, Visualisierung, Interaktion und Softwaretechnik beeinflussen sich wechselseitig. Daher sollen sie nicht isoliert voneinander betrachtet, sondern zu einem abgestimmten Gesamtkonzept vereint werden, welches die Abhängigkeiten zwischen ihnen berücksichtigt.
Einer der gedanklichen Ausgangspunkte für diese Arbeit bestand in der existierenden Software *MASP* (Modeling and Analysis of Solution Principles [BBDH01, MAS13]). Es handelt sich dabei um ein Werkzeug für den Entwurf von Bewegungssystemen auf der Prinzipebene. Einen phasen- oder domänenübergreifenden Entwurf unterstützt *MASP* nicht. Das Werkzeug kam aber aufgrund seiner hervorragenden Zugänglichkeit für den Anwender und seines leistungsfähigen Constraint-Solvers als Basis für Weiterentwicklungen in dieser Richtung in Betracht. Das bedeutet keineswegs, dass die in den folgenden Kapiteln erarbeiteten Konzepte ausschließlich für dieses Werkzeug gültig sind. Vielmehr kann *MASP* als ein Softwarepaket betrachtet werden, das in seinem Grundkonzept und seinen Eigenschaften stellvertretend für viele andere Werkzeuge zur Unterstützung der frühen Phasen steht. Abbildung 1.1 visualisiert die beiden Dimensionen der Arbeit in Bezug auf das Entwurfswerkzeug *MASP*.
Während *MASP* im Rahmen der Betrachtungen zum phasenübergreifenden Entwurf tatsächlich die Grundlage für die Entwicklung eines Softwarewerkzeugs bildete, musste für die Realisierung eines Werkzeugs zum domänenübergreifenden

Entwurf eine vollkommen neue konzeptuelle und softwaretechnische Basis geschaffen werden. Deren Bezeichnung *Hesym* steht für Heterogeneous Systems Modeler. Hesym wird im Rahmen dieser Arbeit für die Domänen Optik, Mechanik und Regelungstechnik prototypisch umgesetzt.

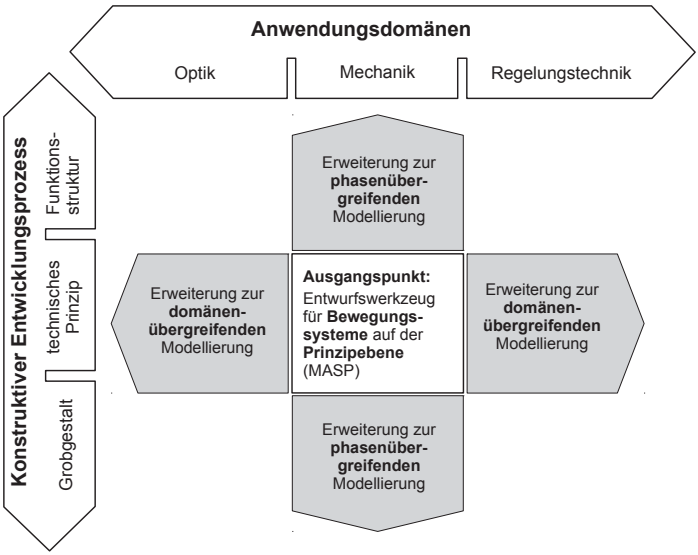


Abbildung 1.1: Grafische Darstellung der beiden Schwerpunktdimensionen der Arbeit: Ausgehend von einem Entwurfswerkzeug für Bewegungssysteme auf der Prinzipebene werden Konzepte für Werkzeuge zum phasen- und domänenübergreifenden Entwurf entwickelt

1.3 Struktur der Arbeit

Kapitel 2 verschafft einen Überblick über verfügbare Softwarewerkzeuge zur Unterstützung der frühen Phasen der Produktentwicklung. Neben einer groben Klassifikation enthält das Kapitel eine Betrachtung der unterschiedlichen Ansätze zur domänenübergreifenden Modellierung sowie der eingesetzten Interaktionskonzepte.
In Kapitel 3 wird ein Ansatz zur phasenübergreifenden Modellierung vorgestellt, der auf der Kopplung der Modelle unterschiedlicher Abstraktionsniveaus beruht. Eine softwaretechnische Realisierung erfolgt auf der Basis des Werkzeugs *MASP*.

Den Hauptteil der Arbeit stellt Kapitel 4 dar. Es beschreibt das Konzept für ein Werkzeug zur domänenübergreifenden Modellierung und Simulation. Es untergliedert sich in Abschnitte über die Modellbildung, die Integration von Simulationsverfahren, ein problemangepasstes Visualisierungs- und Interaktionskonzept sowie die notwendige Softwarearchitektur.
Das Kapitel 5 beschreibt die Implementierung des Softwarewerkzeugs *Hesym*, das die in Kapitel 4 dargestellten Konzepte prototypisch umsetzt.
Die Funktionsweise des entstandenen Werkzeugs wird in Kapitel 6 anhand von Anwendungsbeispielen im Detail erklärt.
Kapitel 7 nimmt eine abschließende Zusammenfassung vor und gibt einen Ausblick über weiterführende Forschungsarbeiten.

Kapitel 2

Stand der Technik

Dieses Kapitel charakterisiert den Stand der Technik bezüglich existierender Entwurfswerkzeuge für frühe Phasen sowie der Vorgehensmodelle zur Produktentwicklung, soweit sie für das entstandene Entwurfswerkzeug *Heterogeneous Systems Modeler* (*Hesym*) sowie für die weiterentwickelten Versionen von *MASP* relevant sind.

2.1 Vorgehensmodelle für phasen- und domänenübergreifende Produktentwicklung

Die Entwicklung von Produkten erfordert grundsätzlich eine aufgabenabhängige und problemangepasste Vorgehensweise. Dennoch umfassen viele Produktentwicklungsprozesse ähnliche Arbeitsschritte und Teilaufgaben, deren Abfolge in Form von Vorgehensmodellen abstrahiert werden kann. Diese lassen sich jeweils als Methodiken für ganze Klassen von Problemen heranziehen. Ein wichtiger prozessorientierter Vertreter solcher Vorgehensmodelle ist der konstruktive Entwicklungsprozess des Maschinenbaus (KEP), der in unterschiedlichen Varianten vorliegt (z. B. [PB03, Rot00]) und in der VDI-Richtlinie 2221 genormt beschrieben ist [VDI93]. Abbildung 2.1 zeigt die Grundstruktur des KEP. Der Entwicklungsprozess durchläuft Phasen, in denen Produktmerkmale konkretisiert und mittels festgelegter Modellrepräsentationen dokumentiert werden.
Andere Methodiken verfolgen modellzentrierte Ansätze. Bei diesen werden die Entwicklungsphasen des Prozesses durch anwendungsabhängige Iterationen von Synthese- und Analyseschritten ersetzt, während derer sich das Produkt dem Soll-

zustand annähert. Dieser Gedanke ist in Hinblick auf ein Entwurfs- und Simulationswerkzeug interessant, insbesondere im Zusammenhang mit einer geeigneten Strukturierung der Produktinformationen, wie sie beispielsweise beim *CPM/PDD*-Ansatz (*Characteristics-Properties-Modeling / Property-driven Design* [Web12], [Web05]), verfolgt wird. Dieser Ansatz unterscheidet zwischen direkt beeinflussbaren Merkmalen und daraus resultierenden Eigenschaften eines Produktes. Entwurfswerkzeuge bilden in ihren Modellen meist die Merkmale des Produktes ab und erlauben in Kombination mit Simulations- und Berechnungsverfahren Aussagen über dessen Eigenschaften.

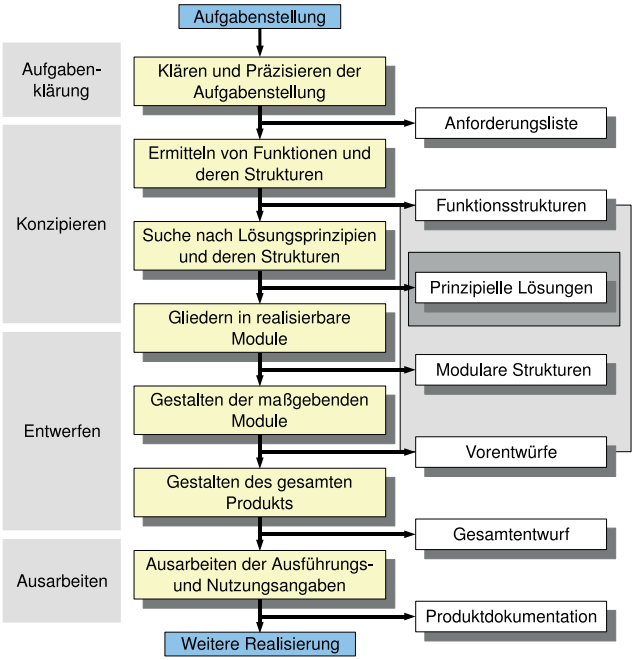


Abbildung 2.1: Konstruktiver Entwicklungsprozess nach VDI 2221

In vielen Anwendungsbereichen haben sich zusätzlich domänenspezifische Methoden entwickelt. So existieren angepasste Vorgehensmodelle beispielsweise für den Reglerentwurf [Phi04] oder die Softwareentwicklung (*Rational Unified Process* [Kru03]). Die übergeordnete Gemeinsamkeit besteht in der Empfehlung eines Prozesses, der von der Aufgabenstellung bis zu einem detailliert ausgearbeiteten und dokumentierten Produkt führt.

Obgleich generalisierte Methoden wie der KEP nach VDI 2221 die Entwicklung heterogener Produkte durchaus zulassen [Kra02], führten weitergehende Überlegungen zu domänenübergreifenden Vorgehensmodellen, die zusätzlich Aspekte des Zusammenspiels der Fachdisziplinen betonen sollen. Meist behandeln sie die Besonderheiten häufig wiederkehrender Kombinationen von Domänen, wie beispielsweise der Mikrosystemtechnik als Zusammenwirken von Mikromechanik, Mikrooptik, Mikroelektronik oder der Feinwerktechnik, die Mechanik, Optik und Elektronik zusammenführt [Wat06, VDI94].
Bestimmte Teilprobleme des domänenübergreifenden Entwurfs bleiben allerdings auch in diesen Ansätzen wenig zufriedenstellend gelöst. Beispielsweise definiert die „VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme“ [VDI04] das sogenannte V-Modell, das den Entwicklungsprozess in Makrozyklen mit zunehmender Konkretisierung unterteilt (Abbildung 2.2). Jeder Zyklus beginnt mit dem Systementwurf als domänenübergreifendes Lösungskonzept, das im anschließenden domänenspezifischen Entwurf in die beteiligen Disziplinen zerlegt und später in der Phase der Systemintegration wieder zu einem Gesamtsystem zusammengefügt wird.

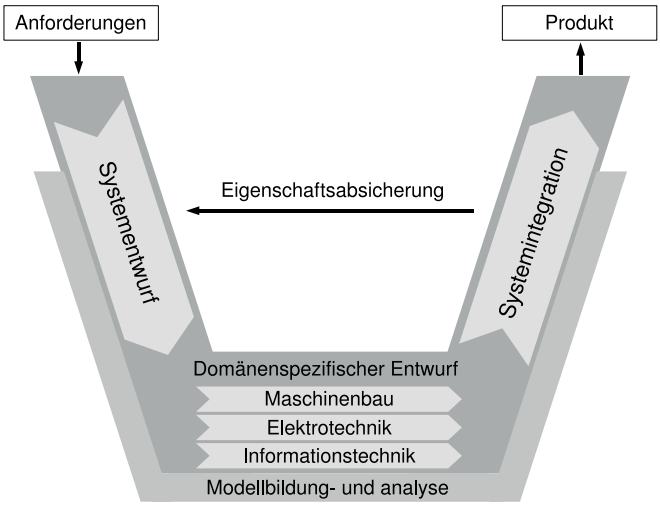


Abbildung 2.2: V-Modell der Mechatronik nach VDI 2206 mit domänenspezifischen Entwurfsphasen

Die Aufspaltung in getrennt verlaufende, domänenspezifische Entwurfsphasen ergibt vor dem Hintergrund der Bearbeitung durch unterschiedliche Fachleute

durchaus Sinn, kann aber auch als Hinweis für eine verbesserungswürdige Werkzeugunterstützung aufgefasst werden. Gelänge eine domänenübergreifende Modellbildung während des gesamten Prozesses, würde der Aufwand für die Zerlegung und anschließende Integration entfallen und an den Nahtstellen der Disziplinen auftretende Probleme früher erkannt werden. Gleichzeitig erleichterte diese Vorgehensweise eine Optimierung nicht nur der Teilsysteme, sondern des Gesamtsystems. Das im Rahmen der Arbeit entwickelte Entwurfswerkzeug für heterogene Systeme versteht sich nicht als Quelle einer eigenständigen Methodik, möchte allerdings dennoch einen Beitrag zur besseren Werkzeugunterstützung in den frühen Phasen bekannter Vorgehensmodelle leisten. Es fügt sich ein in den konstruktiven Entwicklungsprozess nach VDI 2221 und fördert die domänenübergreifende Modellierung technischer Prinzipe am Beispiel von Bewegungssystemen mit optischen und regelungstechnischen Komponenten. Die Kopplung dieser Modelle mit den Systemrepräsentationen der Gestaltebene ist ein weiterer Schwerpunkt.

2.2 Software-Werkzeuge

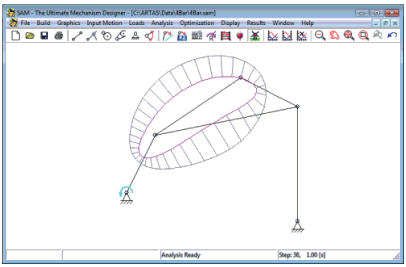
Dieser Abschnitt gibt einen Überblick über existierende Entwurfswerkzeuge. Für eine grobe Einteilung werden domänenspezifische und multidisziplinäre Werkzeuge unterschieden. Die Zuordnung von Softwarepaketen zu diesen Kategorien ist nicht immer eindeutig möglich, da beispielsweise einige domänenspezifische Werkzeuge durch die Verwendung von Ersatzmodellen eine rudimentäre Unterstützung anderer Disziplinen bieten.
Detaillierte Informationen zu den beschriebenen Produkten sind vorrangig auf den Webseiten der Hersteller hinterlegt. Auf eine Auflistung dieser Webseiten im Literaturverzeichnis wurde bewusst verzichtet, da sie problemlos aufzufinden und die Informationen tagesaktuell abrufbar sind.

2.2.1 Domänenspezifische Werkzeuge

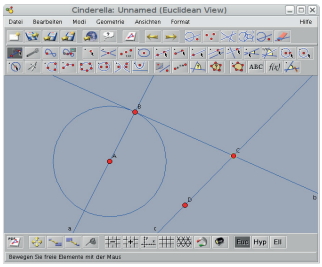
Als domänenspezifische Werkzeuge sollen die Softwareprodukte bezeichnet werden, deren Modellbildung und -präsentation sich klar an eine bestimmte Anwendungsdomäne anlehnen. Im Folgenden werden Werkzeuge für eine Auswahl typischer Anwendungsbereiche vorgestellt. Eine derartige Zusammenstellung kann naturgemäß nicht vollständig sein und soll lediglich einen Eindruck der Möglichkeiten und des Grades der Spezialisierung häufig eingesetzter Softwarewerkzeuge vermitteln. Auch die Einteilung in Domänen ist blickwinkelabhängig und wurde im Rahmen dieser Arbeit aus Sicht der Anwendungsgebiete vorgenommen.

Mechanik

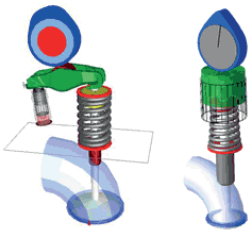
Eine Gruppe von Werkzeugen im Anwendungsbereich der Mechanik soll die Lösung kinematischer Problemstellungen unterstützen. Hierzu werden vorwiegend geometrische Eigenschaften von Bewegungssystemen erfasst und zur Analyse herangezogen. Typische Vertreter dieser Gruppe sind die Produkte *SAM*, *Kintop*, *Genesys* und *MASP* (Abbildung 2.3). Ferner lassen sich Bewegungssysteme in begrenztem Rahmen auch mit Geometriewerkzeugen wie *Cinderella* oder *The Geometer's Sketchpad* analysieren.



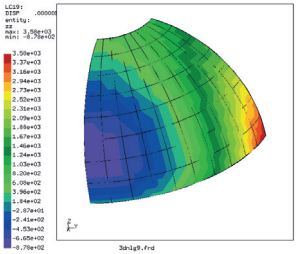
Quelle: www.artas.nl



Quelle: www.cinderella.de



Quelle: www.simpack.com



Quelle: www.calculix.de

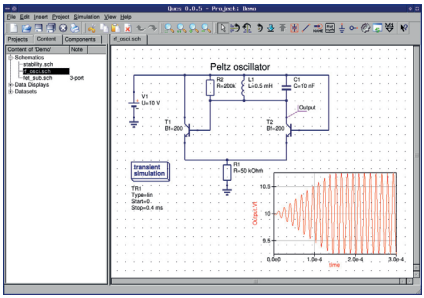
Abbildung 2.3: Beispiele für Werkzeuge zur Modellierung und Analyse mechanischer Problemstellungen (Kinematik: *SAM*, *Cinderella*, MKS: *SIMPACK*, FEM: *CalculiX*)

Eine weitere Gruppe von Werkzeugen widmet sich der Mehrkörpersimulation (MKS). Im Unterschied zu Kinematikwerkzeugen werden hier zusätzliche Parameter des modellierten Systems erfasst, die eine Analyse dynamischer Eigenschaften ermöglichen. Bekannte Beispiele für MKS-Werkzeuge sind *Working Model* und *ADAMS* der Firma MSC sowie *SIMPACK*, *DADS* oder *alaska* [Ala09].
Die Klärung weiterer Fragestellungen der Mechanik ist mit Werkzeugen möglich, die nach dem Prinzip der Finite-Elemente-Methode (FEM) arbeiten. Mit einer

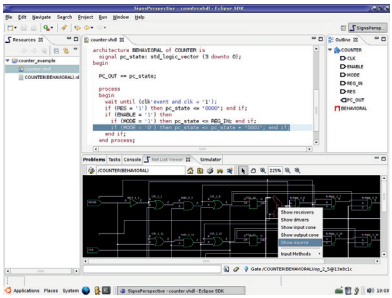
detaillierten Geometriebeschreibung (Unterteilung in Flächen- oder Volumenelemente) und der Beschreibung von Eigenschaften und Randbedingungen lassen sich z. B. die Verformung von Bauteilen oder Probleme der Strömungsmechanik untersuchen. Stellvertretend für die Vielzahl verfügbarer Programmpakete, die zum Teil in Verbindung mit CAD-Werkzeugen verwendet werden, seien hier *Ansys*, *Nastran*, *PTC Mechanica* und *Dassault Simulia* genannt.

Elektrotechnik und Elektronik

Schon seit längerer Zeit existieren Werkzeuge für die Simulation elektrotechnischer und elektronischer Effekte. Grundsätzlich wird zwischen analoger und digitaler Schaltungssimulation unterschieden. Ein großer Teil der Werkzeuge für analoge Schaltungen basiert auf dem Programm *SPICE*, das in seiner ursprünglichen Form bereits 1973 vorgestellt und seitdem in verschiedenen Varianten weiterentwickelt wurde [NP73]. Modelle werden als ein Netz aus Komponenten repräsentiert, deren Verhalten zumeist durch Differentialgleichungen beschrieben ist. Bekannte Produkte sind *PSpice* von *Orcad*, *LTSpice* von *Linear Technology* oder der *Quite Universal Circuit Simulator* (*Qucs*). Einige Beispiele sind in Abbildung 2.4 dargestellt.



Quelle: qucs.sourceforge.net



Quelle: www.iti.uni-stuttgart.de

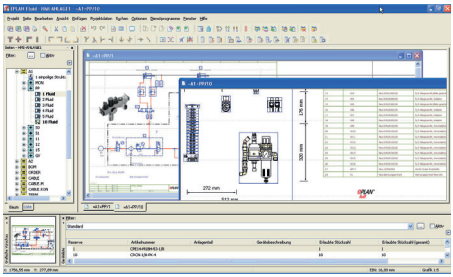
Abbildung 2.4: Werkzeuge für die Modellierung analoger und digitaler Schaltungen (*Qucs*, *Signs*)

Für die Modellierung digitaler Schaltungen kommt meist eine Hardwarebeschreibungssprache wie *VHDL* oder *Verilog* zum Einsatz. Eine solche Sprache spezifiziert Struktur und zeitliches Verhalten einer Hardwarekomponente und wird auch als ausführbare Spezifikation bezeichnet. Die Verarbeitung eines so formulierten Modells erfolgt mittels entsprechender Simulatoren. Darüber hinaus ermöglichen diese

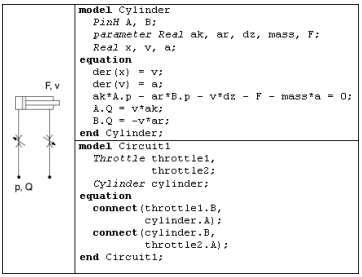
Modelle die Herstellung von Hardwarebausteinen wie *FPGAs* (*Field Programmable Gate Array*) oder *ASICs* (*Application Specific Integrated Circuit*). Ausgehend von *VHDL* und *Verilog* wurden Sprachen entwickelt, die gleichzeitig digitale und analoge Komponenten beschreiben (*VHDL-AMS*, *Verilog-AMS*). Beispiele für Simulatoren sind *ModelSim* der Firma *Mentor Graphics*, *ISE Simulator* von *Xilinx* oder *GHDL*, der auf der freien *GNU Compiler Collection* aufsetzt.

Hydraulik und Pneumatik

Obwohl eigentlich Teilgebiete der Mechanik, existieren für hydraulische und pneumatische Systeme Werkzeuge, die auf deren Behandlung spezialisiert sind. Dabei spielt die Einbindung von Herstellerkatalogen eine besondere Rolle. Die Modellierung beschränkt sich somit in den meisten Fällen auf die Verwendung und Konfiguration existierender Komponenten. Einige Werkzeuge beschreiben ihre Modelle in Simulationssprachen wie *Modelica*, was grundsätzlich die Kopplung mit anderen Domänen ermöglicht. Beliebte Softwareprodukte sind *FluidSim* der Firma *Art Systems*, *ePlan Fluid* der *ePlan GmbH* oder *Numasizing* von Numatics Inc.



Quelle: www.eplan.de



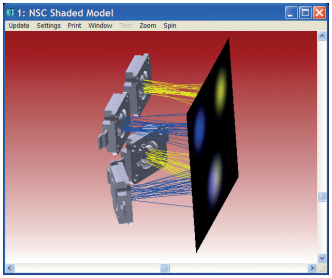
Quelle: www.art-systems.com

Abbildung 2.5: Beispiele für Werkzeuge zur Modellierung hydraulischer und pneumatischer Systeme (*ePlan*, *FluidSim*)

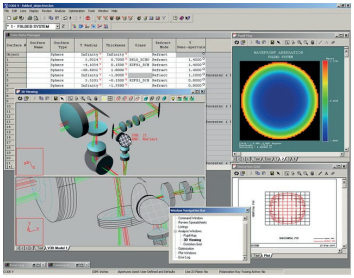
Optik

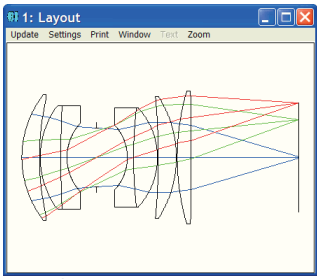
Für die Simulation einzelner optischer Komponenten oder vollständiger optischer Systeme existiert eine Reihe spezialisierter Werkzeuge. Deren Funktionsumfang beginnt bei der Analyse einfacher Strahlengänge anhand von Verfahren der geometrischen Optik und reicht bis zur Berechnung wellenoptischer Phänomene wie

Polarisation, Beugung oder Interferenz. Im Allgemeinen verwenden diese Werkzeuge Modelle, die zusätzlich zu den geometrischen auch die optischen Eigenschaften des beschriebenen Systems erfassen. Bei der Verarbeitung dieser Modelle kommt neben dem häufig eingesetzten *Ray-Tracing* eine Vielzahl individueller optischer Berechnungsverfahren zum Einsatz. Einige Werkzeuge beherrschen in begrenztem Umfang auch die Simulation einfacher mechanischer Systemkomponenten.
Beispiele für häufig eingesetzte Softwarepakete sind *ZEMAX* der *Zemax Development Corporation*, *ASAP* (Advanced Systems Analysis Program) des Herstellers *Breault Research Organization Inc.*, *CODE V* der Firma *Optical Research Associate*, *FRED* von *Photon Engineering LLC* sowie *OSLO* (*Optics Software for Layout and Optimization*) und *TRACEPRO* der *Lambda Research Corporation*.



Quelle: www.zemax.com


Quelle: www.opticalres.com



Quelle: www.zemax.com

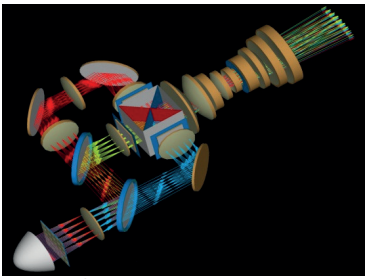
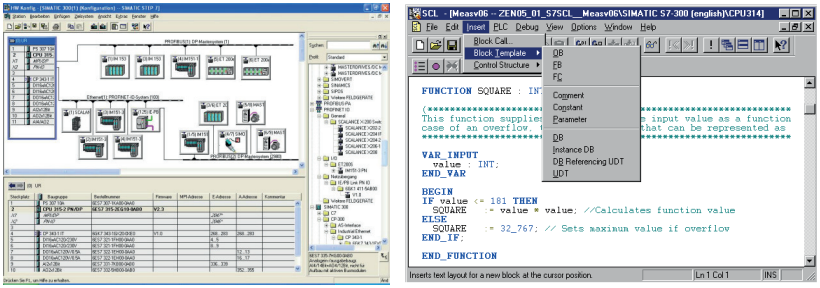

Quelle: www.breault.com

Abbildung 2.6: Beispiele für Werkzeuge zur Modellierung optischer Systeme (*ZEMAX*, *CODE V*, *ASAP*)

Automatisierungstechnik und Reglerentwurf

Der Entwurf automatisierungstechnischer Lösungen wird oft in domänenübergreifenden Werkzeugen wie *Simulink* oder *SciLab* vorgenommen, da hier das zu

beeinflussende System sehr flexibel beschrieben werden kann. Dennoch kommen auch in diesem Segment Spezialwerkzeuge für Entwurf und Programmierung von Steuerungen und Regelungen zum Einsatz, insbesondere um die Kette bis zur hardwaremäßigen Umsetzung, z. B. als Speicherprogrammierbare Steuerung (SPS) zu schließen. Die Beschreibung der Algorithmen geschieht im Quellcode einer Programmiersprache, teilweise aber auch über eine grafische Repräsentation.
Auf dem Markt befinden sich unter anderem *CoDeSys* von *3S Smart Software Solutions*, *logiCad* der *Kirchner Soft GmbH* oder *STEP 7* der *Siemens AG*.



Quelle: www.automation.siemens.de

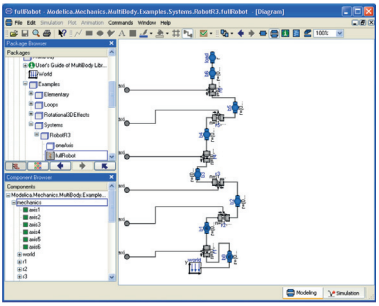
Abbildung 2.7: Werkzeug für den Reglerentwurf (*STEP 7*)

2.2.2 Multidisziplinäre Werkzeuge

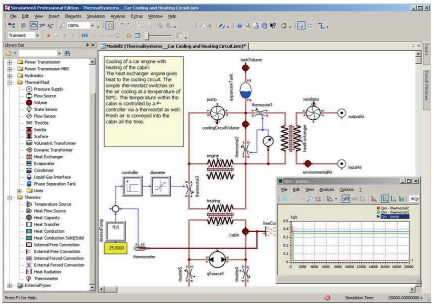
Viele technische Systeme sind in ihren Bestandteilen nicht auf eine bestimmte Domäne beschränkt. Die daraus erwachsende Notwendigkeit domänenübergreifender Modellierung wurde von Forschung und Industrie erkannt und die Entwicklung entsprechender Werkzeuge in Angriff genommen. Zu den Vorteilen solcher Ansätze gehört, dass Analyse und Optimierung des Gesamtsystems möglich werden. Die Integration domänenspezifischer Teilsysteme als eigenständiger Schritt entfällt bzw. geht mit der Modellierung implizit einher. Die zentralen Herausforderungen bei der Entwicklung eines multidisziplinären Entwurfswerkzeugs bestehen in der Bildung, in der Repräsentation und der Verarbeitung des Gesamtmodells sowie in der Schaffung eines unkomplizierten Zugangs für den Anwender. Wie diese Punkte in kommerziellen Produkten im Einzelnen gelöst sind, bleibt aus Gründen des Wettbewerbs häufig ganz oder zumindest teilweise verborgen.

Modelica-basierte Entwurfsumgebungen

Ein beträchtlicher Teil domänenübergreifender Werkzeuge setzt auf *Modelica* auf, einer Modellierungssprache für physikalische Modelle. In *Modelica* beschriebene Systeme stellen eine mathematische Repräsentation des modellierten Objektes dar, meist in Form von Gleichungen oder algorithmischen Bestandteilen. Modelle werden objektorientiert formuliert und können hierarchisch aufgebaut sein.
Modelica wird üblicherweise in einer Entwurfsumgebung zum Einsatz gebracht (Abbildung 2.8). Eine solche umfasst eine Bibliothek vordefinierter, parametrisierbarer Blöcke aus verschiedensten Domänen, die über Schnittstellen miteinander verknüpft zu einem Gesamtmodell kombiniert werden. Die so entstehende *Modelica*-Beschreibung des Systems wird anschließend von einem entsprechenden Simulator interpretiert und verarbeitet. *Modelica* erlaubt außerdem die Programmierung nutzerdefinierter Blöcke.



Quelle: www.dymola.se



Quelle: www.iti.de

Abbildung 2.8: Entwurfsumgebungen auf der Basis von *Modelica*

Beispiele für Entwurfsumgebungen, die aus einem Modellierer und einem Simulator bestehen sind *SimulationX* der *ITI GmbH*, *Dymola* von *Dynasim*, *MathModelica* der *Firma MathCore*, *MapleSim* von *MapleSoft* oder *MOSILAB*, das aus dem Projekt *GENSIM* hervorgegangen ist.

Weitere Entwurfsumgebungen

Eine Reihe weiterer Werkzeuge verfolgt einen ähnlichen Ansatz wie die *Modelica*-basierten Produkte, verwenden allerdings eigenständige Modellformate. Ein Beispiel ist *Simulink*, das auf die Infrastruktur der Mathematiksoftware *MatLab* aufsetzt. Weitere Vertreter dieser Gruppe sind *MSC SimXpert*, *SIMPLORER*, *AutomationStudio* oder *MLDesigner*.

Mittlerweile werden vermehrt unterschiedliche Verfahren der FEM und verwandter Methoden unter dem Begriff *Multiphysics* kombiniert. Die Simulation erfolgt dabei zumeist auf der Grundlage detaillierter Geometrien, die aus CAD-Systemen importiert und entsprechend der Erfordernisse der Berechnungsverfahren unterschiedlich diskretisiert werden. Das Hauptaugenmerk solcher Werkzeuge liegt vorrangig auf der genauen Analyse der Eigenschaften bereits modellierter Teilsysteme bezüglich unterschiedlicher physikalischer Effekte und weniger auf den Entwurf von Produkten in der Konzeptphase, beispielsweise auf der Basis eines Baukastensystems. Beispiele für kommerziell verfügbare Werkzeuge sind *Ansys Multiphysics*, *COMSOL Multiphysics* und *Dassault Multiphysics*.

2.2.3 Strategien der domänenübergreifenden Modellierung

Dieser Abschnitt stellt grundlegende Vorgehensweisen existierender Softwareprodukte zur domänenübergreifenden Modellierung vor.

Nutzung von Analogien und Ersatzmodellen

Die in diesem Abschnitt aufgeführten Werkzeuge verfolgen unterschiedliche Ansätze zur domänenübergreifenden Modellbildung und Simulation. Eine Möglichkeit, interdisziplinäre Betrachtungen mit einem domänenspezifischen Werkzeug anzustellen, ist die Verwendung von Ersatzmodellen. Hierzu werden Analogien zwischen den mathematischen Beschreibungen unterschiedlicher Effekte ausgenutzt. So lassen sich mit dem Elektroniksimulationswerkzeug *SPICE* einfache mechanische Komponenten wie z. B. Feder-Dämpfer-Systeme beschreiben, indem die Masse als Induktivität, die Federkonstante als inverse Kapazität und die Dämpfung als ohmscher Widerstand betrachtet werden. Geeignet ist diese Vorgehensweise jedoch meist nur für weniger komplexe Bestandteile einer fremden Domäne. Auch lassen sich nicht immer geeignete Analogien finden, so dass die Formulierung eines Ersatzmodells häufig nicht gelingt.

Paralleler Einsatz von Werkzeugen

Ein anderer Ansatz zur Behandlung interdisziplinärer Modelle besteht im parallelen Einsatz domänenspezifischer Softwareprodukte. Das Gesamtmodell wird dabei in seine Teildomänen zerlegt, die in entsprechenden Entwurfswerkzeugen modelliert werden. Um das Zusammenwirken der Teilmodelle untersuchen zu können, ist ein Austausch von Informationen zwischen den Werkzeugen notwendig. Dies kann in Form exportierter Daten erfolgen, die in einem anderen Softwareprodukt importiert werden und als Vorgaben oder Randbedingungen wirken. Die Nachteile dieser

Vorgehensweise liegen in möglichen Informationsverlusten bei der Konvertierung der Daten und in der Tatsache, dass die gegenseitige Beeinflussung der Teildomänen nur unzureichend oder überhaupt nicht simuliert werden kann.
Gelingt es hingegen, unterschiedliche Werkzeuge zur Laufzeit zu koppeln und somit einen Informationsaustausch über eine größere Zahl von Simulationsschritten herzustellen, ergeben sich deutlich umfangreichere Möglichkeiten zu Betrachtung der Interaktion zwischen den Teilmodellen (Kosimulation [Bus12]). Allerdings ist diese Art der Kopplung nur für bestimmte Kombinationen weniger Softwareprodukte implementiert, so dass sie keine allgemeine Lösung des Problems darstellt (z. B. [BS12, BKR+04, CKS03]).
Zur Koordination der Zusammenarbeit unterschiedlicher Softwareprodukte kommen gelegentlich Leit- oder Assistenzsysteme zum Einsatz. Diese geben sowohl methodische als auch technische Unterstützung, z. B. bei der Konvertierung der Daten oder der Festlegung einer Anwendungsreihenfolge von Werkzeugen [Küm99, Sch00, Düs01].

Allgemeine Simulationssysteme

Die höchste Stufe der Integration unterschiedlicher Domänen erzielen allgemeine Simulationssysteme. Sie ermöglichen die Modellierung des Gesamtsystems, meist auf Basis einer mathematischen Beschreibung. Diese wird entweder direkt in Form von Gleichungen erstellt oder durch Zusammensetzen vordefinierter Funktionsblöcke, die jeweils einen mathematischen Zusammenhang beschreiben. Der Vorteil dieser Klasse von Werkzeugen liegt in der hohen Flexibilität bezüglich der Formulierung unterschiedlichster Modelle. Ihre Anwendung erfordert allerdings oft ein hohes Maß mathematischer Kenntnisse.

2.2.4 Zugänglichkeit für den Nutzer

Ein weiteres Kriterium zur Bewertung existierender Werkzeuge ist die Qualität der Nutzerschnittstelle, da erst eine solche die produktive Benutzung der Software zulässt. Die Zugänglichkeit für den Anwender wird dabei insbesondere von der Modellrepräsentation, der Visualisierung und den zur Verfügung stehenden Interaktionswerkzeugen bestimmt. Die Frage, wie diese Aspekte gestaltet sein müssen, hängt in großem Maße von der betrachteten Domäne ab. Grundsätzlich lässt sich beobachten, dass spezialisierte Softwareprodukte eine bessere Anschaulichkeit aufweisen, da sie sich aufgrund ihres engeren Fokus eher an der Darstellungs- und Begriffswelt einer bestimmten Domäne orientieren können als allgemeinere Simulationssysteme. Beispielsweise bedient sich die Software *PSPICE* einer Visualisierung

in Form elektronischer Schaltpläne, wie sie traditionell in der Literatur Verwendung finden. Der Ingenieur muss somit keine Transformation seiner gewohnten Vorstellungswelt in eine gesonderte Modelldarstellung vornehmen. Ähnliches gilt für das Kinematikwerkzeug *ADAMS*, das den Entwurf von Bewegungssystemen mittels dreidimensionaler Modellierung unterstützt. Die zur Verfügung stehenden Komponenten entsprechen Konzepten aus der Kinematik (z. B. Getriebeglied, Gelenk).
Allgemeinen Simulationssystemen stellt sich die schwierige Aufgabe, nahezu jeden physikalischen oder logischen Zusammenhang einheitlich abzubilden, dem Anwender zu präsentieren und ihm geeignete Interaktionsmöglichkeiten mit diesem Modell zu bieten. Dazu wird häufig auf Funktionsblöcke oder Sinnbilder zurückgegriffen, die parametrisiert und über bestimmte Schnittstellen zu Schaltbildern verknüpft werden. Hinter den Funktionsblöcken verbergen sich mathematische Teilmodelle, aus denen das Gesamtmodell entsteht. Eine solche Darstellungsform ist für viele Anwendungsfälle recht gut geeignet, beispielsweise für die Modellierung regelungstechnischer Systeme, die ohnehin meist als Blockschaltbilder visualisiert werden. Die anschauliche Abbildung anderer Sachverhalte erweist sich hingegen als schwierig, insbesondere, wenn räumliche Zusammenhänge eine Rolle spielen, wie z. B. in der Optik oder der Kinematik. In diesen Fällen gelingt die Modellierung nur mit großem Aufwand. Der Anwender ist gezwungen, sein Problem in der Sprache des Werkzeugs zu formulieren, die sich vornehmlich an den Erfordernissen der verwendeten mathematischen Verfahren orientiert. Deutlich wird dies am Beispiel der Modellierung einer Schubkurbel in der Software *Simulink*. In einem Demonstrationsvideo „Modeling a Piston“ [Mat13] zum Modul *SimMechanics* wird erklärt, wie in einem ersten Modellierschritt die Modellstruktur mit Hilfe von Sinnbildern (*Machine environment, Ground, Revolute Joint, Crank* usw.) festgelegt wird (Abbildung 2.9). Anschließend findet die Eingabe der Lageparameter statt. Anhand von Eingabedialogen beschreibt der Nutzer Koordinatensysteme und andere Parameter auf der Basis mathematischer Ausdrücke (siehe Abbildung 2.10):

> *„We want the center of mass in the center of the crank shaft. Here is where we use the variable crank_length. I will divide it by two to put it in the middle. This is with respect to CS1.“*

Dieses Zitat verdeutlicht die Umständlichkeit und die Nachteile einer solchen Vorgehensweise. Die numerische Beschreibung der räumlichen Zusammenhänge muss der Vorstellungskraft des Nutzers entspringen. Erst auf dieser Basis entsteht ein dreidimensionales Modell, das entsprechend visualisiert werden kann. Aus Sicht des Anwenders sollte allerdings die direkte Eingabe des 3D-Modells im Vordergrund stehen, aus dem wiederum automatisch das mathematische Modell generiert wird.

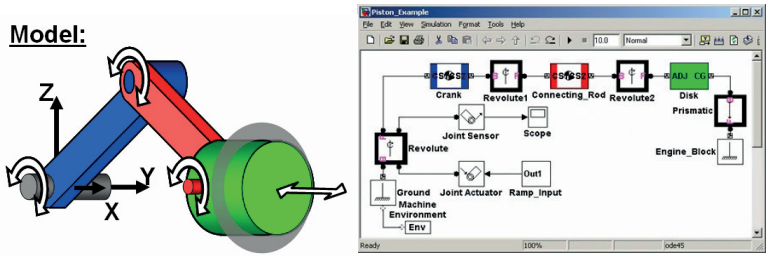


Abbildung 2.9: Bildung des gedanklichen Modells einer Schubkurbel und Umsetzung als Blockschaltbild in der Software *Simulink* (Quelle: [Mat13])

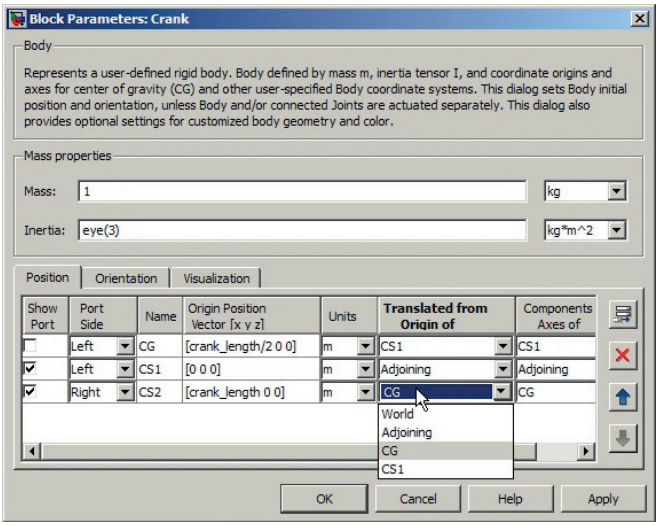


Abbildung 2.10: Umständliche und wenig anschauliche Festlegung von Lageparametern in *Simulink* mit Hilfe komplexer Dialoge und mathematischer Ausdrücke (Quelle: [Mat13])

Ein anderer interessanter Aspekt vieler allgemeiner Simulationssysteme ist die Erweiterbarkeit um benutzerdefinierte Funktionsblöcke. Diese Eigenschaft trägt sehr zur Flexibilität bei der Modellierung beliebiger Systeme bei. Allerdings setzt die Erstellung solcher Blöcke Fähigkeiten im Umgang mit einer Modelliersprache und den mathematischen Grundlagen voraus.
Darüber hinaus fällt auf, dass eine Simulation mit Rückmeldung von Ergebnissen in Echtzeit noch nicht allgemein üblich ist. Derartige Ansätze sind bisher hauptsächlich in Programmen für Lehr- und Unterhaltungszwecke umgesetzt, wo spielerisch physikalische oder technische Zusammenhänge ergründet werden können (z. B. *Phun* [Lac07], *Algodoo* [Alg13]). Eine solche interaktive Echtzeitsimulation erscheint auch für die Produktentwicklung wünschenswert. Insbesondere bei der Erarbeitung von Konzepten in frühen Phasen kann das simulationsgestützte interaktive „Ausprobieren“ bestehender Lösungen helfen, diese zu verstehen, zu verbessern und in neue Lösungen zu überführen.

2.2.5 Zusammenfassung

Bezüglich der Aufgabenstellung der domänenübergreifenden Modellierung und der Zugänglichkeit für den Nutzer lassen sich existierende Werkzeuge in zwei Gruppen einteilen. Auf der einen Seite stehen domänenspezifische ausgelegte Modellier- und Simulationsprogramme, die eine Beschreibung des Entwurfsgegenstandes in der Begriffswelt des Ingenieurs zulassen. Ihre Möglichkeiten zur domänenübergreifenden Systembeschreibung sind auf Analogiemodelle und teilweise vorhandene Werkzeugkopplung beschränkt. Auf der anderen Seite finden sich Werkzeuge, die eine echte domänenübergreifende Modellierung ermöglichen. Bezüglich ihrer Modellpräsentation und Interaktionsmöglichkeiten kommen sie dem Anwender allerdings kaum entgegen. Es existiert demnach Forschungsbedarf für Ansätze, die die positiven Eigenschaften beider Gruppen zusammenführen und zu Werkzeugen führen, die eine domänenübergreifende Modellierung und Simulation ohne Einschränkungen zulassen, während sie gleichzeitig dem Anwender das Modell mit domänenspezifischen Darstellungsweisen zugänglich machen.
Ein anderer Nachteil existierender Werkzeuge besteht darin, dass sie sich üblicherweise nur auf eine bestimmte Phase des KEP beziehen und kaum Unterstützung für eine phasenübergreifende Modellierung bieten. Kapitel 3 beschäftigt sich mit diesem Problemfeld und schlägt Lösungsmöglichkeiten zur Kopplung der Modelle unterschiedlicher Entwurfsphasen vor.

Kapitel 3

Phasenübergreifende Modellierung

3.1 Ausgangslage und Zielsetzung

Nahezu der gesamte Verlauf des konstruktiven Entwicklungsprozesses wird heutzutage von einem breiten Angebot an Softwarewerkzeugen unterstützt. Auch für die frühen Phasen liegen entsprechende Werkzeuge vor. Allerdings findet kaum eines davon einen so weit verbreiteten Einsatz wie z. B. klassische 3D-CAD-Systeme. Einer der Gründe hierfür dürfte in der starken Spezialisierung solcher Werkzeuge zu suchen sein. Typischerweise betrachten Softwarepakete in den frühen Phasen lediglich Teilaspekte eines technischen Systems, beispielsweise die Simulation eines bestimmten physikalischen Effektes, ohne dabei das Produkt in seiner Gesamtheit zu modellieren. Dennoch stellen diese Hilfsmittel eine erhebliche Unterstützung für den Ingenieur dar, da sie eine erste Bewertung einer Lösung und den Vergleich mit anderen Lösungen erlauben.

Eine andere Einschränkung dieser Softwarepakete ergibt sich aus den mangelhaften Kopplungsmöglichkeiten mit den Werkzeugen der nachgelagerten Phasen. Gewonnene Erkenntnisse gelangen bestenfalls über exportierte Informationen, oft aber auch nur in Form von Spezifikationswissen in die Modelle der später eingesetzten Werkzeuge.

Ohnehin konzentriert sich ein Großteil der Werkzeuge auf die Abbildung einzelner Entwicklungszustände eines Produktes. Die Unterstützung des Übergangs zwischen diesen Zuständen ist in den Werkzeugen nicht nur aus informationstechni-

scher, sondern auch aus methodischer Sicht unzureichend umgesetzt. Systematische Hilfestellungen bei der Überführung eines abstrakteren in einen konkreteren Entwicklungszustand sind kaum anzutreffen.
Im Vorfeld dieser Arbeit existierten bereits Ansätze zur phasenübergreifenden Unterstützung des Konstruktionsprozesses. Diese reichen von einem „Systemkonzept für die durchgängige und flexible Rechnerunterstützung des Konstruktionsprozesses“ [Fel89] bis hin zu Aspekten des Wissensmanagements, bei dem die im Rahmen der Entwicklung eines Produktes entstehenden Informationen zu einem Konstruktionsarbeitsraum zusammenfügt (*DIICAD* [Kun02]) oder Modelle verschiedener Abstraktionsniveaus unter Nutzung semantischer Netzwerke abgebildet werden [Amb97]. Andere Arbeiten befassen sich mit dem Problem der Suche innerhalb derart gesammelter Informationen (*ELEIT* [Düs01]).
Gegenstand weiterer Arbeiten ist eine durchgängig rechnerbasierte Produktentwicklung durch Kopplung von Softwaresystemen per Schnittstellendateien [Dyl02], die Koordination von Konstruktions- und Berechnungswerkzeugen in einer gemeinsamen Umgebung [EAA95, FP92], die phasenübergreifende Integration von Gestaltung und Berechnung durch die Einbindung von Simulations- und Berechnungssoftware in spezialisierte Entwurfswerkzeuge (*PHAKOM* [Joh01], *mfK* [MLS97]) und 3D-CAD-Systeme [Amf02] oder die Einbindung von Funktionsmodellen in VR-Darstellungen [Kra09].
Gleichwohl in diesem Bereich mittlerweile eine umfangreiche Palette von Werkzeugen und Hilfsmitteln existiert, steht eine gebrauchstaugliche, nutzerorientierte Softwareunterstützung, die insbesondere die frühen Phasen des KEP umfassend begleitet, verbindet und den Anschluss zu den Werkzeugen späterer Phasen herstellt, weiterhin aus. Hieraus ergab sich das Ziel der in diesem Kapitel beschriebenen Arbeiten: Die Erarbeitung und softwaretechnische Umsetzung eines Konzeptes zur verbesserten Kopplung oder Integration der Modelle unterschiedlicher Phasen des Produktentwicklungsprozesses. Die Untersuchungen sollten am Beispiel der Kinematiksoftware *MASP* (*Modeling and Analysis of Solution Principles*) geschehen, die stellvertretend für eine Reihe ähnlicher Werkzeuge betrachtet werden kann.
Die Software verfügte in der im Jahr 2002 vorliegenden Version über Möglichkeiten zur Eingabe, Visualisierung und Simulation von Mechanismen auf der Prinzipebene [BBDH02]. In *MASP* ermittelte Prinziplösungen lieferten allerdings lediglich Vorgaben, die bei der Gestaltung eines CAD-Modells berücksichtigt werden konnten. Eine automatische Übergabe von Informationen war nicht vorgesehen. Einige Gedanken zur Nutzung des „Technischen Prinzips als Grundlage für den Grobentwurf“ wurden bereits in [Bri01] geäußert. Eine detaillierte Ausarbeitung eines entsprechenden Konzeptes sowie deren softwaretechnische Umsetzung erfolgten nicht.

In den folgenden Abschnitten soll zunächst die Aufgabenstellung einer phasenübergreifenden Modellierung konkretisiert und in den konstruktiven Entwicklungsprozess eingeordnet werden. Darauf folgt eine Betrachtung des Aufbaus und der Funktionsweise von *MASP* als Basis für das hernach beschriebene Konzept eines phasenübergreifenden Entwurfswerkzeugs. Im Anschluss wird die prototypische Umsetzung in Verbindung mit einem CAD-System vorgestellt. Das Kapitel schließt mit dem Konzept und der Umsetzung eines Verfahrens, das die phasenübergreifende Kopplung zur Manipulation von CAD-Modellen in VR-Umgebungen nutzt.

3.2 Vorgehensmodell für einen phasenübergreifenden Entwurf

3.2.1 Einordnung in den konstruktiven Entwicklungsprozess

Während seines Entwurfs durchläuft ein Produkt eine Reihe von Entwicklungsphasen. Obwohl dieser Prozess in der Praxis häufig Iterationen aufweist, verläuft er grundsätzlich in Richtung einer Konkretisierung der Produkteigenschaften. Für jede der Phasen existieren charakteristische Darstellungsformen, die den Entwicklungszustand des Produktes auf einem zweckmäßigen Abstraktionsniveau dokumentieren. Rechnerintern werden solche Darstellungen in Form geeigneter Datenmodelle repräsentiert.
Die in diesem Abschnitt vorgestellten Konzepte zur phasenübergreifenden Modellierung beziehen sich auf die frühen Phasen des konstruktiven Entwicklungsprozesses. Besonderes Augenmerk liegt auf den Abstraktionsebenen Funktionsstruktur, technisches Prinzip und technischer Entwurf (Grobgestalt), die in entsprechenden Modellen abgebildet und zwischen denen ein rechnerunterstützter Übergang vorgenommen und eine parametrische Kopplung ermöglicht werden soll.

3.2.2 Möglichkeiten der Kopplung phasenspezifischer Modelle

Bisher erfolgt die Rechnerunterstützung der durch den konstruktiven Entwicklungsprozess definierten Methodik, indem für jede seiner Phasen Softwarewerkzeuge zur Erstellung geeigneter Modelle zur Verfügung stehen. Zusätzlich zur Behandlung der Modelle der einzelnen Phasen sollte eine informationstechnische Überführung zwischen den Modellen möglich sein. Im Idealfall wäre hierfür ein Produktmodell zu definieren, das alle Entwicklungszustände entlang des Prozesses integriert sowie strukturell und parametrisch in Verbindung setzt. Da ein solches

Vorhaben die kurzfristig kaum zu verwirklichende Zusammenführung aller beteiligten Softwareprodukte oder deren Funktionalitäten impliziert, bietet sich als Lösung die Kopplung der Werkzeuge und somit ihrer Teilmodelle an. Hierbei entsteht zwar zusätzlicher Aufwand bei der Synchronisierung der Modelle, im Gegenzug erfordert diese Vorgehensweise aber weniger weitreichende Modifikationen an den Werkzeugen. Beispielsweise genügt die Schaffung entsprechender Schnittstellen.
Existierende Ansätze zur Modellkopplung verfolgen häufig die Strategie, Softwareunterstützung für die frühen Phasen als Zusatzfunktionalität in das zentrale Werkzeug CAD-System zu integrieren. Diese Vorgehensweise resultiert allerdings hauptsächlich aus pragmatischen und wirtschaftlichen Überlegungen, die auf die große Verbreitung von CAD-Systemen zurückzuführen sind. Die enge Bindung erleichtert zwar den Datenaustausch, bedeutet aber gleichzeitig einen Verlust an Flexibilität, da sich ein solches Werkzeug der auf die spätere Gestaltphase ausgerichteten Herangehensweise und Infrastruktur des CAD-Systems unterordnen muss.
Eine weitere Frage besteht darin, welche Abstraktionsebenen (und damit welche Werkzeuge) zur Kopplung miteinander geeignet sind. Insbesondere ist zu klären, welche Arten von Modellen sich sinnvoll ineinander überführen lassen (Abbildung 3.1). Verschiedene Arbeiten auf dem Gebiet des *Knowledge-based Engineering* zeigten, dass eine Verbindung beispielsweise der Funktionsstruktur oder von Anforderungslisten mit Gestaltmodellen zwar möglich ist, im Sinne einer methodischen Unterstützung aber nur begrenzten Nutzen bietet. Beispiele hierfür sind die *KBE Workbench* in *CATIA V5* [Pro07] oder das iViP-Teilprojekt 3.2, in dem die „Unterstützung des Methodischen Konstruierens durch die Bereitstellung eines Werkzeugs zur Beschreibung der Produktfunktion und die Verknüpfung mit Anforderungsmodell und Bauteilstruktur“ im Vordergrund stand [iVi04].

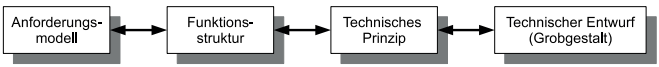


Abbildung 3.1: Mögliche Kette zweckmäßig miteinander koppelbarer Modelle der verschiedenen Abstraktionsebenen

Diese Werkzeuge beschreiben existierende Zusammenhänge zwischen den Abstraktionsebenen, die in dieser Form aber hauptsächlich Zwecke der Dokumentation und Kommunikation erfüllen können. Dies führt zu einer nur eingeschränkten Unterstützung des Produktentwicklungsprozesses, da beispielsweise die direkte Ableitung von Gestaltmerkmalen aus Anforderungen oder der Funktionsstruktur in den wenigsten Fällen systematisch möglich ist. Fügt man in diese Abfolge von Werkzeugen ein weiteres zur Modellierung des technischen Prinzips ein, so bildet es ein geeignetes Bindeglied, das sich in seinem Abstraktionsgrad zwischen den

Modellen der Anforderungsphase und Systemstrukturierung und den deutlich konkreteren Modellen des technischen Entwurfs einordnet. Eine solche Werkzeugkette erleichtert die systematische Arbeit nach der Vorgehensweise des Konstruktiven Entwicklungsprozesses gemäß VDI 2221.

3.2.3 Basiskonzept für den phasenübergreifenden Entwurf mittels verknüpfter Modelle unterschiedlicher Abstraktionsniveaus

Der Entwurfsprozess ist ein durch Iterationen gekennzeichneter Vorgang. Besonders in seinen frühen Phasen muss eine gewählte Lösung häufig verändert oder angepasst werden. In Hinblick auf die Verwendung von Softwarewerkzeugen ist das kaum problematisch, solange die Iterationen nur auf einer der Abstraktionsebenen stattfinden. Dies ist beispielsweise der Fall, wenn Gestalteigenschaften modifiziert werden, die nicht prinziprelevant sind, oder wenn Änderungen am technischen Prinzip vorgenommen werden, bevor die Weiterentwicklung zu einer Gestaltbeschreibung durchgeführt wurde. Demgegenüber gestaltet sich die Situation komplizierter, wenn das Modell eines Abstraktionsniveaus so verändert wird, dass eine oder mehrere der anderen Abstraktionsebenen beeinflusst werden. Um die Dokumentation des Entwurfsprozesses und aller Entwicklungszustände konsistent zu halten, müssen die Modelle auf den einzelnen Abstraktionsniveaus gepflegt werden. Dies geschieht entweder manuell oder mit der Unterstützung durch bidirektionale Beziehungen, die zwischen den Modellen definiert wurden.
Abbildung 3.2 illustriert die Idee der bidirektionalen Beziehungen an einem Beispiel. Die in der Funktionsstruktur geforderte Verstärkung eines Weges soll mittels eines tangential angetasteten zweiseitigen Hebels als technisches Prinzip realisiert werden. Zwischen beiden Abstraktionsniveaus lässt sich ein Zusammenhang bezüglich des Verstärkungsfaktors f und den projizierten Hebellängen d_1 und d_2 formulieren. Gleichzeitig spiegeln sich die funktionsrelevanten Maße aus dem technischen Prinzip in der Grobgestalt wider. Somit ist eine übergreifende Verknüpfung qualitativer (Modellstruktur) und gleichzeitig quantitativer Zusammenhänge (Parameter) gegeben.
Eine weitere Frage stellt sich bezüglich der Dauer, während der die Modelle im Entwicklungsprozess gekoppelt bleiben sollen. Die Abbildungen 3.3 bis 3.5 verdeutlichen mögliche Vorgehensweisen. Während in Abbildung 3.3 lediglich Parameter an den nächsten Entwicklungszustand übergeben werden, ist in Abbildung 3.4 eine zeitweilige Kopplung beider Modelle vorgesehen: Die zu Anfang aufgestellte Funktionsstruktur wird zu einer Prinziplösung weiterentwickelt. Während das technische Prinzip analysiert und optimiert wird, bleiben die Modelle gekoppelt. Somit ist in den folgenden Iterationen eine synchronisierte Arbeit auf beiden Abstraktionsebe-

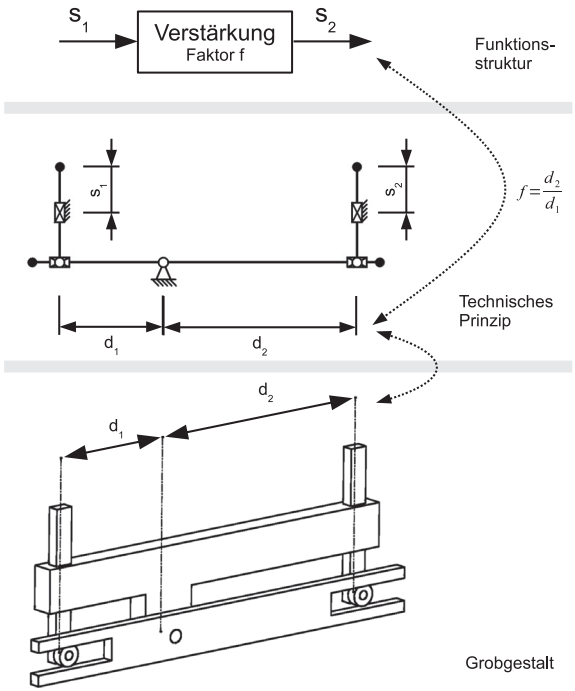


Abbildung 3.2: Zusammenhänge zwischen Modellen unterschiedlicher Abstraktionsebenen am Beispiel eines zweiseitigen Hebels

nen möglich. Anschließend erfolgen die Trennung von der Funktionsstruktur und die Festlegung der Grobgestalt ausgehend von der Prinziplösung. Auch diese beiden Entwicklungszustände bleiben durch bidirektionale Beziehungen während der Gestaltungsphase gekoppelt. In dieser Phase wird die Realisierbarkeit des technischen Prinzips aus Sicht des Gestaltentwurfs überprüft und gegebenenfalls überarbeitet. Die bidirektionale Kopplung beider Modelle erleichtert diesen Prozess, da die Auswirkungen von Änderungen sofort ersichtlich werden. Ist die Grobgestalt bezüglich ihrer prinzipreleventen Eigenschaften hinreichend gut geprüft, kann die Detaillierung anhand des Gestaltmodells fortgesetzt werden.
Obwohl diese zeitweilige Kopplung der Modelle bereits Vorteile bietet, sollte das langfristige Ziel in einer dauerhaften Kopplung der Modelle aller Phasen von der Anforderungsliste bis zum Detailentwurf bestehen (Abbildung 3.5). Hierfür sind allerdings weiterführende Betrachtungen notwendig, die den Rahmen dieser Arbeit bei weitem überschreiten.

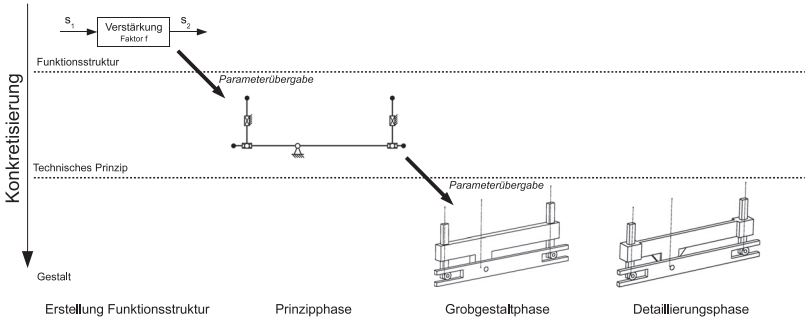


Abbildung 3.3: Sequentielle Arbeit mit Modellen unterschiedlicher Abstraktionsniveaus

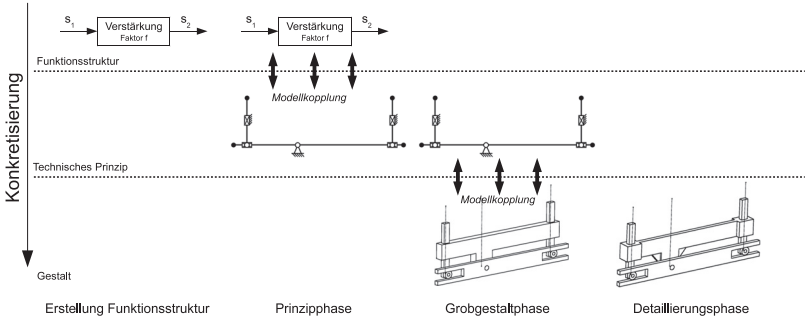


Abbildung 3.4: Teilweise parallele Arbeit mit bidirektional verknüpften Modellen

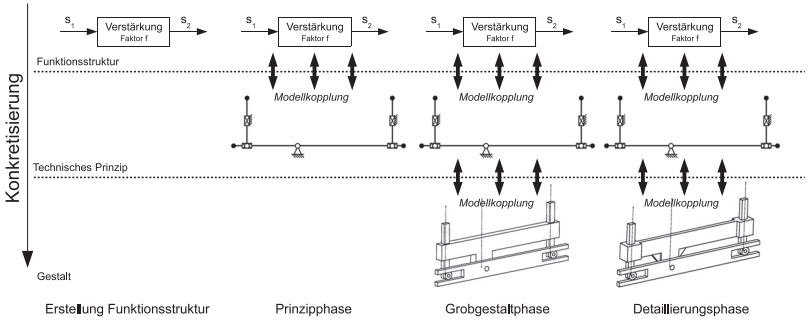


Abbildung 3.5: Vollständig parallele Arbeit mit bidirektional verknüpften Modellen

3.3 Modellierung des technischen Prinzips mit MASP

Dieser Abschnitt beschreibt die wesentlichen Eigenschaften des Aufbaus und der Modellrepräsentation von *MASP* (Modeling and Analysis of Solution Principles), soweit sie zum Verständnis des im Rahmen dieser Arbeit entwickelten Konzepts für ein phasenübergreifendes Entwurfswerkzeug beitragen. *MASP* dient dabei als Basis für die prototypische Umsetzung dieses Werkzeugs.

3.3.1 Philosophie und Aufbau von MASP

MASP ist eine Software zur grafisch-interaktiven Modellierung des technischen Prinzips von Bewegungssystemen. Sie wird an den Fachgebieten Graphische Datenverarbeitung und Konstruktionstechnik der TU Ilmenau entwickelt. *MASP* verwendet ein constraint-basiertes Modell, welches auch zur Bewegungssimulation geeignet ist. Die Kommunikation mit dem Nutzer erfolgt mittels einer Prinzipsymbolik, auf deren Basis eine automatische Erzeugung und Modifikation des Constraint-Netzes durchgeführt wird (Abbildung 3.6). Der Nutzer muss somit keine Kenntnisse über die Interna des Modells besitzen und kann sich auf die Lösung seiner eigentlichen Entwurfsaufgabe konzentrieren. Eine ausführliche Beschreibung der Modellierung mittels kinematischer Symbolik befindet sich in [Bri01]. Abbildung 3.7 zeigt die Benutzeroberfläche von *MASP* mit einem modellierten Koppelgetriebe.

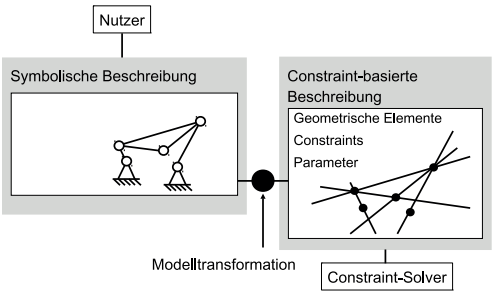


Abbildung 3.6: Transformation zwischen nutzerorientierter und constraintbasierter Modellrepräsentation in *MASP*

Der obere Teil des Fensters enthält die Werkzeugleisten, darunter liegen die Zeichenfläche sowie ein kontextabhängiger Hilfetext. In der Werkzeugleiste befinden unter anderem die verfügbaren Symbole. Sie sind in Prinzipelemente (z. B. Getriebeglied,

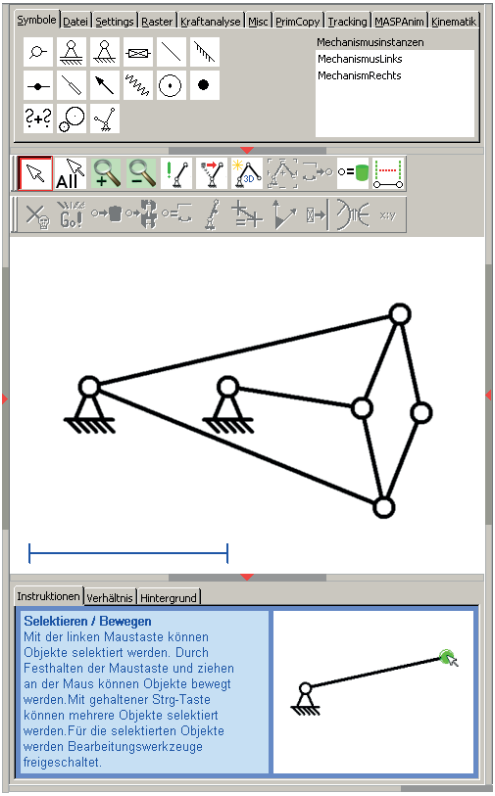


Abbildung 3.7: Benutzeroberfläche von *MASP*

Rad, Feder) und Kopplungen (Drehgelenk, Schubgelenk u.ä.) aufgeteilt. Aus den Symbolen werden nach dem Baukastenprinzip Bewegungssysteme zusammengesetzt, indem der Anwender sie ausgewählt und auf der Zeichenfläche positioniert. Dort werden sie mittels einfacher Interaktionen verknüpft, wobei Prinzipelemente über entsprechende Kopplungen miteinander in Beziehung stehen. Bereits während der Modellierung erfolgt die automatische Erzeugung des constraint-basierten Modells, so dass eine interaktive Bewegungssimulation auch mit unvollendeten Modellen jederzeit möglich ist. Weiterhin erlaubt die constraint-basierte Beschreibung eine sehr einfache Modifikation der Struktur und der Parameterwerte des Modells. Ein einfaches Modellierbeispiel ist in Abbildung 3.8 anhand einer Viergelenkkette dargestellt. Der Vorgang beginnt mit der Auswahl zweier gestellfester Drehgelenke

(links oben). Diese werden per Mauscursor grob positioniert, wobei eine spätere Eingabe genauer Koordinaten möglich ist. Auf ähnliche Weise werden dem Modell zwei freie Drehgelenke hinzugefügt (rechts oben). Anschließend werden die Getriebeglieder modelliert (links unten). Hierzu wählt der Anwender jeweils Anfangs- und Endpunkt des Elements. Liegen diese wie im Beispiel in der Nähe eines Drehgelenks, wird dieses automatisch mit dem Getriebeglied verknüpft sowie dessen Länge und Lage angepasst. Das Ergebnis ist im rechten unteren Teil des Bildes dargestellt. Der gesamte Modelliervorgang kann von einem geübten Anwender in weniger als 30 Sekunden absolviert werden.

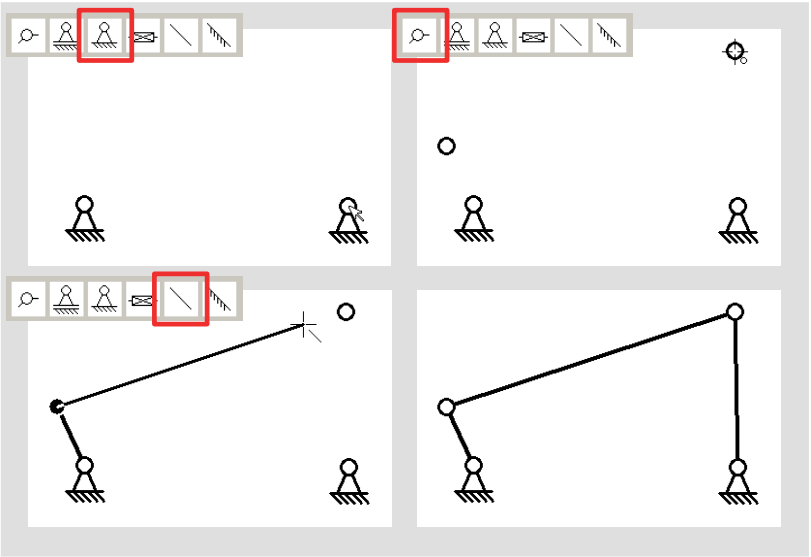


Abbildung 3.8: Modelliervorgang in *MASP* am Beispiel einer Viergelenkkette

3.3.2 Constraint-basierte Modellierung

MASP beschreibt seine Prinzipelemente und -kopplungen sowie Zusammenhänge zwischen diesen mittels geometrischer Constraints. Constraints lassen sich in diesem Kontext als Zwangsbeziehungen zwischen Objekten (geometrische Elemente, Parameter) auffassen und beschränken deren Freiheitsgrade [BR98]. Ein constraint-basiertes Modell ist aus einer Reihe solcher Objekte und Constraints zusammengesetzt. Mittels eines Constraint-Solvers kann dieses Modell in einen

konsistenten Zustand überführt werden, d.h. in einen Zustand, in dem alle Zwangsbedingungen erfüllt sind. Ein constraint-basiertes Modell kann vollbestimmt (alle Freiheitsgrade gebunden), unter- oder überbestimmt (Freiheitsgrade ungebunden bzw. mehrfach gebunden) sein.
Der in *MASP* eingesetzte geometrische Constraint-Solver basiert auf den Arbeiten von [Brü87, Brü93, Hsu96, HB97, Dör11]. Er umfasst die im Folgenden aufgeführten zweidimensionalen Grundelemente und Beziehungen. Auf diese werden die im nächsten Abschnitt dargestellten Prinzipelemente und -kopplungen zurückgeführt.

Geometrische Grundelemente in *MASP*

- Punkt (Koordinaten x, y)
- Linie (Punkt und Richtung)
- Kreis (Mittelpunkt und Radius)
- Parameter (Skalarwert)

Constraints in *MASP*

- Gleichheit zwischen zwei Punkten
- Abstand zwischen zwei Punkten
- Punkt befindet sich auf Gerade
- Punkt befindet sich auf Kreis
- Geraden sind parallel mit Abstand
- Winkel zwischen zwei Geraden
- Gerade tangential zu Kreis
- Kreis tangential zu Kreis
- Gleichungsbeziehung, z. B. $y = mx + n$

Der geometrische Solver ermöglicht außerdem die Behandlung von Überbestimmtheit mittels Priorisierung von Constraints sowie die Angabe von Gültigkeitsbedingungen (Conditional Constraints) zur Beschreibung von Schaltvorgängen [BBD99].

3.3.3 Repräsentation des technischen Prinzips in MASP

Modelle von Bewegungssystemen in *MASP* setzen sich aus Prinzipelementen und Kopplungen zusammen. Diese stehen explizit in Zusammenhang miteinander und dokumentieren somit die Struktur der Prinziplösung (Beispiel: Getriebeglied A ist über Drehgelenk B mit Getriebeglied C verbunden). Der so entstehende Graph aus Elementen und Kopplungen ist automatisch in eine constraint-basierte Beschreibung überführbar [BDR05].
Jedes Prinzipsymbol umfasst eine festgelegte Menge geometrischer Grundelemente und Constraints. Es repräsentiert einen Teilgraphen. Durch Constraint-Beziehungen zu anderen Teilgraphen entsteht das Gesamtmodell. Abbildung 3.9 zeigt den Aufbau von Prinzipelementen aus geometrischen Grundelementen am Beispiel eines Getriebegliedes. Zwei Punkte werden mittels Punkt-auf-Gerade-Constraints mit einer Geraden verknüpft. Hierbei stellt der Constraint-Solver sicher, dass die Gerade durch beide Punkte verläuft oder im Umkehrschluss sich beide Punkte auf der Geraden befinden. Zusätzlich wird ein Abstands-Constraint zwischen den Punkten definiert, um die Länge des Getriebegliedes festzulegen. Als Ergebnis ist ein Liniensegment entstanden, das dem Anwender mit der Semantik „Getriebeglied" präsentiert und in einer entsprechenden Symboldarstellung visualisiert wird. Dabei ist es nicht die Aufgabe des Anwenders, die beschriebene Definition auf Basis der geometrischen Grundelemente vorzunehmen. Er kommt allein mit dem Symbol Getriebeglied in Kontakt. Ein anderes Prinzipsymbol in *MASP* ist das Drehgelenk (Bild 3.9, rechte Seite). Es ist durch einen Punkt im Constraint-Graphen repräsentiert. Wird der Endpunkt eines Getriebegliedes mit dem Drehgelenk verbunden, generiert *MASP* eine Gleichheitsbeziehung zwischen beiden Punkten.

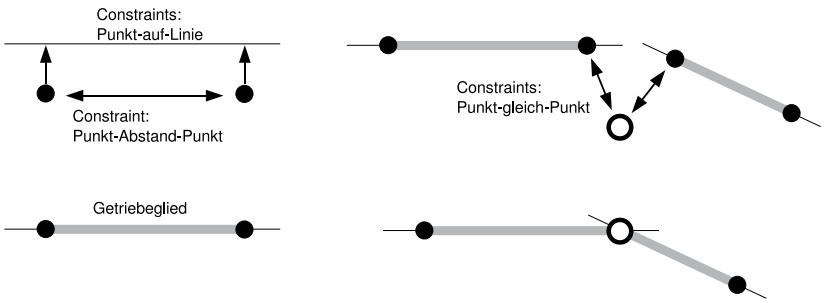


Abbildung 3.9: Aufbau von Prinzipsymbolen aus geometrischen Grundelementen und Constraints

MASP stellt folgende Prinzipsymbole bereit:

-  – Getriebeglied
-  – Gestellfestes Getriebeglied
-  – Rad (Zahnrad, Reibrad)
-  – Drehgelenk
-  – Gestellfestes Drehgelenk
-  – Loslager
-  – Feder
- – Punkt auf Element
-  – Schubgelenk
-  – Linearantrieb
-  – Abrollbeziehung

3.4 Konzept eines Softwarewerkzeugs für den phasenübergreifenden Entwurf

Dieser Abschnitt stellt das im Rahmen der Arbeit entwickelte Konzept vor, mit dem *MASP* durch Kopplung mit anderen Werkzeugen zu einem phasenübergreifenden Entwurfshilfsmittel für Bewegungssysteme weiterentwickelt wurde.

3.4.1 Überblick

Geeignet für die phasenübergreifende Kopplung sind Werkzeuge, die auf strukturorientierten, parametrischen Produktmodellen aufbauen. Diese können als Graphen aus Produktmerkmalen und deren Beziehungen betrachtet werden. Abbildung 3.10 illustriert diese Sichtweise am Beispiel der Abstraktionsebenen Funktionsstruktur, technisches Prinzip und Grobgestalt.

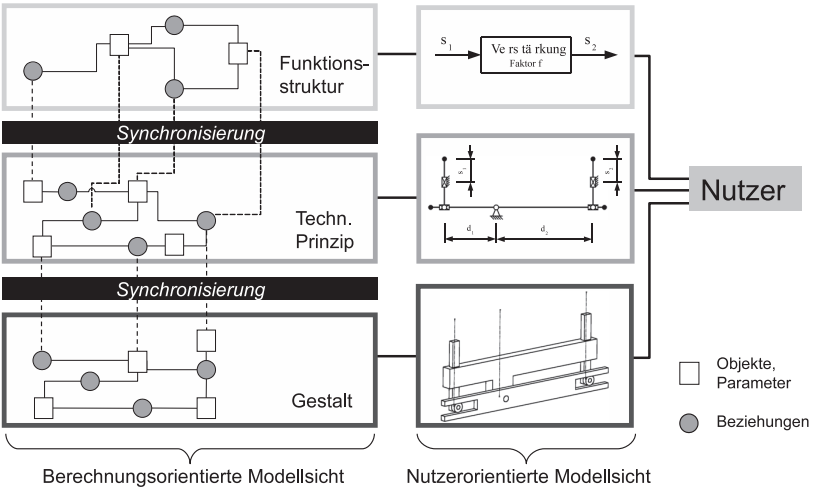


Abbildung 3.10: Konzept für ein phasenübergreifendes Entwurfswerkzeug

Das Konzept sieht vor, zwischen den Teilgraphen der Abstraktionsebenen Beziehungen zu formulieren, die zwei Aspekte widerspiegeln. Zum einen wird der Entwicklungsverlauf bestimmter Produktmerkmale entlang der Entwurfsphasen verfolgt. Zu diesem Zweck werden Abstraktions- und Konkretisierungsbeziehungen zwischen funktional äquivalenten Modellbestandteilen hinterlegt. Zusätzlich zu dieser strukturellen (qualitativen) Verknüpfung erfolgt eine parameterbezogene (quantitative) Beschreibung der Zusammenhänge in Form mathematischer Ausdrücke. Die so miteinander in Beziehung gesetzten Graphen stellen ein Hilfsmittel zur Synchronisierung der Teilmodelle dar und bilden somit die Grundlage für die konsistente Bearbeitung des Entwurfsgegenstands auf mehreren Abstraktionsniveaus (siehe auch [BDR03, BBH+02]).
Da die Formulierung entsprechender Zusammenhänge einen zusätzlichen Aufwand bedeutet, empfiehlt es sich, ihre Generierung mit dem Entwurfsvorgang der jeweils nächsten Phase zu verbinden und, soweit möglich, zu automatisieren.
Der Prototyp des phasenübergreifenden Werkzeugs sowie viele der folgenden Betrachtungen konzentrieren sich auf die Kopplung des technischen Prinzips mit der Grobgestalt. Die folgenden Abschnitte betrachten daher insbesondere zwei Problemstellungen: die Erzeugung einer Grobgestalt auf der Basis des technischen Prinzips und die Generierung und Pflege von Beziehungen zwischen beiden Abstraktionsniveaus.

3.4.2 Erzeugung einer Grobgestalt aus dem technischen Prinzip

Das technische Prinzip eines Bewegungssystems umfasst funktionsrelevante Informationen (Koordinatensysteme, Maße, Parameter und Beziehungen). Es kann als Basis für einen Gestaltforderungsplan herangezogen werden. Für die vollständige Beschreibung einer (Grob-)Gestalt sind weitere Informationen erforderlich, die nicht aus dem technischen Prinzip hervorgehen (Detailentscheidungen über die Lösungsvariante, Geometrieinformationen, zusätzliche Lageinformationen, Werkstoffparameter [HSS83]). Es existieren mehrere Möglichkeiten, Festlegungen zu diesen Informationen zu treffen. Neben einer vollständig manuellen Eingabe der Grobgestalt bieten sich eine halbautomatische und eine katalogbasierte Erzeugung von Gestaltvarianten an. Die Voraussetzungen für die unterschiedlichen Vorgehensweisen sind ähnlich. Der Grundgedanke besteht in der Erstellung von Bauteilen, die mit einem oder mehreren Prinzipelementen korrespondieren und deren Gestaltausprägung von den Prinzipkopplungen beeinflusst wird, über die sie miteinander in Beziehung stehen. Darüber hinaus ist auch der Zusammenbau der Bauteile zu Baugruppen von den Kopplungen abhängig.
Eine typische Situation für einen Übergang vom technischen Prinzip zur Grobgestalt ist in Abbildung 3.11 dargestellt: Die Volumenkörper für zwei Getriebeglieder werden aus der vom Prinzipelement definierten Länge sowie einem Rechteckprofil erzeugt. Das Prinzipsymbol des Drehgelenks, das beide Getriebeglieder miteinander verbindet, liefert ebenfalls die Gestalt eines Bauteils (Achse) sowie Gestaltforderungen an die Volumenkörper der Getriebeglieder (Bohrungen). Außerdem steuert das Drehgelenk die Erzeugung geometrischer Beziehungen (Constraints) innerhalb der Baugruppe.

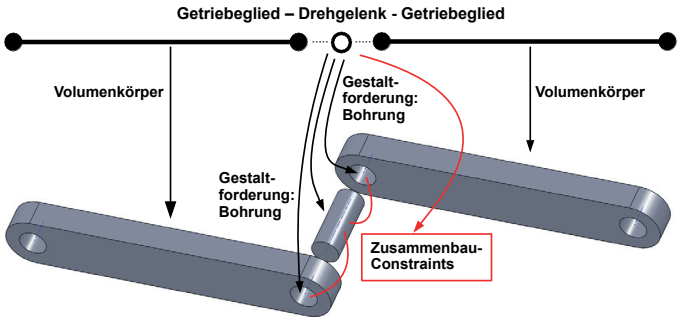


Abbildung 3.11: Gestaltforderungen von Prinzipelementen und -kopplungen am Beispiel zweier über ein Drehgelenk verbundener Getriebeglieder

Manuelle Erzeugung der Grobgestalt

Beim manuellen Verfahren zur Erzeugung der Grobgestalt werden ausgehend vom technischen Prinzip ausschließlich grundlegende geometrische Gestaltforderungen wie Koordinatensysteme und funktionsrelevante Maße im CAD-Modell vorgegeben. Die eigentlichen Volumenkörper erstellt der Konstrukteur unter Berücksichtigung der vorgegebenen Parameter. Die resultierende Arbeitserleichterung beim eigentlichen Übergang vom technischen Prinzip zur Gestalt fällt gering aus. Im Gegenzug erlaubt diese Vorgehensweise allerdings umfassende gestalterische Einflussmöglichkeiten. Da bei der Bauteilkonstruktion geeignete Beziehungen zu den Prinzipinformationen geschaffen werden, ist eine automatische Anpassung von Gestaltparametern möglich, wie sie beispielsweise bei Änderung der Parametrisierung des technischen Prinzips notwendig wird (siehe Abschnitt 3.4.3).

Halbautomatische Ableitung der Grobgestalt

Eine erste Abschätzung der räumlichen Realisierbarkeit eines Bewegungssystems (Bauraum, Kollisionsfreiheit usw.) ist oft bereits mit Hilfe simpler geometrischer Formen möglich. Dieser Gedanke führt zu einer Vorgehensweise, bei der der Konstrukteur den Prinzipelementen und -kopplungen eine Grobgestalt aus einfachen Formelementen zuweist. Deren Parametrisierung kann teilweise automatisch erfolgen (z. B. Durchmesser eines Rades, Länge eines Getriebegliedes). Alle weiteren Informationen (z. B. Querschnitte, Werkstoffe) werden mittels Standardvorgaben oder Auslegungsberechnungen bestimmt (z. B. [Spe03]) oder bei Bedarf manuell angegeben.
Ein solcher Ansatz ist naturgemäß weniger flexibel als eine manuelle Vorgehensweise, führt aber schnell und mit begrenztem Aufwand zu einem Ergebnis. Er erweist sich besonders dann als nützlich, wenn in kurzer Zeit eine große Zahl von Prinziplösungen bezüglich möglicher Gestaltvarianten verglichen werden soll.

Katalogbasierte Erstellung der Grobgestalt

Eine Weiterentwicklung der Idee der Formelemente ist die kataloggestützte Arbeitsweise. Diese verwendet eine Bibliothek, in der zu den Prinzipelementen und -kopplungen unterschiedliche Gestaltvarianten abgelegt sind. Durch Auswahl entsprechender Gestaltausführungen konkretisiert der Anwender die Prinziplösung zu einem Volumenmodell.
Als Quellen für Kataloginhalte bieten sich sowohl traditionelle Konstruktionskataloge ([Rot00], [KK98], [Kas92]), als auch bereits digital vorliegende Datenbestände wie Normkataloge aus CAD-Systemen oder elektronische Herstellerkataloge an

([Tra12], [Par12]). Zur Verwendung mit dem Entwurfswerkzeug erfordern beide Varianten eine inhaltliche Aufbereitung. Diese bedeutet zwar einen erheblichen Mehraufwand, der durch spätere Zeiteinsparungen aber durchaus zu rechtfertigen ist. Die Flexibilität der Gestalterzeugung ist von Umfang und Qualität des Kataloges abhängig.

Einschätzung und Auswahl der Verfahren

Die vorgestellten Ansätze weisen unterschiedliche Stärken und Schwächen auf. Ihre Eignung muss im Kontext der jeweiligen Problemstellung bewertet werden. Die halbautomatischen und katalogbasierten Verfahren bieten Vorteile, wenn schnell eine Ausgangsposition für die Ermittlung der Gestalt erreicht werden soll. Im fortschreitenden Prozess der Detaillierung treten sie aber zugunsten nutzererzeugter Gestaltmerkmale in den Hintergrund. Die beschriebenen Vorgehensweisen sind kombinierbar, da sie lediglich unterschiedliche Wege von einem Lösungsprinzip zu einem Volumenmodell beschreiben. Der Prototyp einer phasenübergreifenden Version von *MASP* verwendet für die Behandlung der Prinzipelemente verstärkt den halbautomatischen Ansatz, während er bei der Gestaltermittlung der Kopplungen eine katalogbasierte Arbeit bevorzugt. Die später entwickelte CAD-gekoppelte Version von *MASP* bezieht auch die manuelle Vorgehensweise mit ein.

3.4.3 Verknüpfung von Prinzip und Gestalt durch Zuordnungselemente

Neben der Erzeugung einer Grobgestalt aus dem technischen Prinzip ist für ein phasenübergreifendes Entwurfswerkzeug die Abbildung des Zusammenhangs zwischen den Modellen beider Abstraktionsebenen von Bedeutung. Hierzu sollen neben strukturellen Merkmalen (Zuordnung von Prinzipelementen zu Gestaltelementen) auch quantitative Informationen (Zusammenhänge zwischen Parameterwerten) berücksichtigt werden.
Die Basis für die Zuordnung bilden invariante Eigenschaften, die sich sowohl im technischen Prinzip als auch im Volumenmodell der Grobgestalt wiederfinden. Charakteristische Beispiele solcher Eigenschaften sind

- Koordinatensysteme,
- Punkte, Vektoren, Ebenen sowie
- Abstände, Winkel.

Die Identifikation und Verknüpfung dieser Merkmale erfolgt in Abhängigkeit vom angewandten Gestalterzeugungsverfahren manuell oder automatisch. Sobald die

Abbildung der notwendigen Zusammenhänge besteht, ist eine automatische Synchronisierung von Parameterwerten möglich.
Die technische Realisierung der parametrischen Verknüpfungen setzt auf den Modellgraphen der Abstraktionsniveaus auf. Da hier hauptsächlich geometrische Informationen hinterlegt sind, lassen sich Beziehungen zwischen ihren Bestandteilen beschreiben. Obwohl beliebig komplexe Zusammenhänge formulierbar sind, handelt es sich hierbei meist um einfache mathematische Beziehungen, wie beispielsweise die Identität zweier Vektoren oder Verhältnisgleichungen zwischen Parametern.
Die modellübergreifende Synchronisierung geschieht zu definierten Zeitpunkten, beispielsweise bei Änderungen an einem der Modelle. Ausgehend von der Quelle der Änderungen (Ausgangmodell) erfolgt die Berechnung abhängiger Parameterwerte im jeweils anderen Modell (Zielmodell). Dabei kann es zur Verletzung von Constraints oder ähnlichen Bedingungen im Zielmodell kommen, so dass dieses wiederum in einen konsistenten Zustand überführt werden muss. Im Anschluss kann eine erneute Synchronisierung mit dem Ausgangsmodell notwendig sein. Ein solcher Prozess könnte mehrere Iterationen zur Folge haben und unter Umständen nicht terminieren (z. B. falls widersprüchliche Forderungen modelliert wurden).
Zusätzlich zur parameterbezogenen, quantitativen Kopplung der Modelle sieht das Konzept eine weitere, strukturorientierte Sicht vor. Diese ordnet Gruppen von Prinzipelementen Gruppen von Volumenkörpern oder Features zu (siehe Abbildung 3.12). Die Gruppierung ist erforderlich, da in vielen Fällen einem Prinzipelement

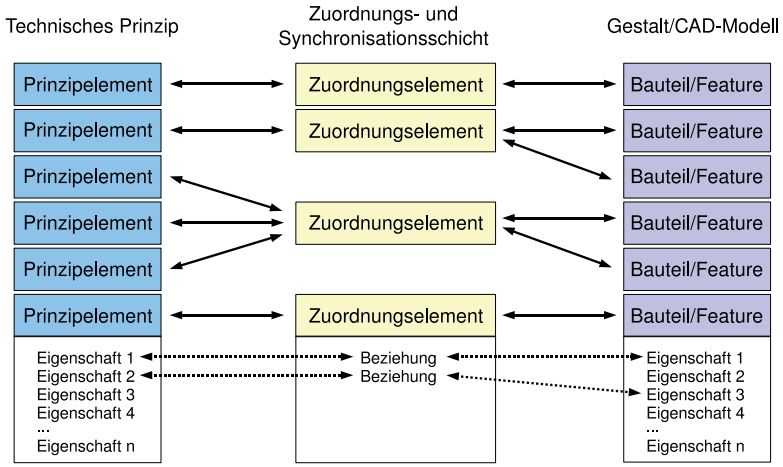


Abbildung 3.12: Zuordnungsmodell zwischen Prinzip- und Gestaltmerkmalen

nicht genau ein Gestaltelement zugeordnet werden kann. Beispielsweise können zur Repräsentation eines Prinzipelementes mehrere Volumenkörper oder Features notwendig sein.
Die phasenübergreifenden Verknüpfungen über die Zuordnungselemente stellen Abstraktions- und Konkretisierungsbeziehungen zwischen den Entwicklungszuständen des Konstruktionsobjektes dar. Bei der Arbeit mit dem Entwurfswerkzeug dienen sie zusätzlich als strukturierendes Hilfsmittel für die Verwaltung äquivalenter Modellbestandteile. Soll z. B. ein Prinzipelement aus dem Modell entfernt werden, so erscheint es im Sinne der Konsistenz sinnvoll, zum einen die beteiligten Volumenkörper und zum anderen die entsprechenden modellübergreifenden Beziehungen automatisch zu löschen.
Zuordnungselemente (siehe Abbildung 3.13) repräsentieren Entscheidungen, die im Rahmen der Gestaltfindung getroffen wurden und können somit auch als Gestaltentscheidungselemente betrachtet werden. Sie sind spezifisch bezüglich einer Ziel-Software (z. B. ein bestimmtes CAD-System oder ein Szenengraph) und einer Gestaltvariante, die für eine Gruppe von Prinzipsymbolen gewählt wurde.

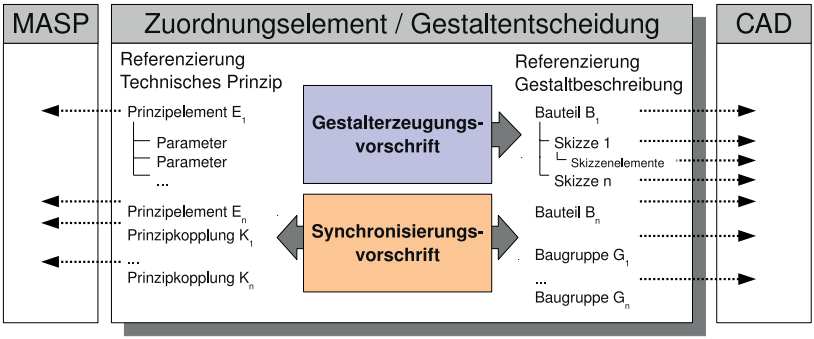


Abbildung 3.13: Aufbau eines Zuordnungs- und Gestaltentscheidungselements zur Kopplung mit einem CAD-System

Die Hauptaufgaben eines Zuordnungselementes bestehen in der anfänglichen Erzeugung der Gestalt aus Prinzipelementen und der anschließenden Synchronisierung dieser Informationen mit der Prinzipbeschreibung. Ein Zuordnungselement referenziert die Bestandteile der beteiligten Modelle, die für die Beschreibung des abgebildeten Zusammenhangs notwendig sind. So könnte man für die in Abbildung 3.11 dargestellte Situation beispielsweise drei Zuordnungselemente folgendermaßen ausarbeiten: Die Zuordner für die Getriebeglieder referenzieren einerseits die entsprechenden Prinzipsymbole mit deren Parameter „Länge“, andererseits verweisen

sie in das CAD-System auf eine Skizze einschließlich ihrer Skizzenelemente und den daraus erstellten Volumenkörper. Für das Drehgelenk entsteht ebenfalls ein solches Konstrukt, das allerdings zusätzlich zum erzeugten Volumenkörper die Skizzen der gekoppelten Getriebeglieder referenziert und diese zur Erfüllung der Gestaltforderung „Bohrung“ manipuliert.
Die persistente Speicherung der Referenzen setzt eindeutig identifizierbare Merkmale der Modellbestandteile voraus (z. B. Kennnummer, *unique ID*). Während diese in *MASP* uneingeschränkt zur Verfügung stehen, liefern die Programmierschnittstellen einiger CAD-Systeme solche Informationen nur teilweise. In diesen Fällen muss auf potenziell problematische Konzepte wie nutzerdefinierte Attribute oder Namen zurückgegriffen werden.

3.5 Prototypische Implementierung des Werkzeugs

Zum oben beschriebenen Konzept entstand eine begleitende Implementierung. Diese beschränkte sich in einer ersten Ausbaustufe auf die Arbeit mit einem Szenengraphen, der die Erzeugung einfacher räumlicher Modelle und die Erprobung des Zuordnungsmodells ermöglichte. Später erfolgte eine Integration von *MASP* in das Modellierwerkzeug *COSMOS* (COnceptual Solid MOdeling System), das am Fachgebiet Graphische Datenverarbeitung der TU Ilmenau entwickelt wurde. Mit diesem Schritt stand eine umfangreiche Palette neuer Visualisierungs- und Interaktionswerkzeuge zur Verfügung, die eine detailliertere Gestaltmodellierung erlaubten. Insbesondere die Möglichkeiten zur Parametrisierung und räumlichen Anordnung der Volumenmodelle konnten nun ausgebaut werden (Abbildung 3.14).
Aufbauend auf den bis dahin gesammelten Erfahrungen erfolgte die Weiterentwicklung des Werkzeugs in Hinblick auf die Kopplung mit einem CAD-System. Beschränkte sich die Volumenerzeugung in den vorangegangenen Ansätzen auf die *Constructive Solid Geometry (CSG)*, konnte nun auf deutlich komplexere Beschreibungsmittel zurückgegriffen werden. Die notwendigen Überlegungen zum Umgang mit diesen sind in Abschnitt 3.6.3 (Verfahren zur Gestalterzeugung) dargelegt.

3.6 CAD-Schnittstelle

Dieser Abschnitt behandelt die Kopplung von *MASP* mit parametrischen CAD-Systemen. Im Vergleich zu den vorangegangenen Lösungen ist die Beschreibung

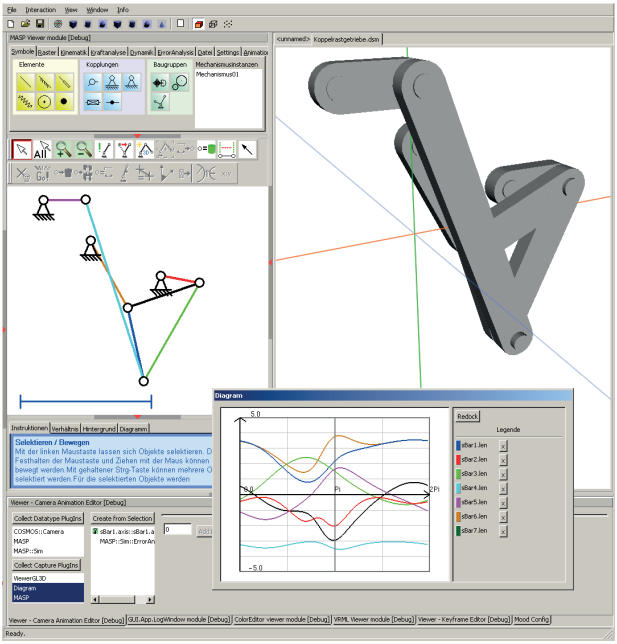


Abbildung 3.14: Integration von *MASP* in *COSMOS3000*

ausführlicher, da CAD-Systeme die bei weitem wichtigsten Werkzeuge bei der Bearbeitung mechanischer Konstruktionsobjekte sind.

3.6.1 Kopplungsmöglichkeiten mit CAD-Systemen

Zur Realisierung der Kopplung mit einem CAD-System existieren zwei Möglichkeiten:

- der Datenaustausch durch Import und Export von Dateien und
- der Zugriff auf das CAD-Dokument per Programmierschnittstelle des CAD-Systems.

Ein Datenaustausch per Datei besitzt den Vorteil, dass die Übergabe der Informationen asynchron erfolgen kann, das CAD-System für den Export also nicht vorliegen muss. Allerdings existiert eine große Zahl unterschiedlicher, teilweise sehr komplexer CAD-Dateiformate, deren Unterstützung mit erheblichem Aufwand

verbunden ist. Ein weiterer Nachteil sind die fehlenden Interaktionsmöglichkeiten zwischen dem CAD-System und dem Entwurfswerkzeug. Eine Kopplung im eigentlichen Sinne ist nicht möglich (z. B. die direkte Synchronisierung von Parametern). Da besonders der letzte Punkt eine problematische Einschränkung für die Verknüpfung von technischem Prinzip und Gestaltmodell und damit für die ebenenübergreifende Modellierung darstellt, wurde der Datenaustausch über Programmierschnittstellen (APIs, Application Programming Interface) gelöst. Somit ist eine direkte, nahtlose Kopplung mit einem CAD-System zur Durchführung bidirektionaler Interaktionen möglich.

3.6.2 Architektur der CAD-Schnittstelle

Um eine Einschränkung auf ein bestimmtes Softwareprodukt zu vermeiden, wurde eine abstrahierte Schnittstelle geschaffen, die sich durch Transformationsmodule an die Programmierschnittstellen konkreter CAD-Systeme anpasst. Dies hat den Vorteil, dass das Entwurfswerkzeug nur eine einzige Schnittstelle unterstützen muss, während die einzelnen Transformationsmodule die Schnittstellen der CAD-Systeme bedienen (Abbildung 3.15). Mit einer solchen Strukturierung ist sichergestellt, dass auch bei API-Änderungen seitens der CAD-Systeme (z. B. in neuen Versionen) keine Interna des Entwurfssystems betroffen und die zu wartenden Programmteile klar abgegrenzt sind. Durch die Transformatoren werden die nach außen zur Verfügung gestellten Fähigkeiten der einzelnen CAD-Systeme dem Entwurfssystem gegenüber vereinheitlicht. Die Schnittstelle hält Grundoperationen bereit, die zum Funktionsumfang einer großen Zahl von CAD-Systemen gehört:

- Erstellung und Positionierung von Formen durch Primitive, Skizzen und einfache Extrusionen
- Boolesche Operationen
- Baugruppen-Operationen (Positionierung, Verknüpfung, Constraints)

Neben der Vereinheitlichung wird mit diesen Grundoperationen eine zweckangepasste Vereinfachung der umfangreichen Möglichkeiten und Operationen von CAD-Systemen erreicht. Eine Beschränkung auf die zur Übermittlung des technischen Prinzips bzw. der Grobgestalt notwendigen Funktionen ist sinnvoll, da alle weiteren Detaillierungen des Grobentwurfs ohnehin später in der gewohnten Arbeitsweise mit dem CAD-System vorgenommen werden.
Um eine persistente Kopplung der Modelle (im Entwurfssystem und im CAD-System) zu erreichen, werden seitens des Entwurfswerkzeugs die CAD-System-internen Bezeichner der exportierten Objekte gespeichert. Mittels dieser Informa-

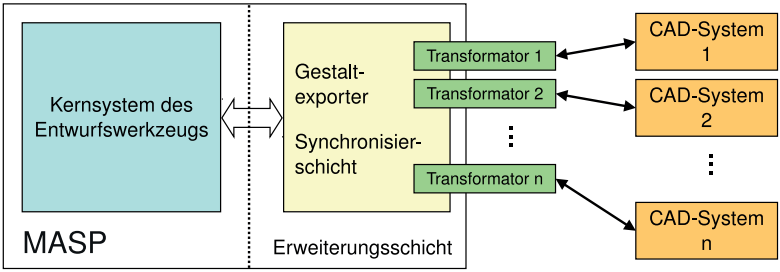


Abbildung 3.15: Abstrahierte Schnittstelle zur Kopplung von *MASP* mit CAD-Systemen

tionen ist nach dem erneuten Laden eine Wiederherstellung der Verknüpfungen zwischen beiden Modellen möglich.

Synchronisierung

Ein zentraler Aspekt der Modellkopplung ist die Konsistenzerhaltung zwischen dem technischen Prinzip und dem CAD-Modell bezüglich der in den Zuordnungselementen beschriebenen Beziehungen (siehe Abschnitt 3.4.3). Da beide Modelle vom Nutzer bearbeitet werden können, muss eine Verfolgung relevanter Änderungen vorgenommen werden. Dies bedeutet, dass alle Manipulationen an Prinzipelementen in den ihnen entsprechenden Gestaltmerkmalen berücksichtigt werden, während Änderungen prinziprelevanter Gestaltparameter zur Anpassung des technischen Prinzips führen müssen. In *MASP* liefert der zugrundeliegende Constraint-Solver Informationen über von Änderungen betroffene Geometrien und Parameter. Ähnliche Konzepte zur Änderungsverfolgung finden sich in CAD-Systemen. Hier kommen z. B. Zeitstempel oder Bauteilversionen zur Anwendung, die bei jeder Manipulation entsprechender Modellbestandteile inkrementiert werden und somit als Hinweis auf geänderte Inhalte dienen.
Bezüglich der Richtung und des Zeitpunktes der Synchronisierung lassen sich unterschiedliche Modi definieren. Änderungen können einerseits automatisch mit ihrem Auftreten oder auf Veranlassung des Nutzers in das jeweils andere Modell übernommen werden. Weiterhin ist die Betrachtung beider Modelle als gleichberechtigt (bidirektionale Propagierung von Änderungen) oder als bevorzugt und nachgeordnet (Master-Slave) möglich. Die letztere Vorgehensweise überträgt Änderungen ausschließlich von einem Modell in das andere. Dort vorgenommene Änderungen werden verworfen oder von vornherein blockiert.

3.6.3 Methoden der Erzeugung von Volumenkörpern

Dieser Abschnitt beschäftigt sich mit der Frage, wie unter Nutzung eines CAD-Systems die Volumenkörpererzeugung so gestaltet wird, dass einerseits eine Unterstützung des Nutzers bei der Konkretisierung des technischen Prinzips zur Grobgestalt gegeben ist, andererseits die Erzeugung der bidirektionalen Beziehungen automatisiert wird. Hierzu werden eine Reihe von Methoden vorgeschlagen, die unterschiedliche Stärken und Schwächen aufweisen. Sie können auf Zusammenbauebene (Assembly), prinzipiell auch auf Bauteilebene, kombiniert werden.

Skelett-Methode

Die Skelett-Methode stellt das einfachste der betrachteten Verfahren dar und bildet gleichzeitig die Grundlage für alle nachfolgenden Methoden. Der Name leitet sich aus der Vorgehensweise ab, bei der mittels Hilfslinien, -kreisen und -punkten sowie Bemaßungen ein Skelett des Zielbauteils im CAD-System erstellt wird. Dieses repräsentiert die funktionsrelevanten geometrischen Maße und Merkmale der zu konkretisierenden Prinzipelemente. Abbildung 3.16 verdeutlicht das Verfahren am Beispiel eines ternären Getriebegliedes. Dieses wird in Form zweier bemaßter Hilfslinien repräsentiert, die einen gemeinsamen Punkt besitzen und zwischen denen ein entsprechender Winkel definiert ist. Die untere der beiden Hilfslinien ist im Koordinatenursprung horizontal verankert. Damit ist die Skizze vollständig definiert.

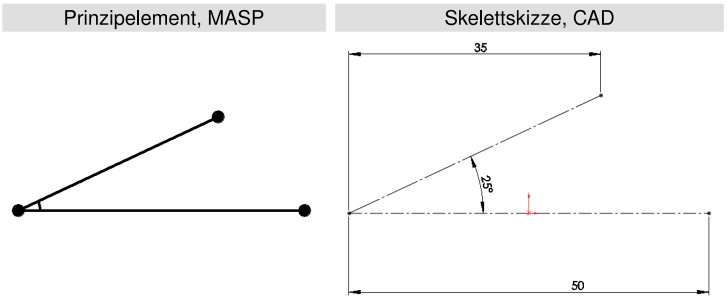


Abbildung 3.16: Übertragung der funktionsrelevanten geometrischen Maße als Hilfselemente

Bis zu diesem Punkt arbeitet das Verfahren automatisch. Die nachfolgende Definition des eigentlichen Volumenkörpers erfordert einige manuelle Arbeitsschritte. Das Zwischenergebnis besteht in einer Basisskizze (Skelett), die die ein-eindeutige

Zuordnung der Prinzipinformationen zu den Skizzenbestandteilen zulässt. Auf der Grundlage der Skelettskizze erarbeitet der Anwender die Bauteilskizze. Wichtig ist hierbei die Schaffung von Beziehungen zwischen den eingegebenen Skizzenelementen und den Hilfselementen (Skelett) in der Form, dass funktionsrelevante Gestaltmerkmale an die Skelettskizze gebunden werden. Der Volumenkörper des Bauteils entsteht durch Austragung (Sweep) der Bauteilskizze (Abbildung 3.17).

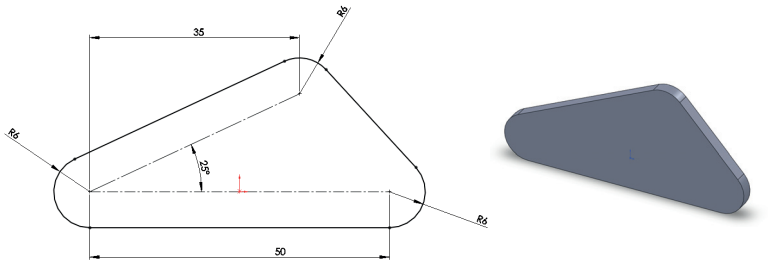


Abbildung 3.17: Definition der Bauteilskizze in Abhängigkeit des Skelettes und Austragung zum Volumenkörper

Als Ergebnis des Prozesses entsteht im CAD-System ein Bauteil, dessen funktionsrelevante Gestaltmerkmale über die Parameter der Skelettskizze (und damit der Prinzipskizze) steuerbar sind. Weiterhin werden Veränderungen der funktionsrelevanten Merkmale des Volumenkörpers nur über die zugrundeliegende Skelettskizze vorgenommen. So entstehende Parameteränderungen lassen sich auf das Prinzipelement zurückführen, da die Zuordnung seiner Eigenschaften zu denen der Skelettskizze bekannt ist. Damit ist die bidirektionale Kopplung beider Modelle erreicht.
Die Vorteile des Verfahrens bestehen in der Einfachheit seiner softwaretechnischen Implementierung, seiner Robustheit und den kaum eingeschränkten Einflussmöglichkeiten des Konstrukteurs. Demgegenüber stehen der manuelle Aufwand bei der Gestaltfindung und die Anforderung an den Konstrukteur, Skizzen und Volumenkörper in Abhängigkeit von der Skelettskizze zu definieren.

Vorlagenmethode

Die Skelettmethode erfordert einen gewissen manuellen Aufwand bei der Gestaltfestlegung. Eine Vereinfachung dieses Prozesses ist wünschenswert und kann beispielsweise durch die Verwendung eines Kataloges von Lösungen oder Bauteilvorlagen erfolgen. Diese Form der Unterstützung ist leicht zu realisieren, da eine mit der

Skelettmethode zu einem Prinzipelement erzeugte Gestaltvariante als Lösung für alle gleich strukturierten Prinzipelemente herangezogen werden kann.
Dies ist möglich, da alle diese Lösungen auf identischen Skelettskizzen beruhen und somit eine automatische Verknüpfung mit dem Prinzipelement vorgenommen werden kann (Abbildung 3.18). Gleichzeitig ist damit die bidirektionale Kopplung

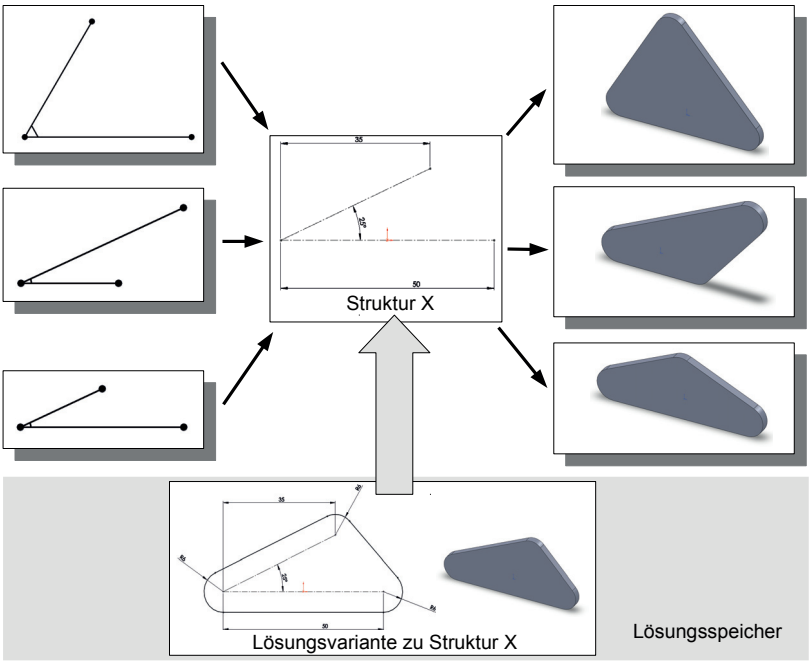


Abbildung 3.18: Vorlagenmethode: Bibliothek von Lösungsvarianten für eine Prinzipstruktur

zwischen Prinzip und Gestalt sichergestellt. Die Lösungsvarianten werden als Vorlagen in einer Bibliothek gesammelt und den entsprechenden Prinzipelementen zugeordnet. Wählt der Anwender eine Lösung aus, erstellt das Entwurfswerkzeug eine Kopie des Datensatzes und fügt ihn in das CAD-Modell ein. Auf diese Weise entstehen unabhängig voneinander parametrisierbare Instanzen der Vorlage. Jede so gebildete Instanz kann ihrerseits weiterbearbeitet und als neue (detaillierter ausgearbeitete) Vorlage in die Bibliothek aufgenommen werden.
Die Vorlagenmethode fördert die Wiederverwendung bereits erstellter Lösungen. Sie ist verhältnismäßig leicht zu implementieren. Die Erzeugung der Gestaltvari-

anten kann interaktiv und ohne Programmierkenntnisse durchgeführt werden. Als nachteilig erweist sich die Notwendigkeit, für jeden Typ von Prinzipelement (oder Kombinationen aus solchen) mindestens eine Gestaltvariante zu erstellen. Interaktiv erstellte Lösungen sind im Allgemeinen spezifisch für ein bestimmtes CAD-System. Dies widerspricht in Teilen dem Konzept einer verallgemeinerten Schnittstelle (siehe 3.6.2). Gestaltlösungen für Kopplungen sind nur eingeschränkt interaktiv zu modellieren, da von ihnen gestellte Gestaltforderungen an gekoppelte Bauteile Programmlogik (z. B. als Skriptcode) erfordern können.

Modulare Methode

Die Skelett- und Vorlagenmethoden erstellen Gestaltlösungen für festgelegte Gruppen von Prinzipelementen. Betrachtet man die Anzahl hierbei möglicher Kombinationen, so offenbaren sich die Einschränkungen dieser Verfahren bezüglich ihrer Flexibilität. Bereits mit einfachen Getriebegliedern lassen sich unterschiedlichste Topologien formen (Abbildung 3.19). Für jede einzelne sind gesonderte CAD-Vorlagen erforderlich.

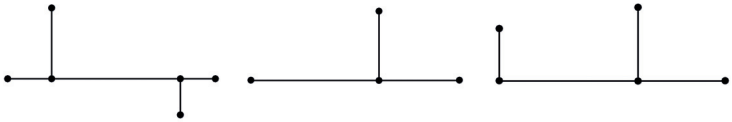


Abbildung 3.19: Beispiele für unterschiedliche Strukturen von Getriebegliedern

Prinzipelemente und aus ihnen gebildete Gruppen setzen sich aus einer begrenzten Anzahl von Basiselementen zusammen. Es liegt daher nahe, auch auf der Seite des CAD-Systems ein Baukastensystem aus korrespondierenden, beliebig kombinierbaren Gestaltbeschreibungselementen zu erstellen. Dieser Gedanke bildet die Grundlage für die Modulare Methode. Jedes Prinzip- oder Basiselement bewirkt die Erzeugung bestimmter Features des zu erstellenden Bauteils. Die so entstehenden unabhängigen Teilvolumen verschmelzen mittels Boolescher Operationen zu einem Gesamtkörper (Abbildung 3.20). Die Abstimmung der Bestandteile des Baukastensystems beeinflusst die Qualität und die Flexibilität der Lösungsmöglichkeiten. Unzureichend aufeinander abgestimmte Komponenten können zu unsinnigen Volumenkörpern führen. Auf der anderen Seite hat der vollständige Ausschluss potentiell problematischer Kombinationen Einschränkungen der Flexibilität zur Folge.
Zu jedem Basiselement lassen sich unterschiedliche Gestaltausprägungen definieren. Ihre Kombination ermöglicht eine große Vielfalt an Varianten zur Beschreibung der

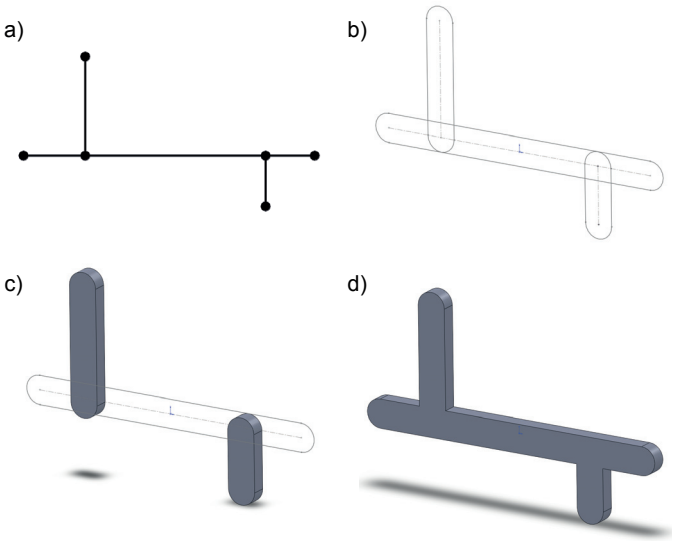


Abbildung 3.20: Erzeugung unabhängiger Teilkörper und anschließende Verschmelzung mittels boolescher Operationen

Bauteilgestalt. Der Aufwand für den Anwender beschränkt sich auf den Konfigurationsprozess und die Auswahl der Teil-Features. Das Konzept ermöglicht außerdem die unkomplizierte Integration der Gestaltforderungen von Kopplungen, indem diese Teil-Features für die zu koppelnden Bauteile liefern.
Die Zusammensetzung der Bauteile aus Teilvolumen kann zu ungünstigen Bauteilstrukturen führen, die für eine Weiterverwendung nur bedingt geeignet sind. Insbesondere die Definition des Gesamtvolumens auf der Basis voneinander unabhängiger Skizzen spiegelt die Grundstruktur und die Intention des Bauteils möglicherweise nur eingeschränkt wider und erschwert somit die nachfolgende Ausarbeitung des Entwurfs.

Skizzenmethode

Das Ziel der Skizzenmethode ist die Erzeugung von Bauteilskizzen, die dem Ergebnis der Arbeitsweise eines Konstrukteurs entsprechen. Ähnlich der Modularen Methode betrachtet sie den systematischen Aufbau von Prinzipskizzen aus Grundelementen. Bei der Skizzenmethode ergibt sich das Bauteil jedoch nicht aus Teilkörpern, sondern ist das Ergebnis einer zusammengesetzten Skizze.

Die Grundlage der Skizzenmethode ist eine Bibliothek aus Teilskizzen, die den Basiselementen des technischen Prinzips zugeordnet sind (Abbildung 3.21). Jede Teilskizze verfügt über eine Reihe geometrischer Schnittstellen (Punkte, Liniensegmente). Diese dienen zur Verbindung mit anderen Teilskizzen beim Einfügen in eine Bauteilskizze. Die Schnittstellen werden unter Anwendung geometrischer Constraints (z. B. Identität von Punkten, Kollinearität) miteinander verknüpft. Die entstehende Gesamtskizze dient zur Erzeugung des Volumenkörpers (Abbildung 3.21, rechts unten).

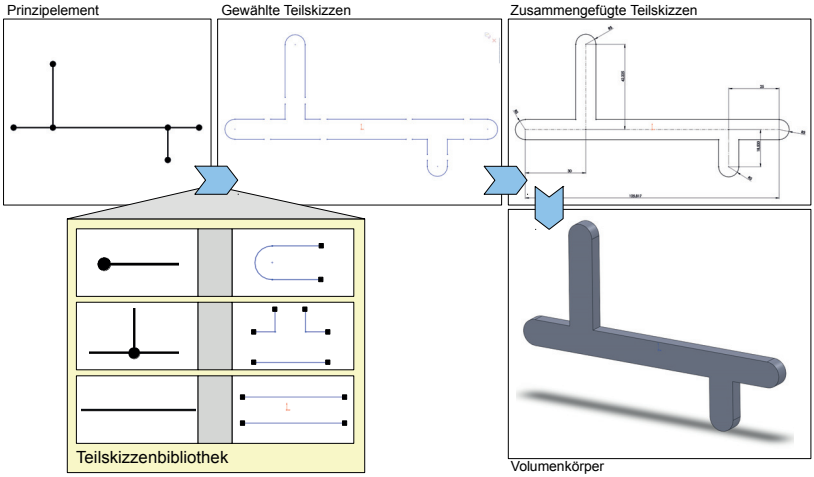


Abbildung 3.21: Zusammensetzung einer Skizze aus Skizzenbausteinen mit feststehenden Schnittstellen

Als Ergebnis der Skizzenmethode entstehen übersichtlich strukturierte Bauteilskizzen, die für eine Weiterbearbeitung gut geeignet sind. Die Implementierung des Verfahrens ist verhältnismäßig aufwendig, da sie eine Beschreibungssprache für Teilskizzen und ihre Schnittstellen erfordert. Darüber hinaus setzt die Anwendung der Beschreibungssprache ein hohes Maß an Sorgfalt voraus, um sinnvoll kombinierbare Teilskizzen und vollständig definierte Gesamtskizzen zu erhalten.

3.6.4 Implementierung der CAD-Schnittstelle

Die Implementierung der CAD-Kopplung basiert auf der bei CAD-Systemen weit verbreiteten COM-Schnittstelle. COM (*Component Object Model*) ist ein Teil des Windows-Betriebssystems und stellt keine CAD-Schnittstelle im eigentlichen

Sinne dar, da sie lediglich als Kommunikationsschicht zwischen Softwarepaketen dient, über die anwendungsspezifische Inhalte ausgetauscht werden. Die COM-Schnittstelle kann über verschiedene Programmiersprachen angesprochen werden, so auch mittels C++, das gleichzeitig die Implementierungsgrundlage von *MASP* darstellt. Die prototypische Implementierung von Adaptern für die CAD-Systeme *SolidWorks* und *Autodesk Inventor* entstand teilweise in studentischen Arbeiten [Möb06, Wei08].
Das in *MASP* eingesetzte Zuordnungsmodell wurde im Zuge der Schnittstellenimplementierung so erweitert, dass anstelle interner Szenengraph-Objekte auch externe Objekte und Parameter des CAD-Systems referenziert werden können. Weiterhin werden Objektbeziehungen wie Constraints in das CAD-System übertragen, so dass das Modell auch ohne *MASP* zur Bewegungssimulation im CAD-System eingesetzt werden kann. Die Implementierung der Volumenerzeugung umfasst die Skelett- und Vorlagenmethoden. Zur Modularen Methode und zur Skizzenmethode existieren experimentelle Umsetzungen („Proof-of-Concept"), die im Zusammenhang mit den oben erwähnten studentischen Arbeiten entstanden. Unter Einsatz der Referenzierung der CAD-internen Bezeichner ist der Prototyp in der Lage, das Modell in *MASP* und im CAD-System zu speichern und zu einem späteren Zeitpunkt unter Wiederherstellung der Kopplung zu laden.

3.6.5 Anwendungsbeispiel

Die Abbildungen 3.22 bis 3.25 zeigen den Ablauf der phasenübergreifenden Modellierung unter Verwendung der CAD-Schnittstelle am Beispiel eines Filmgreifergetriebes. Ausgehend von einem bereits erstellten Prinzipmodell des Getriebes im Entwurfswerkzeug *MASP* soll eine Grobgestalt entwickelt werden. Hierzu erfolgt der Aufbau einer Verbindung zum CAD-System *SolidWorks*, über die anschließend Bauteildokumente mit Basisskizzen für die Getriebeglieder und Drehgelenke erzeugt werden. Anhand der in *MASP* modellierten Elementverknüpfungen werden die Bauteile zu einer Baugruppe mit geeigneten Beziehungen (Constraints) zusammengefügt. Entsprechend der Skelettmethode (siehe Abschnitt 3.6.3) beginnt der Anwender, auf der Grundlage der Basisskizzen im CAD-System die Bauteilgeometrien zu entwerfen (Abbildung 3.22).
Es folgt die Ausgestaltung der Bauteile mit Skizzenelementen oder Features, die sich weiterhin an den Vorgaben der Basisskizzen des Skelettes orientieren. Das Ergebnis ist ein beliebig detailliertes CAD-Modell des Filmgreifergetriebes (Abbildung 3.23). Treten in der Folge Änderungen am technischen Prinzip auf, z. B. durch die Anpassung des Längenparameters eines Getriebegliedes, erfolgt eine Übertragung des geänderten Wertes in die Basisskizze des zugeordneten Bauteils. Aufgrund der

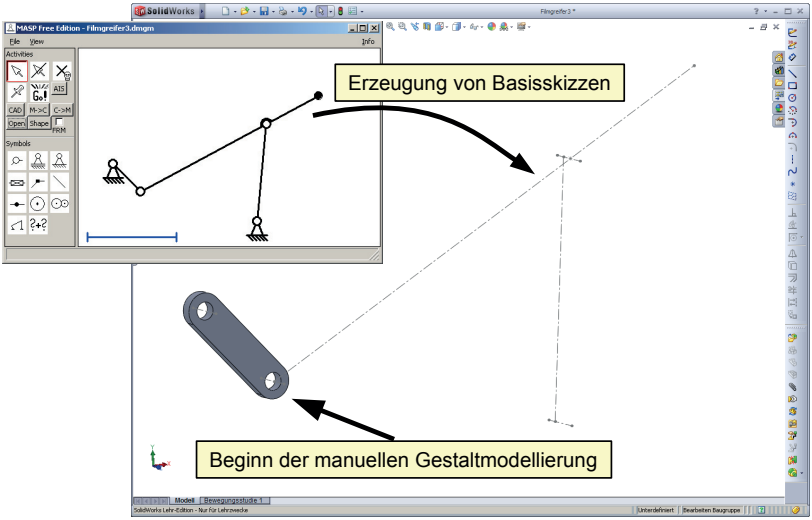


Abbildung 3.22: Automatische Erzeugung von Basisskizzen im CAD-Modell auf der Basis des technischen Prinzips und Beginn der Gestaltmodellierung (manuell)

während der Modellierung erzeugten Abhängigkeiten propagiert sich die Änderung auf angrenzende Geometrieelemente (Abbildung 3.24).
Darüber hinaus ist es möglich, die Gestalt nahezu beliebig zu manipulieren. Da prinziprelevante Geometrieparameter über die Basisskizze definiert und grundsätzlich nur über diese beeinflussbar sind, gelingt die automatische Rückführung der Änderungen auf die zugehörigen Prinzipelemente (Abbildung 3.25). Somit ist auch die zweite, schwieriger zu realisierende Richtung der bidirektionalen Kopplung des technischen Prinzips mit dem CAD-Modell ermöglicht.

3.6.6 Einschätzung und Bemerkungen

Während der Erarbeitung und Umsetzung des Konzepts zeigten sich verschiedene Eigenheiten und Grenzen der Lösung sowie der verwendeten CAD-Schnittstellen. Bezüglich der Volumenerzeugung lässt sich festhalten, dass sich die Skelettmethode im Vergleich zu den übrigen Verfahren als relativ robust erweist. Dies ist auf ihre Einfachheit und nicht zuletzt auf die Einbeziehung der Kompetenz des Ingenieurs zurückzuführen. Als Konsequenz erfordert die Skelettmethode einen höheren manuellen Aufwand.

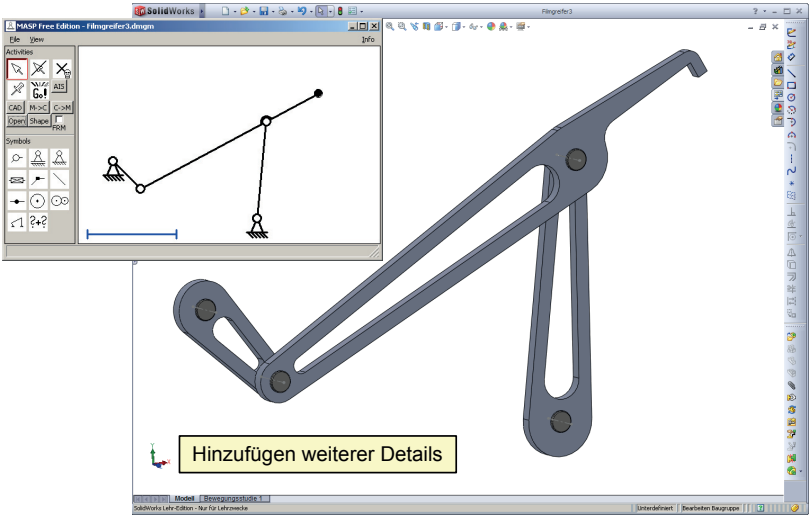


Abbildung 3.23: Detaillierung der Grobgestalt mit den Interaktionsmethoden des CAD-Systems

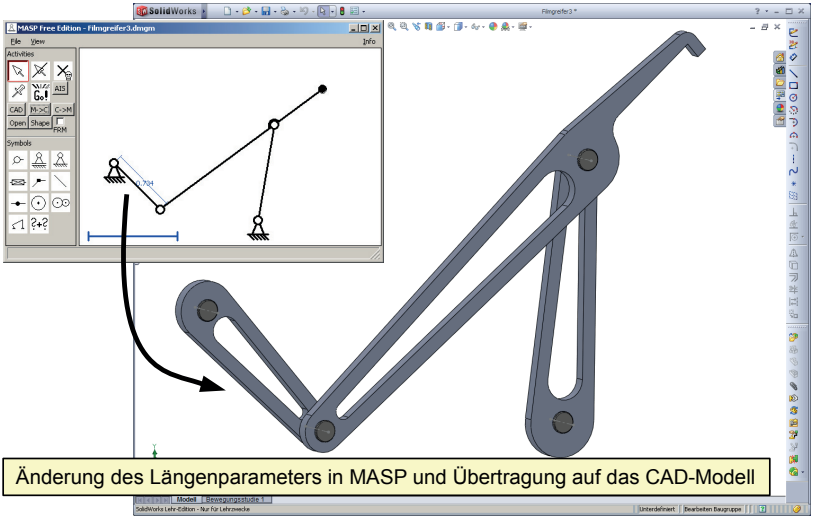


Abbildung 3.24: Übertragung von Änderungen aus dem *MASP*-Modell in das CAD-Modell

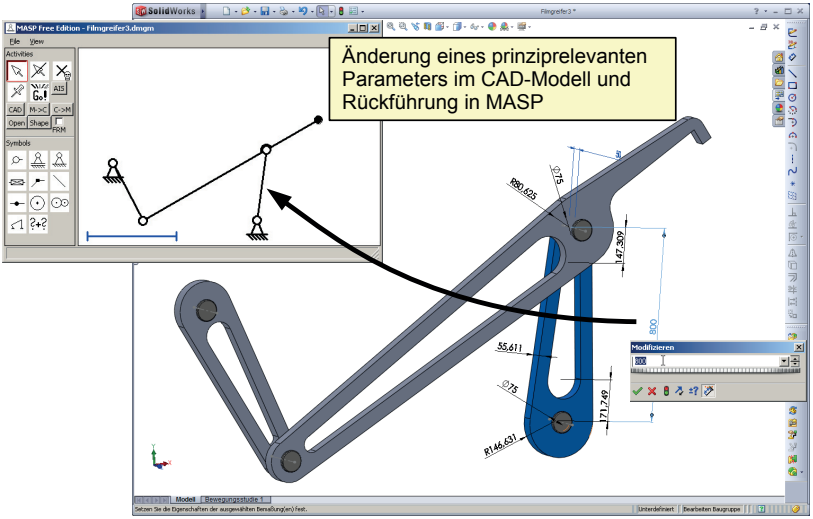


Abbildung 3.25: Übertragung von Änderungen aus dem CAD-Modell in das Modell von *MASP*

Ein gesteigerter Automatisierungsgrad entlastet den Anwender, führt andererseits allerdings zu weniger robusten Lösungen. So ist nicht immer sichergestellt, dass automatisch erzeugte Features in ihrem Zusammenwirken sinnvolle Volumenkörper ergeben. Beispielsweise ist es möglich, dass Gestaltforderungen von Kopplungen nicht berücksichtigt werden können, wenn der Basiskörper nicht die notwendigen Voraussetzungen erfüllt oder Kollisionen mit anderen Forderungen auftreten. Eine solche Situation ist in Abbildung 3.26 dargestellt: Ein Getriebeglied steht mit einem Dreh- und einem Schubgelenk in Beziehung. Während die gewählte Gestaltvariante des Drehgelenks eine Bohrung fordert, sieht das Schubgelenk eine Führungseinrichtung vor. Werden beide Forderungen umgesetzt, beeinträchtigen sie ihre Funktion gegenseitig.
Eine automatische Erkennung solcher Konflikte ist möglich, ihre Auflösung bleibt in erster Linie dem Anwender überlassen. Eine rechnerunterstützte Behandlung ist mit beträchtlichem Aufwand verbunden und im Rahmen dieser Arbeit nicht vorgesehen. Lediglich der Gedanke eines Baukastensystems aus aufeinander abgestimmten Bestandteilen bietet eine gewisse Führung in Richtung konfliktarmer Lösungen. Eine zu starke Fokussierung auf diese Strategie führt andererseits zu verringerter Flexibilität, was im Gespräch mit Konstrukteuren häufig als unerwünscht herausgestellt wurde.

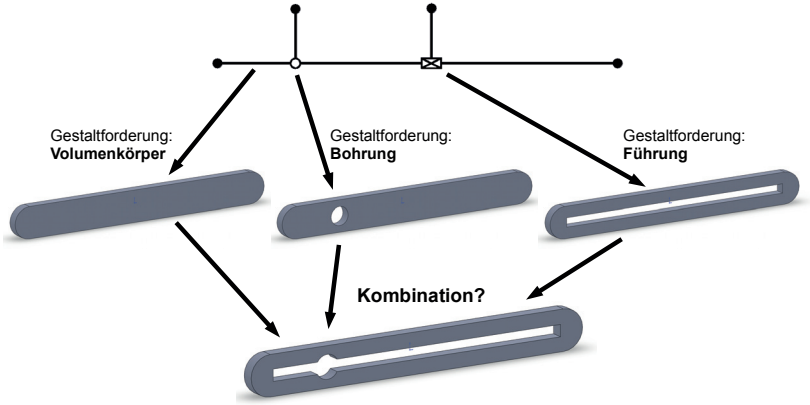


Abbildung 3.26: Widersprüchliche Gestaltforderungen

Andere Einschränkungen resultieren aus den Eigenschaften der beteiligten Softwareprodukte. Beispielsweise sollte aus konstruktionstechnischer Sicht die Erzeugung der Bauteilgestalt mit der Definition der Koppelstellen und Wirkflächen beginnen. Die durch das CAD-System vorgegebene Arbeitsweise hingegen fügt die Geometrie der Koppelstellen erst als Features zu einem zuvor erstellten Volumenkörper hinzu. Folglich ist auch die CAD-Schnittstelle an diese Vorgehensweise gebunden, die somit möglicherweise nicht den Vorstellungen des Konstrukteurs gerecht wird.
Die CAD-Schnittstelle erzeugt anfangs Bauteile ohne Volumenelemente, die anhand ihrer Koordinatensysteme kinematisch verknüpft werden. Da das in *MASP* erstellte Modell zweidimensional ist, fehlt jede Tiefeninformation bei der Anordnung der Bauteile. Diese ergibt sich erst aus den später modellierten Geometrien und bleibt bis dahin undefiniert. Das unfertige Modell verfügt somit über zusätzliche Freiheitsgrade, die sich auf Interaktionen störend auswirken (Modell kann „zerreißen"). Ein anderes Problem betrifft die Erzeugung vollständig definierter Skizzen. Besonders das Zusammensetzen von Teilskizzen erfordert eine sorgfältige Koordination, so dass alle Freiheitsgrade der Skizzenelemente gebunden werden, gleichzeitig aber keine Überbestimmung auftritt.
Auch die Programmierschnittstellen (APIs) der CAD-Systeme geben Anlass zur Kritik. Häufig sind sie lückenhaft dokumentiert und umständlich implementiert. Beispielsweise erlaubt die API von *SolidWorks* keine direkten Zugriff auf das interne Modell. Manipulationen sind lediglich über simulierte Benutzereingaben möglich (Beispiel: Anstelle von *„Verbinde A mit B"* ist *„Wähle A, wähle B, Verbinde alle derzeit gewählten Objekte"* notwendig). Dies kann zu Seiteneffekten führen. Wird

beispielsweise ein Skizzenpunkt auf einer existierenden Linie erzeugt, so entsteht gleichzeitig und automatisch eine (unerwünschte) Beziehung *Punkt-auf-Linie*, da das Interaktionssystem von *SolidWorks* dies vorsieht. Auch die Verwendung von Begriffen aus natürlichen Sprachen als Bezeichner zur Objektidentifikation erschwert das algorithmische Auffinden bestimmter Modellbestandteile (z. B. die globale *Ebene vorn* ist in der englischen Sprachversion über die Bezeichnung *Front Plane* aufzufinden).
Nicht zuletzt ist der Faktor Mensch zu berücksichtigen. Der Anwender verfügt über eine große Auswahl an Möglichkeiten, das Modell absichtlich oder versehentlich so zu verändern, dass für die automatische Synchronisierung notwendige Bestandteile nicht mehr zu identifizieren sind (Löschen, Umbenennen). Hilfsmittel zur Sperrung relevanter Modellbestandteile existieren in CAD-Systemen kaum.
Es bleibt festzuhalten, dass die phasenübergreifende Werkzeugkopplung durchaus realisierbar ist, auch wenn dabei Spezialfälle und Eigenheiten der eingesetzten Werkzeuge zu berücksichtigen sind. Insbesondere bietet die automatisierte Volumenerzeugung Raum für Verbesserungen und weitere Forschung. Dennoch ist mit der bidirektionalen Modellkopplung ein entscheidendes Ziel des Vorhabens erreicht. Die damit ermöglichte automatische Parametersynchronisierung stellt eine hilfreiche Arbeitserleichterung im Umgang mit den Modellen des technischen Prinzips und der Grobgestalt dar.

3.7 Kopplung von MASP, CAD und VR-Systemen

Die in den vorangegangenen Abschnitten beschriebene Kopplung von *MASP* mit einem CAD-System kann nicht nur zur Synchronisierung von Modellen unterschiedlicher Entwurfsphasen, sondern auch zur Parametrisierung von CAD-Modellen in der Virtuellen Realität (VR) eingesetzt werden. Die Grundidee basiert hierbei auf der Reduktion der Anzahl manipulierbarer Parameter auf einen in VR-Umgebungen beherrschbaren und für die Lösung einer bestimmten Aufgabe notwendigen Umfang.

3.7.1 Problemstellung und Grundidee der Lösung

In VR-Umgebungen werden CAD-Modelle üblicherweise als exportierte, statische Dreiecksmengen dargestellt. Damit gehen Semantik und Parametrik der Features des Modells verloren. Modelle können dann lediglich räumlich transformiert und gegebenenfalls entsprechend ihrer Bauteilstruktur zerlegt werden. Damit reduziert sich der Nutzen der VR-Darstellung meist auf ein Hilfsmittel zur Verständnisbildung und zur Kommunikation.

Sollen alle Möglichkeiten der CAD-Modellierung in einer VR-Umgebung verfügbar gemacht werden, wäre die Integration eines CAD-Kerns und die Implementierung einer den technischen Möglichkeiten von VR-Systemen angepassten Nutzeroberfläche notwendig (Ein- und Ausgabegeräte). Im Vergleich zur Nutzung von Arbeitsstationen mit konventioneller Maus- und Tastatursteuerung muss hier vor allem mit Problemen wie der häufig kleineren Auflösung und dem schwächeren Kontrast der Anzeigegeräte sowie der geringeren Feinfühligkeit der Eingabegeräte (z. B. „Flystick“ am ausgestreckten Arm) umgegangen werden.
Von Teillösungen wie der Aktualisierung geänderter CAD-Daten im VR-Modell abgesehen (z. B. [SPSS09]), ist die Umsetzung einer solchen VR-CAD-Software bislang nicht bekannt. Einer der Gründe hierfür ist zweifellos der Aufwand, der mit dem der Programmierung eines vollständigen CAD-Systems vergleichbar ist, während angesichts der geringen Verbreitung von VR-Systemen kein nennenswerter Markt für solche Lösungen existiert.
Ein anderer Ansatz, der zumindest einen Teil der Funktionalität von CAD-Systemen in VR-Umgebungen überträgt, besteht in der gleichzeitigen Verwendung von VR- und CAD-Software unter Einsatz einer Kommunikationsschnittstelle. CAD-Modelle werden dabei weiterhin als tesselierte Daten in der VR-Umgebung dargestellt, zusätzlich allerdings mit (ausgewählten) Informationen zur Parametrik angereichert. Anhand dieser Informationen können interaktiv Änderungswünsche formuliert und an das CAD-System übermittelt werden. Das aus den Änderungen resultierende CAD-Modell wird (in Teilen) neu tesseliert und in das VR-System importiert. Gelingt die Abarbeitung dieser Vorgänge in kurzer Zeit, wird der Anwender in die Lage versetzt, interaktiv nachvollziehbare Änderungen am dargestellten Modell vorzunehmen.
Der beschriebene Ansatz ermöglicht zwar nicht die Konstruktion neuer Bauteile, erlaubt aber, vorhandene Modelle zu manipulieren und Varianten zu bilden. Dabei liegt eine besondere Anforderung in der geeigneten Auswahl und Darstellung der beeinflussbaren Parameter. Welche Parameter für den Anwender nützlich sind, hängt von der Aufgabenstellung ab. Ein mögliches Auswahlkriterium im Kontext von Bewegungssystemen sind Parameter, die das zugrundeliegende technische Prinzip definieren. Auf dieser Parameterauswahl baut die in den folgenden Teilabschnitten beschriebene Lösung auf. Sie greift auf die in Abschnitt 3.6 beschriebene Kopplung von *MASP* mit einem CAD-System zurück, bei der zweckmäßig strukturierte CAD-Modelle anhand einer überschaubaren Anzahl von (Prinzip-)Parametern manipuliert werden. Das Gesamtkonzept wurde unter Nutzung von *MASP*, der CAD-Software *SolidWorks* und der VR-Software *VD2* (*Virtual Design 2*) prototypisch umgesetzt. Die Erprobung der Prototypen erfolgte in der VR-Einrichtung *FASP* (Flexible Audiovisuelle Stereoprojektion) der TU Ilmenau, einer dreiseitigen

Projektionseinrichtung mit veränderlichem Winkel zwischen den Projektionsflächen, die neben der Visualisierung auch die Auralisierung von Produktmodellen erlaubt [HSR+11, HRWT10].

3.7.2 Notwendige Teillösungen

Zur Realisierung der vorgeschlagenen Lösung sind einige Voraussetzungen zu schaffen. Dazu gehört die Kopplung von *MASP* mit einem CAD-System, die bereits in den vorangegangenen Abschnitten dokumentiert wurde. Eine weitere Teillösung ist die Kopplung von *MASP* mit einer VR-Umgebung, um die constraint-basierte Modellierung von Bewegungssystemen in der Virtuellen Realität und damit die Parametrisierung von CAD-Modellen zu ermöglichen. Darüber hinaus muss das Problem der Tesselierung von CAD-Modellen und deren automatischer Aktualisierung in der VR-Umgebung gelöst werden. In einem letzten Schritt werden die genannten Bausteine zu einer Gesamtlösung zusammengefügt.

3.7.3 Kopplung von MASP mit einer VR-Umgebung (FASP)

Um die Manipulation der Parameter von CAD-Modellen mittels *MASP* aus einer VR-Umgebung heraus vornehmen zu können, muss *MASP* zunächst mit dieser gekoppelt werden. Da zu erwarten ist, dass beide Softwarepakete auf unterschiedlichen Rechnern laufen, bietet sich die Kopplung über eine Netzwerkschnittstelle an (Abbildung 3.27). Die CAD-Software selbst kommt bei dieser Teillösung vorerst nicht zum Einsatz.
Für die hier beschriebene Lösung kam das VR-System *VD2* (*Virtual Design 2* der *ICIDO GmbH*, vormals *VRCom GmbH*) zum Einsatz. Es erlaubt die Programmierung von Zusatzmodulen mit nutzerdefinierter Funktionalität (Plug-Ins). Ein solches Zusatzmodul wurde für die Kopplung mit *MASP* erstellt. Es erhält über ein Netzwerkprotokoll Befehle von *MASP* und setzt diese in Operationen auf dem Szenengraph von *VD2* um. Zu den im Protokoll definierten Operationen zählen:

- Erstellen der Visualisierung eines neuen Prinzipelementes
- Löschen der Visualisierung eines Prinzipelementes
- Transformieren der Visualisierung eines Prinzipelementes
- Änderung von Parametern der Visualisierung eines Prinzipelementes

Prinzipelemente werden als einfache grafische Formen im Szenengraph von *VD2* repräsentiert und entsprechend der in *MASP* berechneten Lage positioniert. Eine Abbildung des eigentlichen kinematischen Modells erfolgt in *VD2* nicht. Das

VR-Modell wird somit von der Bewegungssimulation in *MASP* gesteuert. Die Bedienung von *MASP* erfolgt weiterhin über den Arbeitsplatz des Simulationsrechners (Maus und Tastatur).

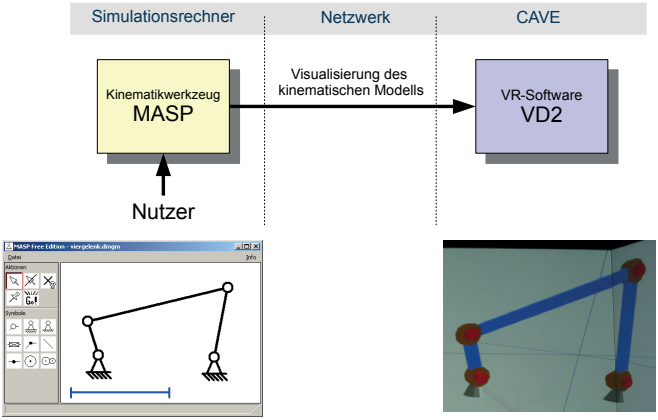


Abbildung 3.27: Kopplung von *MASP* mit der VR-Software *VD2* unter Nutzung eines Netzwerkprotokolls

Zur Schaffung einer tatsächlich immersiven VR-Anwendung muss in einem nächsten Schritt die Bedienung von *MASP* aus der VR-Umgebung heraus ermöglicht werden. Hierfür wird ein Rückkanal von *VD2* zu *MASP* geschaffen, indem Interaktionen mit den in der Virtuellen Realität dargestellten Prinzipelementen registriert und als Steuerbefehle über die Netzwerkschnittstelle an *MASP* übertragen werden (Abbildung 3.28). Dort werden diese Befehle ausgeführt, und es wird mittels des Constraint-Solvers ein konsistentes Modell erzeugt. So entstehende Änderungen werden an *VD2* gesendet und dort in den Szenengraph eingepflegt. Der gesamte Vorgang von der Interaktion bis zur Visualisierung des Ergebnisses findet während einer Zeitspanne statt, die kurz genug ist, um vom Nutzer als verzögerungsfrei wahrgenommen zu werden.
Die direkte Interaktion mit dem Prinzipmodell (Greifen und Bewegen von Prinzipelementen) wird durch ein einfaches Menü ergänzt, das die Auswahl des Interaktionsmodus (Erstellen, Löschen, Modifizieren) und der zu erstellenden Prinzipelemente erlaubt. Das Ergebnis ist eine interaktive VR-Anwendung zur Modellierung und Simulation von Bewegungssystemen, bei der die VR-Umgebung als Benutzeroberfläche für *MASP* fungiert. *MASP* übernimmt dabei die eigentliche Berechnung der kinematischen Modelle.

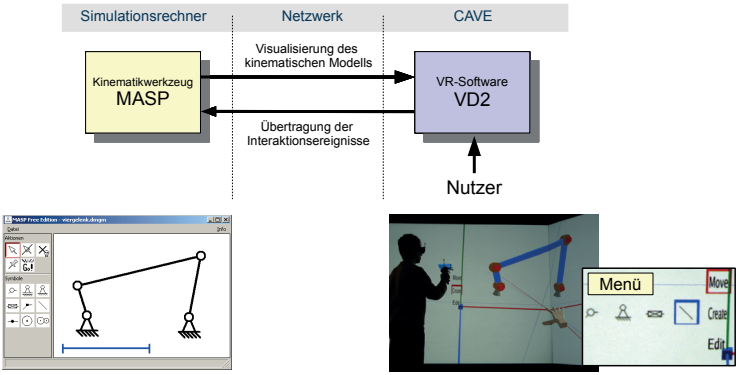


Abbildung 3.28: *VD2* als Benutzeroberfläche für *MASP*

3.7.4 Übertragung tesselierter Modelle aus einem CAD-System in eine VR-Umgebung

Zur Darstellung von CAD-Modellen in einer VR-Umgebung müssen diese zunächst tesseliert werden. Die dazu benötigte Funktionalität ist Bestandteil vieler 3D-CAD-Systeme. Ein exportiertes (tesseliertes) Modell liegt meist als Datei in einem verbreiteten Format wie z. B. VRML vor. Dieses kann in die VR-Software *VD2* importiert und dort visualisiert werden. Bei der Übertragung auf diesem Weg gehen Feature-Semantik und Parametrik der CAD-Modelle größtenteils verloren. Es entstehen statische Dreiecksmodelle, die bestenfalls die ursprüngliche Bauteilstruktur als Hierarchie widerspiegeln. Änderungen müssen daher am CAD-Modell vorgenommen und als aktualisierte Dreiecksmodelle neu in das VR-System übertragen werden.
Um den Export- und Importvorgang zu koordinieren und zu automatisieren, wurde für das CAD-System und die VR-Software jeweils ein Plug-In entwickelt. Das Plug-In des CAD-Systems exportiert auf Anforderung einer steuernden Instanz veränderte Modelle und überträgt diese per Netzwerk an das Plug-In des VR-Systems. Dort wird das Teilmodell geladen und in den Szenengraph eingefügt. Sofern aus der gleichzeitig gepflegten Zuordnungstabelle hervorgeht, dass bereits ein (veraltetes) Teilmodell im Szenengraph existiert, wird dieses zunächst entfernt und an seiner Stelle der aktualisierte Datensatz verwendet (Abbildung 3.29).
Das beschriebene Verfahren ermöglicht die Visualisierung von CAD-Modellen in einer VR-Umgebung, wobei Änderungen an den Modellen im CAD-System in eine laufende VR-Sitzung übernommen werden.

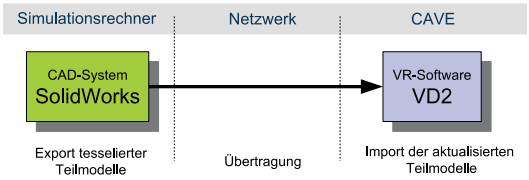


Abbildung 3.29: Übertragung exportierter CAD-Modelle aus *SolidWorks* in die VR-Software *VD2*

3.7.5 Kombination der Teillösungen

Die beschriebenen Teillösungen zur Kopplung von *MASP* mit dem CAD-System *SolidWorks*, von *MASP* mit *VD2* und zur Übertragung tesselierter Modelle von *SolidWorks* nach *VD2* fügen sich zu der in Abbildung 3.30 dargestellten Gesamtlösung zusammen.

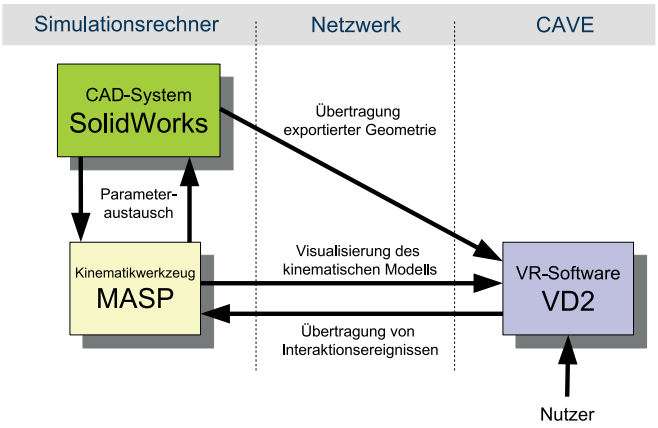


Abbildung 3.30: Gesamtaufbau der Kopplung von *MASP*, *VD2* und *SolidWorks*

In einem ersten Schritt modelliert der Anwender mittels der *MASP*-VR-Schnittstelle ein Bewegungssystem. Dieses liegt als constraint-basiertes Modell in *MASP* und als visualisierte Prinzipelemente im Szenengraph von *VD2* vor. Anschließend wird *MASP* mit dem CAD-System gekoppelt und dort auf Basis des technischen Prinzips ein Volumenmodell erzeugt. Dieser Schritt kann katalogbasiert in der Virtuellen Realität oder manuell an der Arbeitsstation des CAD-Systems erfolgen. Die Bauteile des fertigen CAD-Modells werden als Dreiecksmodelle exportiert und per

Netzwerk an das Plug-In der VR-Umgebung übertragen. Dieses fügt die exportierten Bauteile im Szenengraph unterhalb der Knoten ein, die die Prinzipsymbole repräsentieren, so dass eine Transformation der Bauteile dem Zustand des Simulationsmodells in *MASP* entsprechend erfolgt.
Über die *MASP*-VR-Schnittstelle kann der Anwender nun mit dem Modell des Bewegungssystems interagieren. Zustandsänderungen des *MASP*-Modells (z. B. Bewegungen entsprechend des Freiheitsgrades) werden in Form von Transformationen im Szenengraph umgesetzt. Diese wirken sich sowohl auf die Prinzipsymbole als auch auf die tesselierten CAD-Modelle aus. Bei Änderung von Parametern (z. B. Maße des Bewegungssystems) erfolgt neben der Anpassung der Symbolgeometrien im Szenengraph auch eine Aktualisierung des CAD-Modells. Dieses wird neu exportiert und wie in Abbildung 3.29 beschrieben in die VR-Umgebung reimportiert.
Als Ergebnis entsteht eine Lösung zur Modellierung von Bewegungssystemen in einer VR-Umgebung. Darüber hinaus fungiert das *MASP*-Modell als vereinfachende Schnittstelle zu einem komplexen CAD-Modell, das somit über wenige, ausgewählte Parameter beeinflusst werden kann.

3.7.6 Anwendungsbeispiel

Zur Demonstration der Funktionsweise der beschriebenen Lösung soll an das Beispiel des Filmgreifergetriebes aus Abschnitt 3.6.5 angeknüpft werden. Abbildung 3.31 (links) zeigt das in der VR-Umgebung modellierte Getriebe. Seine Visualisierung erfolgt mittels einfacher räumlicher Prinzipsymbole. Das parallel in *MASP* entstandene, mit dem VR-Modell verknüpfte Prinzipmodell ist auf der rechten Seite der Abbildung dargestellt. Ebenfalls rechts ist das aus dem Prinzipmodell entwickelte Volumenmodell im CAD-System *SolidWorks* abgebildet.
Anschließend erfolgen die Tesselierung des CAD-Modells und der Import in die VR-Umgebung. Dort werden die Dreiecksmodelle der Einzelteile den Visualisierungen der Prinzipelemente zugeordnet und im Szenengraph an den entsprechenden Stellen verankert (Abbildung 3.32, links). Es besteht die Möglichkeit, beide Modelle überlagert (links) oder einzeln (rechts) darzustellen. An die Modelle gerichtete Interaktionsereignisse werden an *MASP* übertragen und dort auf die einzelnen Prinzipelemente angewandt.
Es ist möglich, dass Interaktionen neben Zustandsänderungen auch Parameteränderungen hervorrufen. Ein Beispiel ist in Abbildung 3.33 zu sehen. Aus der VR-Umgebung heraus wurde eine Verlängerung der Antriebskurbel des Filmgreifergetriebes gefordert. Nachdem diese Anforderung an das constraint-basierte Modell in *MASP* übertragen und dort berücksichtigt wurde, erfolgt die Anpassung des CAD-Modells entsprechend der in *MASP* ermittelten Parameterwerte. Die geän-

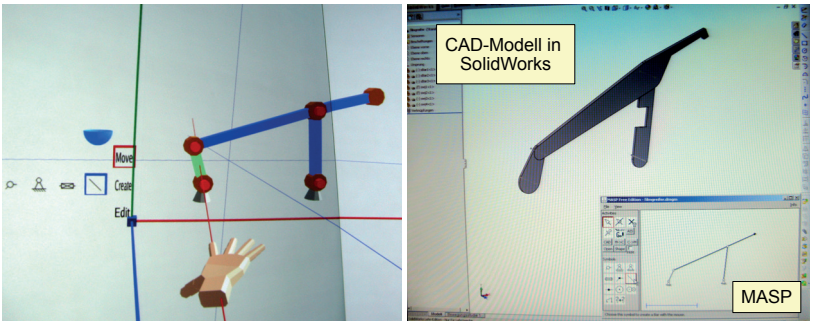


Abbildung 3.31: Beispiel Filmgreifergetriebe in *VD2* (links), *SolidWorks* und *MASP* (rechts)

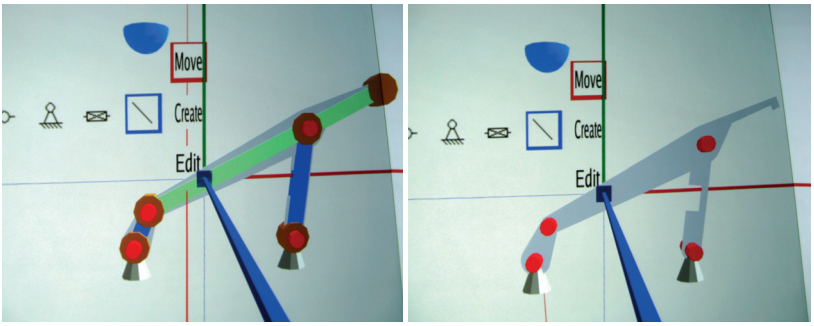


Abbildung 3.32: Filmgreifergetriebe in *VD2* mit (links) und ohne (rechts) Überlagerung der importierten CAD-Modelle mit Prinzipsymbolen

derten Teile des CAD-Modells werden erneut exportiert und in die VR-Umgebung übertragen, wo sie die bisher verwendeten Dreiecksmodelle ersetzen. Abbildung 3.33 (links) zeigt den Zustand des VR-Modells, bei dem der Szenengraph bereits entsprechend der geänderten Maße angepasst ist und der Austausch des Dreiecksmodells unmittelbar bevorsteht. Auf der rechten Seite des Bildes ist das aktualisierte Bauteil (Längenparameter) integriert.

3.7.7 Erfahrungen und Ausblick

Die Beispielimplementierung (Abbildung 3.34) verdeutlicht, dass das vorgeschlagene Verfahren eine zweckmäßige Technik zur Manipulation von CAD-Modellen in

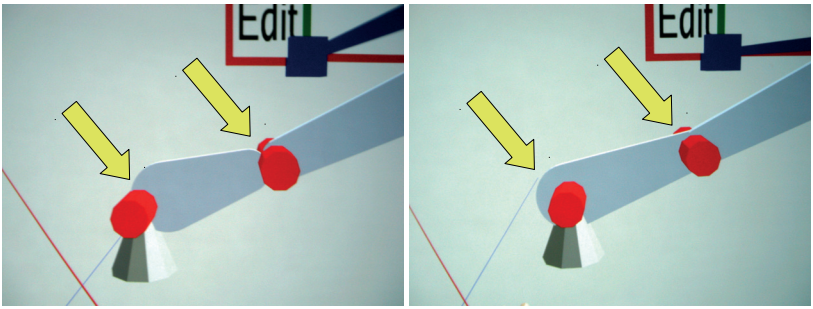


Abbildung 3.33: Geänderter Längenparameter des Getriebegliedes. Links: Transformationen im Szenengraph bereits verändert, tesseliertes Modell noch nicht aktualisiert. Rechts: Geändertes CAD-Modell importiert.

einer VR-Umgebung darstellt. Es ist auf der Basis des derzeitigen Standes der Technik umsetzbar und funktioniert verzögerungsarm genug, um ein interaktives Arbeiten zu ermöglichen. Die notwendigen Interaktionen sind dabei mit VR-typischen Ein- und Ausgabegeräten beherrschbar (Auflösung, Feinfühligkeit usw.).
Der beschriebene Ansatz basiert auf der aufgabenbezogenen Auswahl von Parametern eines CAD-Modells, die in einer VR-Umgebung visualisiert und zur vereinfachten Manipulation des Modells herangezogen werden. Das mit dem CAD-Modell gekoppelte, in *MASP* modellierte technische Prinzip ist dabei nur eine mögliche Parameterauswahlstrategie. Grundsätzlich ist jeder Parameter eines CAD-Modells zur Darstellung in einer VR-Umgebung geeignet. Neben der Visualisierung des Parameterwertes sind grafische Hilfsmittel zu seiner Beeinflussung sinnvoll („Handles“, Griffe, z. B. Maßangaben wie in CAD-Systemen). Zu den möglichen Strategien zur Zusammenstellung einer überschaubaren Menge beeinflussbarer Werte gehören die manuelle Auswahl von Parametern, die Visualisierung der Parameter eines betrachteten Einzelteils, einer bestimmten Skizze oder der Zusammenbauabhängigkeiten in einer Baugruppe. In diesen Fällen lässt sich die Lösung auf der Basis direkter Kommunikation zwischen dem CAD-System und der VR-Umgebung, also ohne *MASP* als Vermittler, realisieren.
Die interaktive Modellierung und Simulation von Bewegungssystemen stellt für sich betrachtet eine nützliche Funktionalität in VR-Umgebungen dar. Die Ausdehnung der Umsetzung auf räumliche Getriebe wäre ein folgerichtiger nächster Schritt.
Die Beispielimplementierung wurde lediglich mit Modellen von geringer Komplexität getestet. Interessant wären daher Aussagen über das Laufzeitverhalten

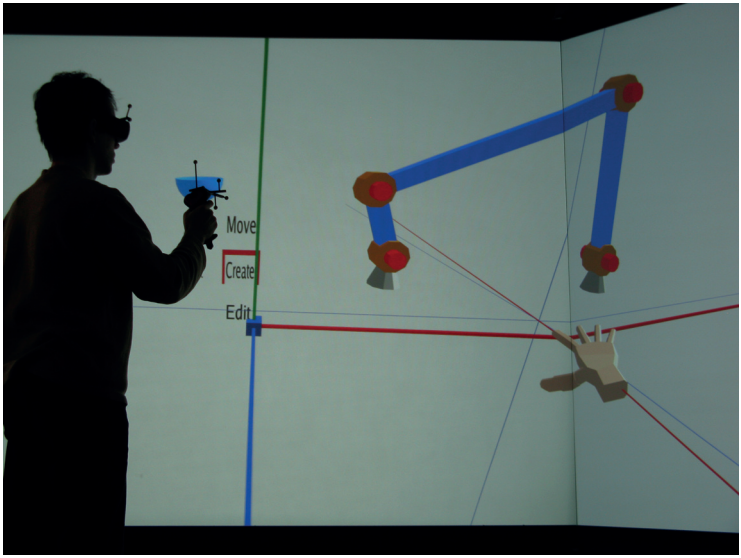


Abbildung 3.34: Modellierung eines Mechanismus mit der prototypischen Implementierung der *MASP*-CAD-VR-Kopplung in der *FASP*

der Lösung in Kombination mit umfangreichen Modellen. Deren Behandlung (Parameteränderung, Tesselierung, Übertragung und Import) ist rechenintensiv und könnte zu einer zeitlich verzögerten Rückmeldung der Auswirkungen von Interaktionen führen. Um diesem Problem zu begegnen, erscheint der Einsatz einer durchdachten Änderungsverfolgung zur Bestimmung der zu aktualisierenden Teilmodelle lohnenswert. Darüber hinaus könnte die Reaktion auf Interaktionen zweistufig gestaltet werden, um dem Anwender eine verzögerungsarme Rückmeldung zu geben. In einem ersten Schritt erfolgt eine vereinfachte Darstellung zu erwartender Änderungen anhand der Parametervisualisierung. Anschließend wird das geänderte CAD-Modell asynchron nachgeliefert, sobald dieses zur Verfügung steht.

Kapitel 4

Werkzeug für die integrierte und nutzerzentrierte Modellierung heterogener Systeme

Das vorliegende Kapitel verschiebt den Fokus der Arbeit von der phasenübergreifenden Modellierung mechanischer Systeme hin zu einem domänenübergreifenden Entwurf. Das Ziel besteht in der Erarbeitung und Umsetzung eines Konzeptes für ein Softwarewerkzeug, das den interaktiven Entwurf räumlicher heterogener Systeme in frühen Phasen der Produktentwicklung unterstützt. Die im vorangegangenen Kapitel betrachteten Konzepte zur phasenübergreifenden Modellbildung behalten ihre Gültigkeit, werden im Kontext der domänenübergreifenden Modellierung allerdings nicht weiter vertieft.

Das zu entwickelnde Werkzeug soll die Vorteile der in Kapitel 2 beschriebenen Ansätze zusammenführen. Hierzu soll einerseits die Produktstruktur im konstruktionstechnischen Sinne domänenübergreifend abgebildet werden, um den universellen Modelliermöglichkeiten allgemeiner Simulationssysteme nahe zu kommen und gleichzeitig eine simulative Abschätzung des Verhaltens des Gesamtsystems zu ermöglichen. Auf der anderen Seite wird die Absicht verfolgt, die einfache Zugänglichkeit der domänenspezifisch arbeitenden Werkzeuge durch eine anschauliche, auf die Begriffswelt des Ingenieurs ausgerichtete Modelldarstellung in Kombination mit einer unkomplizierten, leicht zu erschließenden Nutzerinteraktion zu erreichen. Auf diese Weise soll die Aufmerksamkeit des Nutzers auf das zu lösende Problem und weniger auf die eigentliche Modelliertätigkeit gelenkt werden.

Eine weitere Zielvorgabe besteht in der umfangreichen Erweiterbarkeit des Entwurfswerkzeugs, beispielsweise um die Unterstützung zusätzlicher Domänen oder neuer Berechnungsverfahren. Hierzu ist eine geeignete Softwarearchitektur zu entwickeln.

4.1 Ausgangslage und Vorgehen

Wie bereits im Fall der phasenübergreifenden Modellierung fanden auch die in diesem Kapitel beschriebenen Arbeiten ihren gedanklichen Ursprung in dem Entwurfswerkzeug *MASP*, wenngleich das Ergebnis lediglich in einem konzeptuellen Zusammenhang zu dieser Software steht. Den Ausgangspunkt der Überlegungen bildete der Wunsch nach einem Softwarewerkzeug, das in frühen Entwurfsphasen eine domänenübergreifende Modellierung unterstützt, diese eng mit Simulationsverfahren koppelt und dabei gleichzeitig einen nahezu intuitiven Zugang für den Ingenieur bietet. Mit *MASP* waren viele dieser Ideen bereits hervorragend für die interaktive Modellierung und Simulation ebener mechanischer Systeme umgesetzt. Der Gedanke, dass sich der dabei verwendete constraint-basierte Ansatz auch zur Behandlung anderer Problemstellungen eignet, führte anfangs zu der Idee, *MASP* zu einem domänenübergreifenden Werkzeug auszubauen. Zur Untersuchung entsprechender Konzepte entstand der Prototyp *MASP/H* (der Buchstabe H steht für „heterogene Systeme“).
Ein der Deutschen Forschungsgemeinschaft in diesem Zusammenhang vorgeschlagenes Vorhaben wurde mit Interesse aufgenommen. Eine Förderung war allerdings an die Bedingung geknüpft, dass das entstehende Werkzeug dreidimensionale Strukturen unterstützen sollte. Zusammen mit den Ergebnissen der Voruntersuchungen im Rahmen von *MASP/H*, welche die Grenzen von dessen Software-Architektur und des verwendeten geometrischen Constraint-Solvers aufzeigten, führte diese plausible Forderung zu der Entscheidung, einen neuen, weiter gefassten Ansatz zu wählen. Vorangegangene Erfahrungen zeigten, dass ein Entwurfswerkzeug mit den geforderten Eigenschaften nicht allein durch Erweiterung zu diesem Zeitpunkt verfügbarer Softwareprodukte zu realisieren war, die zudem vergleichbaren oder deutlicheren Einschränkungen als *MASP* unterlagen. Daher wurde ein neues Konzept für eine besser geeignete Entwurfs- und Simulationsplattform erarbeitet, das eine Reihe vorteilhafter Eigenschaften von *MASP* mit einschließt. Diese Plattform trägt die Bezeichnung *Hesym* (Heterogeneous Systems Modeler).
Die folgenden Abschnitte fassen die relevanten Schritte und Erkenntnisse auf dem Weg zu diesem Konzept zusammen, das anschließend detailliert dargelegt wird. Die Umsetzung des Konzeptes in Form eines Softwarepaketes ist in Kapitel 5 beschrieben.

4.2 Ansätze auf der Basis existierender Softwarewerkzeuge

Beim Entwurf heterogener Systeme stößt der Ingenieur oft an die Grenzen der eingesetzten Softwarewerkzeuge. Diese sind, wie in Kapitel 2 dargelegt, häufig für die Behandlung lediglich einer bestimmten Domäne ausgelegt oder wenig nutzerfreundlich. Demgegenüber beschränken sich Entwurfsgegenstände der Produktentwicklung selten auf eine einzelne Domäne. Dieser Widerspruch lieferte die Motivation für die Entwicklung eines leicht zu bedienenden, domänenübergreifend arbeitenden Entwurfswerkzeugs. Um den Aufwand für die Umsetzung eines solchen Vorhabens zu reduzieren, sollte ein bestehendes domänenspezifisches Softwarepaket den Ausgangspunkt darstellen und zu einem domänenübergreifenden Entwurfswerkzeug weiterentwickelt werden. Die Wahl fiel auf *MASP*, da es in der vorliegenden Fassung bereits über einige der Zieleigenschaften verfügte. Das Werkzeug versetzte den Anwender in die Lage, innerhalb kurzer Zeit ein Bewegungssystem zu modellieren und dessen (kinematisches) Verhalten zu simulieren. Eine besondere Herausforderung bestand bei diesem anfänglich gewählten Ansatz in der Nachnutzung eines existierenden Konzeptes und der daraus resultierenden Softwarearchitektur, die ursprünglich für ein anderes, weniger weit gefasstes Einsatzgebiet ausgelegt waren. In den folgenden Unterabschnitten werden die Architektur des erweiterten Werkzeugs sowie die bei der Umsetzung gewonnenen Erfahrungen wiedergegeben.

4.2.1 Konzept einer integrierten Lösung mit homogenem Modell und homogener Simulationsumgebung

Eine wesentliche Voraussetzung für die gesamtheitliche Betrachtung eines technischen Systems ist ein Produktmodell, das die relevanten Informationen und Bestandteile aller beteiligten Domänen abbildet. In der Konzeptphase (z. B. auf der Ebene des technischen Prinzips) bietet sich eine strukturorientierte Modellbildung an, die Modellelemente explizit in Beziehung zueinander setzt. Bei domänenspezifischen Werkzeugen leiten sich das Modell und die dafür benötigten Datenstrukturen zumeist aus den Vorgaben eines bestimmten Berechnungsverfahrens oder Simulators ab. Beispiele hierfür sind Entwurfswerkzeuge für elektronische Schaltungen, deren Modelle als Netze von Bauelementen abgelegt sind oder Simulationswerkzeuge der numerischen Strömungsmechanik, die ihre Modelle als in finite Elemente unterteilte Volumen strukturieren. Sollen Komponenten anderer Domänen in solchen Werkzeugen behandelt werden, ist die Formulierung von Ersatzmodellen im Rahmen der vorgegebenen Berechnungsverfahren notwendig.

So ist im Elektroniksimulator SPICE beispielsweise die Behandlung von Masse-Feder-Dämpfer-Systemen mittels einer Analogiebetrachtung zu L-C-R-Gliedern möglich (z. B. [KP06]).
Das Entwurfswerkzeug *MASP*, das ursprünglich für den Entwurf von Bewegungssystemen ausgelegt war, baut seine Modelle als Constraint-Netz eines geometrischen Constraint-Solvers auf (ausführlich beschrieben in [Dör11]). Bauelemente anderer Domänen können in diese Software genau dann integriert werden, wenn mit den Mitteln des Constraint-Solvers ein geometrisches Ersatzmodell gebildet werden kann. In einem erweiterten Prototyp von *MASP* (*MASP/H*) werden beispielsweise optische Systemkomponenten unter Verwendung von Gesetzen der geometrischen Optik abgebildet.
Die Bildung solcher Ersatzmodelle muss nicht zwangsläufig dem Nutzer überlassen werden, sondern kann durchaus automatisch erfolgen. Ohnehin entspricht die Formulierung berechnungsgeeigneter Modelle, z. B. als Gleichungssystem oder Graph nicht der gewohnten Begriffswelt und Arbeitsweise des Konstrukteurs und sollte daher nicht zu seinem Aufgabenbereich gehören. Es bietet sich daher die Trennung von Nutzersicht und Berechnungssicht an, die technisch in Form zweier paralleler Modelle realisiert wird.
Diese Trennung ist auch in *MASP* umgesetzt. So besteht hier das Berechnungsmodell aus einem Graphen von geometrischen Primitiven und Beziehungen zwischen diesen. Das nutzerorientierte Modell umfasst hingegen symbolhaft dargestellte Prinzipelemente, die Ausschnitte des Berechnungsmodells mit einer getriebetechnischen Semantik versehen. Das Konzept setzt außerdem eine automatische Transformation zwischen beiden Modellsichten voraus, so dass die Bildung des Berechnungsmodells für den Anwender hinter dem nutzerorientierten Modell verborgen bleibt (Abbildung 4.1). Dieser in *MASP* für mechanische Problemstellungen umgesetzte Ansatz kann auf viele andere Domänen übertragen werden.

4.2.2 Erfahrungen mit dem Prototyp MASP/H

Im Rahmen dieser Arbeit entstand der Prototyp *MASP/H*, mit dem Erkenntnisse über die Eignung der Constraint-Technik zur Modellierung heterogener Systeme gewonnen werden sollten. *MASP* wurde beispielhaft um Modellelemente anderer Domänen erweitert. Für jedes dieser Modellelemente wurde eine constraint-basierte Repräsentation erstellt. Wie das Beispiel der Bewegungssysteme zeigt, eignet sich diese Form der Beschreibung hervorragend für die Modellierung von Systemen, deren funktionsrelevante Merkmale überwiegend mit geometrischen Mitteln abbildbar sind. Gleichzeitig ermöglicht die constraint-basierte Modellierung eine anschauliche Präsentation des Entwurfsgegenstandes einschließlich seiner räumlichen Merkmale wie Maße und Anordnungsrelationen.

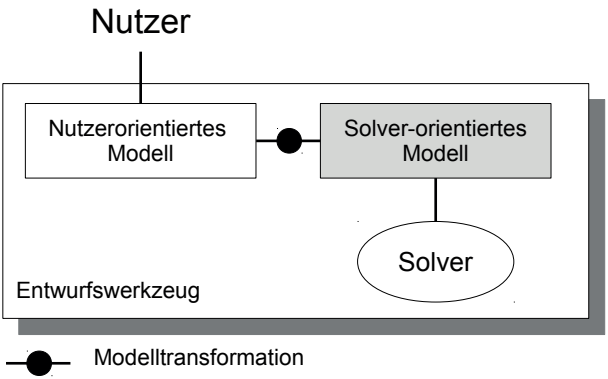


Abbildung 4.1: Aufbau eines Entwurfswerkzeugs als integrierte, solverspezifische Lösung

Erwartungsgemäß gelang in *MASP/H* die Modellierung optischer Komponenten ebenfalls problemlos. Hierzu wurden Zusammenhänge der geometrischen Optik auf der Basis geometrischer Primitive und Beziehungen beschrieben. Abbildung 4.2 illustriert diese Vorgehensweise am Beispiel der Umsetzung von Brechungs- und Reflexionsgesetz.

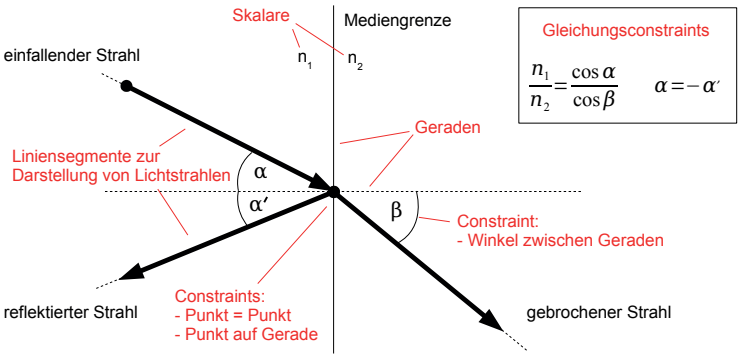


Abbildung 4.2: Beispiel für Verfahren der geometrischen Optik, abgebildet mit geometrischen Primitiven und Constraints

Darüber hinaus erläutert Abschnitt 6.1 die Modellierung eines optomechanischen Systems auf der Grundlage geometrischer Constraints in *MASP/H* ausführlich.

Hervorzuheben ist die dabei entstandene, nahtlose Integration mechanischer und optischer Teilsysteme in einem Simulationsmodell. Dieses erlaubt beispielsweise die direkte Berechnung optischer Eigenschaften (Strahlengang, Brechungsindex eines optischen Mediums) anhand der Vorgaben mechanischer Komponenten und umgekehrt. Gegenüber der weithin verbreiteten getrennten Modellierung und Simulation der Teilsysteme stellt eine solche ganzheitliche Betrachtung eine deutliche Erleichterung während des Entwurfsvorgangs dar [RBD07].
Probleme bereitete hingegen die Modellierung regelungstechnischer Systeme. Hier zeigte sich, dass die Behandlung von Komponenten mit integrierenden oder differenzierenden Eigenschaften mit den Beschreibungsmitteln des geometrischen Constraint-Solvers nicht oder nur umständlich möglich war. Eine mögliche Lösung besteht im Zurückgreifen auf alternative Berechnungsverfahren für diese Komponenten, die allerdings die ganzheitliche Behandlung des Gesamtsystems mit nur einem einzigen Solver in *MASP/H* verhindert.
Gleichzeitig zeigte der Versuch, zusätzliche Berechnungsverfahren zu integrieren, die Grenzen der Erweiterbarkeit von *MASP/H* auf. Softwaretechnisch ist *MASP/H* so aufgebaut, dass die Datenstrukturen der Modellelemente auf den Schnittstellen und Datentypen des geometrischen Constraint-Solvers aufsetzen und somit einzig für die Verarbeitung mit diesem ausgelegt sind. Andere Berechnungsverfahren benötigen Datenstrukturen mit den für sie definierten Schnittstellen. Grundsätzlich ist es möglich, solche Strukturen in die vorhandenen Modellelemente von *MASP/H* zu integrieren. Diese erweiterten Modellelemente stellen allerdings Ansammlungen unterschiedlicher mathematischer Modelle dar, die keinen gemeinsamen Zustand besitzen und für deren Synchronisierung keine Infrastruktur existiert.
Weiterhin muss für so vorgenommene Erweiterungen der Quellcode des Entwurfswerkzeugs als Ganzes angepasst werden, was einem wichtigen Grundsatz moderner Softwareentwicklung widerspricht und die Koordination parallel arbeitender Entwickler erschwert. Hier bietet sich unter anderem die Verwendung einer Plug-In-Schnittstelle an, die insbesondere vor dem Hintergrund eines häufig zu erweiternden Softwareproduktes große Vorteile mit sich bringt (siehe auch Abschnitt 4.8.5).
Zusammenfassend stellt *MASP/H* ein interaktives Baukastensystem mit integrierten Simulationsmöglichkeiten dar, das dank anschaulicher Darstellung einen schnellen und unkomplizierten Entwurf von Prinziplösungen ermöglicht. Gleichzeitig verdeutlicht *MASP/H* die Grenzen der Erweiterbarkeit solcher domänenspezifischer, integrierter Entwurfswerkzeuge. Diese werden vom eingesetzten Solver und der Formulierbarkeit entsprechender Ersatzmodelle gesetzt. Die oben beschriebenen Erweiterungsversuche zeigen, dass eine Ausweitung der Modellunterstützung auf beliebige Domänen in *MASP/H* als schwierig einzustufen ist.

4.2.3 Weitere Ansätze auf der Grundlage existierender Werkzeuge

Aus den Untersuchungen der vorangegangenen Abschnitte geht hervor, dass domänenspezifische Werkzeuge keine gut geeignete Basis für die Entwicklung eines allgemein verwendbaren, domänenübergreifenden Entwurfswerkzeugs darstellen. Eine mögliche Alternative bildet der Einsatz allgemeiner Simulationsumgebungen (z. B. *Simulink*, *Modelica*, siehe Abschnitt 2.2.3), die gewöhnlich keinen größeren Einschränkungen bezüglich der Modellierung von Systemen beliebiger Anwendungsdomänen unterliegen.
Aufgrund ihrer universellen Natur greifen allgemeine Simulationswerkzeuge meist auf sehr abstrakte, wenig spezialisierte Modellvisualisierungen, beispielsweise in Form von Blockdiagrammen, zurück. Diese vermitteln dem Ingenieur zwar einen Eindruck von der Systemstruktur, weisen aber Defizite bei der Wiedergabe anderer Systemeigenschaften wie räumlicher Zusammenhänge auf. Gleichwohl es auch hier Bestrebungen gibt, domänenspezifische Darstellungen in diese Werkzeuge zu integrieren, visualisieren diese nur einzelne Aspekte oder Domänen aus dem Gesamtmodell als „Inseln“. Zusammenhänge zu anderen Teilmodellen und damit eine anschauliche Darstellung des Gesamtmodells gehen auf diese Weise verloren. Darüber hinaus erlauben diese Visualisierungen keine Interaktion mit dem Modell. Ein direkter Zugriff auf Systemparameter und deren Beeinflussung anhand des sichtbaren Modells ist in der Regel nicht möglich (Beispiele in [Fri06] und Abbildung 4.3).

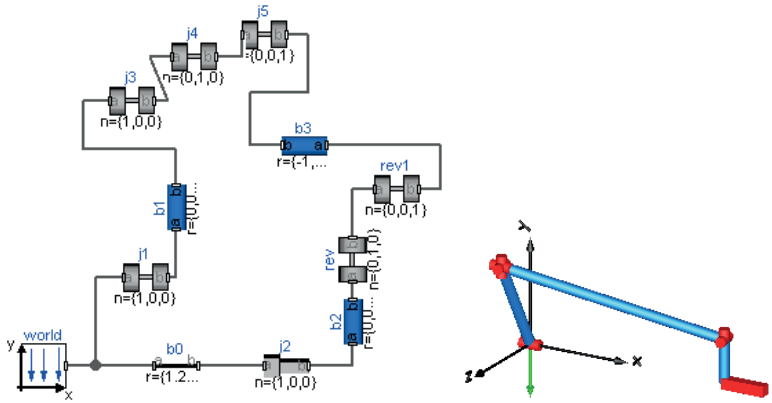


Abbildung 4.3: Modell einer Viergelenkkette in *Modelica* als Blockdiagramm (links) und als 3D-Visualisierung (rechts). Die anschaulichere 3D-Darstellung erlaubt keine interaktive Modellierung. Quellen: [Mod13] und [Ass08].

Ein anderer Ansatz zur domänenübergreifenden Modellierung besteht in der Nutzung unterschiedlicher, domänenspezifischer Softwarewerkzeuge, deren Zusammenwirken von einer zentralen Koordinationsinstanz gesteuert wird. Hierzu existieren bereits seit langem Ideen (z. B. [SKW92], [SAKJ98]), die unter dem Gesichtspunkt der derzeitigen hard- und softwaretechnischen Möglichkeiten und neuer Nutzerbedürfnisse analysiert und angepasst werden müssen. Die beteiligten Softwarepakete bieten meist einen guten Zugang für den Anwender, zeigen allerdings Nachteile beim Datenaustausch untereinander. Da die domänenspezifischen Teilmodelle bei solchen Leitsystemen auf unterschiedliche Werkzeuge verteilt sind und eine Laufzeitkopplung nicht oder nur unzureichend unterstützt wird, erschwert sich die Simulation des Gesamtsystems erheblich. Ohne einheitliche Schnittstellen zur Kopplung und Koordination von Werkzeugen, deren Realisierung an den konkurrierenden Softwareanbietern scheitern dürfte, erweist sich auch dieser Ansatz als nur eingeschränkt geeignet.

4.3 Konzept einer Software-Plattform für Entwurf und Simulation heterogener Systeme

Das neu zu entwickelnde Werkzeug *Hesym* realisiert eine Mischform aus den oben beschriebenen Konzepten und versucht, deren Vorteile zu kombinieren. Es handelt sich dabei um die Idee einer Software-Plattform für Modellierung und Simulation, in deren Mittelpunkt ein domänenübergreifendes Gesamtmodell steht, welches dem Anwender zusammenhängend und in domänenspezifischen Begriffen und Darstellungsweisen zugänglich gemacht wird. Dabei umfasst das Werkzeug keinen vordefinierten Satz an Anwendungsdomänen und Simulationsverfahren, sondern stellt eine Infrastruktur zur Verfügung, die eine problemangepasste Erweiterung jederzeit möglich macht.
Der vorliegende Abschnitt beleuchtet diese Idee zunächst aus Sicht des Anwenders, schlägt anschließend eine Systemarchitektur vor und und stellt schließlich eine Liste von Kernaufgaben zusammen, die bei der detaillierten Ausarbeitung eines solchen Konzeptes gelöst werden müssen.

4.3.1 Vorteilhafte Merkmale domänenspezifischer Entwurfs- und Simulationswerkzeuge aus Sicht des Anwenders

Am Beispiel der Kinematiksoftware *MASP*, die stellvertretend für die Gruppe domänenspezifischer Werkzeuge steht, lassen sich Teilkonzepte zur Modellbildung, Simulation und Nutzerinteraktion identifizieren, auf deren Zusammenwirken die

einfache Bedienung der Software zurückzuführen ist (siehe auch die Usability-Untersuchung von *MASP* in [KF06]). Es folgt ein Überblick über diese Konzepte, da sie aus Anwendersicht einen erheblichen Einfluss auf die Konzipierung nahezu aller Aspekte eines Entwurfswerkzeugs für heterogene Systeme ausüben.

Zweckmäßiges Abstraktionsniveau

In *MASP* werden Bewegungssysteme auf der Ebene des technischen Prinzips beschrieben. Das Modell beschränkt sich auf die wichtigsten funktionsrelevanten Eigenschaften. Diese lassen sich einerseits mit geringem manuellem Aufwand eingeben und sind andererseits hinreichend vollständig für eine simulative Abschätzung des kinematischen Verhaltens.

Baukastenprinzip

MASP stellt einen definierten Satz von Bauelementen und Symbolen zur Verfügung, mit deren Hilfe eine systematische Beschreibung von Mechanismen vorgenommen wird. Dies führt zu einer gewissen Einschränkung der Flexibilität (im Vergleich z. B. zur Modellierung beliebiger Formen in einem CAD-System). Gleichzeitig verringert sich aber der Aufwand bei der interaktiven Modellierung, da im Unterschied zu einer rein geometrischen Betrachtung die Semantik der Bauelemente einbezogen wird, so dass eine automatische Ableitung bestimmter, für den Modellaufbau benötigter Informationen möglich wird. Weiterhin bedeutet das Vorhandensein konkreter Bauelemente anstelle bloßer geometrischer Primitive und Constraints eine hervorragende Führung des Nutzers bei der Formulierung einer Lösung.

Anschauliche Darstellung und abgestimmtes Interaktionskonzept

Die grafische Darstellung des technischen Prinzips in *MASP* orientiert sich an einer aus getriebetechnischer Fachliteratur (z. B. [Lic65], [Hai61]) bekannten, leicht verständlichen Symbolik (Abbildung 4.4). Über diese wird gleichzeitig der Zugriff auf die Parameter der Modellelemente abgewickelt (z. B. per Kontextmenü). Weiterhin steht eine Reihe zweckmäßiger Modellieroperationen zur Verfügung, die im Zusammenhang mit einer durchdachten Maus- und Tastaturkoordination ein zügiges Arbeiten ermöglichen.

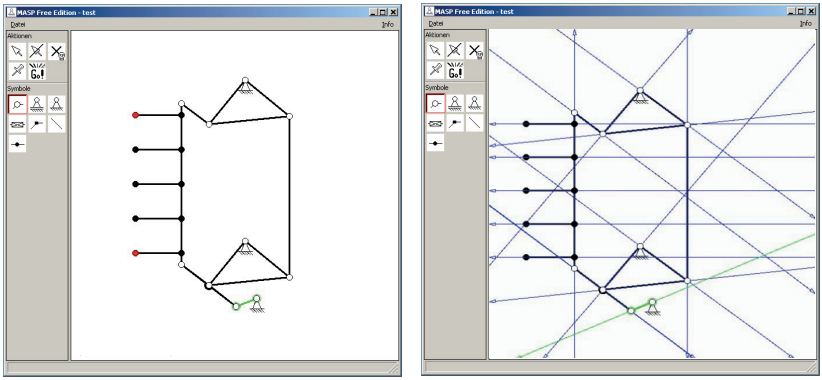


Abbildung 4.4: Entwurfswerkzeug *MASP* mit Benutzeroberfläche und visualisierter Constraint-Darstellung des Modells (rechts)

Transformation zwischen nutzerorientierten und berechnungsgeeigneten Modellen

In *MASP* existiert neben der nutzerorientierten, symbolbasierten Beschreibung des Entwurfsobjektes eine berechnungsgeeignete, constraint-basierte Repräsentation, die für die Bewegungssimulation besonders gut geeignet, für den Nutzer aber schwer verständlich ist. Um diese Modelldarstellung zu verbergen, findet eine automatische Transformation von und zur nutzerorientierten Beschreibung statt (Abbildung 4.4). Auf diese Weise arbeitet der Anwender ausschließlich mit der ihm vertrauten Symbolik.

4.3.2 Architektur der Plattform für Entwurf und Simulation

Eines der größten Hemmnisse bezüglich der Behandlung unterschiedlicher Domänen bei der in Abschnitt 4.2.1 beschriebenen Architektur ist die Verwendung eines Modells, das allein mit den Datenstrukturen des integrierten Solvers beschrieben wird. Daher sieht das Plattformkonzept neben den berechnungs- und nutzerorientierten Modellsichten die Verwendung eines domänenübergreifenden, solverunabhängigen Modells vor (Abbildung 4.5). Die so entstehende, zentrale Beschreibung des technischen Systems orientiert sich nicht mehr primär an den Erfordernissen (Datenstrukturen) eines bestimmten Berechnungsverfahrens oder Constraint-Solvers und öffnet das Werkzeug somit für die Verwendung unterschiedlicher Simu-

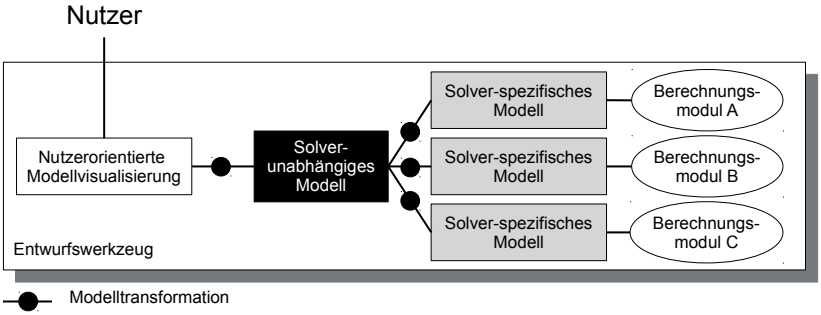


Abbildung 4.5: Entwurfswerkzeug als Plattform für Berechnungsmodule

lationsverfahren. Die Begriffe „solverunabhängig“ und „solverspezifisch“ werden hier gewählt, um sperrige Bezeichnungen wie „berechnungsverfahrensunabhängig“ zu vermeiden, selbst wenn definitionsabhängig nicht jedes Berechnungsverfahren einen Solver darstellt.

Da die Auswertung des domänenübergreifenden Gesamtmodells mittels eines einzelnen Solvers als schwierig einzustufen ist, werden aufgabenspezifische Berechnungsmodule zur Behandlung von Teilmodellen herangezogen. Zu diesem Zweck existiert eine Konvertierungsschicht, die in den solverunabhängigen Modellelementen abgelegte Semantiken, Parameter und Beziehungen interpretiert und eine Transformation in eine solverspezifische Repräsentation vornimmt.

Das solverunabhängige, domänenübergreifende Modell stellt gleichzeitig ein Medium zur Synchronisierung des Datenbestandes der Berechnungsmodule dar, so dass nur eine begrenzte Anzahl von Konvertern implementiert werden muss. Ein weiterer Vorteil ist, dass sowohl der Modellbeschreibungskern als auch die Visualisierungs- und Konvertierungsschichten mittels Plug-Ins erweiterbar sind und somit jederzeit weitere Lösungselemente und Berechnungsverfahren, auch aus zusätzlichen Domänen, in die Software-Plattform einbezogen werden können.

Während die Erweiterung des solverunabhängigen Modells kaum Probleme bereitet, ist die Einbindung von Solvern und Berechnungsverfahren an die Bedingung gebunden, dass diese entweder als Programmbibliothek vorliegen, über eine entsprechende Kommunikationsschnittstelle verfügen oder selbst implementiert werden können. Auch die Koordination der Berechnungsmodule bei gleichzeitigem Einsatz bedeutet zusätzlichen Aufwand.

4.3.3 Anforderungen und Arbeitsschwerpunkte bei der Konzipierung eines Softwarewerkzeugs für den Entwurf heterogener Systeme

Bei der Entwicklung eines Entwurfswerkzeugs für heterogene Systeme sind unterschiedliche Aspekte zu berücksichtigen, die in gegenseitiger Wechselwirkung stehen und daher in ihrer Gesamtheit für die Entstehung des Werkzeugs von Bedeutung sind. In den vorangegangenen Abschnitten wurden Anforderungen und Kernprobleme aus verschiedenen Blickwinkeln aufgezeigt, aus denen sich zusammenfassend die folgenden Arbeitsschwerpunkte ableiten lassen.

Domänenübergreifende Modellbildung

Für die betrachteten Domänen soll ein geeignetes Baukastensystem entstehen, d.h. es werden Komponenten gesucht, die einen Kompromiss aus Flexibilität und Vereinfachung des Modelliervorgangs darstellen. Gleichzeitig stellt sich die Frage, welche Informationsmenge zur Beschreibung der Eigenschaften der Modellelemente aus den einzelnen Domänen notwendig ist. Weiterhin ist ein domänenübergreifendes, von Berechnungsmodulen unabhängiges Datenmodell zur rechnerinternen Repräsentation der ermittelten Modellelemente zu konzipieren.

Integration von Berechnungsmodulen

Berechnungsmodule unterscheiden sich zum Teil stark in ihren Anforderungen an Eingabedaten, im Format ihrer Ergebnisrepräsentation und in ihrer Arbeitsweise. Die Schaffung einer vereinheitlichenden Schnittstelle (Zeitkonzept, Konsistenzbegriff, Struktur-/Zustandsänderungen usw.) ist daher für den parallelen Einsatz unterschiedlicher Berechnungsverfahren erforderlich. Ein weiterer Schwerpunkt liegt in der verlustarmen Transformation zwischen berechnungsmodulunabhängiger und berechnungsmodulspezifischer Modelldarstellung.

Softwarearchitektur

Die Heterogenität der zu modellierenden Systeme stellt besondere Anforderungen an die Softwarearchitektur des Entwurfswerkzeugs. Es ist nicht anzunehmen, dass dieses einen von Anfang an bekannten, klar eingegrenzten Funktionsumfang hat. Vielmehr wird eine ständige Erweiterung zusätzliche Funktionalität (weitere Domänen mit neuen Modellelementen, Berechnungs- und Visualisierungsverfahren usw.) vorzunehmen sein. Daher sollen Erweiterungsschnittstellen für nahezu jeden Aspekt als Kernbestandteil des Konzepts für die notwendige Flexibilität sorgen.

Für das Entwurfswerkzeug als grafisch-interaktives System bietet sich ein Aufbau in Anlehnung an Architekturmuster wie *Model-View-Controller* [Ree79] an.

Integration in den Entwurfsprozess und die Werkzeugkette

Das Entwurfswerkzeug konzentriert sich auf Aspekte früher Phasen. Um eine Insellösung zu vermeiden, empfiehlt sich die Kopplung mit Werkzeugen für die Unterstützung der nachfolgenden Phasen (insbesondere Gestaltentwurf mit CAD-System). Dabei ist zwischen einem einfachen Export funktionsrelevanter Parameter und einer bidirektionalen Kopplung zu unterscheiden.

Räumliche Interaktion

Die auch in *MASP* umgesetzte Idee, Bewegungssysteme bereits mittels weniger Nutzereingaben beschreiben zu können, soll auch für die Modellierung heterogener Systeme zur Anwendung kommen. Die Umsetzung erweist sich als aufwendig, da im Gegensatz zu den ebenen Interaktionen in *MASP* eine räumliche Interpretation zweidimensionaler Eingaben sowie eine anschauliche und eindeutige Präsentation dreidimensionaler Modelle erreicht werden muss.

Abstimmung

Jeder der aufgeführten Punkte stellt Anforderungen an die Gestaltung des Gesamtsystems. Umgekehrt muss jede Komponente solche Anforderungen berücksichtigen. Die Entwicklung des Entwurfswerkzeugs kann sich also nicht auf einzelne Teilgebiete beschränken, sondern sollte auf einer ganzheitlichen Betrachtung aufbauen.

Mit dem folgenden Abschnitt beginnt die detaillierte Betrachtung der hier aufgeführten Schwerpunkte, während das darauffolgende Kapitel auf die Umsetzung des Gesamtkonzepts als Prototyp eingeht.

4.4 Modellierung heterogener Systeme

Dieser Abschnitt erörtert Anforderungen und Möglichkeiten zur rechnerinternen Repräsentation technischer Systeme in frühen Phasen der Produktentwicklung und beschreibt den für das Entwurfswerkzeug *Hesym* gewählten Ansatz.

4.4.1 Begriff „Heterogenes System“

Der Begriff „Heterogenes System“ beschreibt im Rahmen dieser Arbeit ein technisches System, das Komponenten unterschiedlicher technischer Anwendungsdomänen umfasst. Für das Konzept und die prototypische Implementierung des Werkzeugs sollen beispielhaft die Bereiche Mechanik, Optik und Regelungstechnik betrachtet werden, ohne jedoch andere Domänen explizit auszuschließen.

4.4.2 Anforderungen und Problemstellungen

Für das Modell zur Abbildung des Entwurfsgegenstands lassen sich eine Reihe zentraler Ziele formulieren. Das Modell sollte die Beschreibung der Komponenten der beteiligten Domänen ermöglichen und diese zu einer gesamtheitlichen Darstellung des Entwurfsgegenstandes zusammenführen. Darüber hinaus muss es sowohl für den Entwurf als auch für die Simulation geeignet sein. Weiterhin fordert das Gesamtkonzept des Entwurfswerkzeugs erweiterbare Mittel zur Beschreibung des Modells, beispielsweise um zusätzliche Domänen integrieren zu können.
Als Abstraktionsniveau für das Modell bieten sich aus konstruktionstechnischer Sicht sowohl die Funktionsstruktur als auch das technische Prinzip an. Beide sind gebräuchliche Darstellungsformen der frühen Phasen. Sie umfassen eine Informationsmenge, die einerseits hinreichend kompakt ist, um sie rechnergestützt mittels weniger Interaktionsschritte formulieren zu können und die sich andererseits in ihrer Aussagekraft für die simulative Abschätzung grundlegender Produkteigenschaften eignet.
Es erscheint jedoch nicht zweckmäßig, einer der beiden Abstraktionsebenen als alleinige Modellierungsgrundlage den Vorzug zu geben, da beide Darstellungsformen in einzelnen Anwendungsdomänen ein unterschiedliches Gewicht besitzen oder die verwendeten domänenspezifischen Darstellungsformen sich nicht in die Sichtweise des KEP einordnen. Beispielsweise verfügen die Beschreibungen elektronischer Schaltungen oder regelungstechnischer Systeme verbreitet über Merkmale einer Funktionsstruktur, während optische Schemata eine größere Nähe zum technischen Prinzip aufweisen.

Vor diesem Hintergrund wird vorgeschlagen, eine Vermischung oder Kombination der Darstellungsformen beider Abstraktionsniveaus im Gesamtmodell zuzulassen. Dies könnte aus methodischer Sicht als Aufweichung des Entwicklungsprozesses nach VDI 2221 aufgefasst werden. Der Ansatz bietet jedoch eine Reihe von Vorteilen in der Praxis. So wird es in vielen Fällen überhaupt erst möglich, die Simulation eines Gesamtsystems vorzunehmen, das sich aus Teilsystemen zusammensetzt, die in unterschiedlichen Abstraktionsebenen vorliegen. Weiterhin entfällt die Notwendigkeit, die Modellierung der Teilsysteme auf ein bestimmtes Abstraktionsniveau zu normieren, was für den Nutzer eine ungewohnte, da unübliche Darstellung zur Folge hätte.
Eine andere Frage beschäftigt sich mit der mathematischen Repräsentation des modellierten Objektes. Aus Sicht des Entwurfs ist die logische Struktur des Systems von besonderer Bedeutung. Eine Unterteilung des Systems in parametrisierte, semantikbehaftete Komponenten, wie sie etwa von den Symbolen des technischen Prinzips verkörpert werden, erscheint daher zweckmäßig. Demgegenüber orientieren sich die Repräsentationen vieler Simulationsverfahren weniger an der Produktstruktur, sondern vielmehr an den ihnen zugrunde liegenden mathematischen Zusammenhängen. Diese unterscheiden sich teilweise deutlich. Dieser Umstand erschwert die Auswahl einer bestimmten mathematischen Darstellung für das Gesamtmodell. Hinzu kommt, dass im Sinne der Erweiterbarkeit beliebige domänenspezifische Berechnungsmethoden integriert werden sollen, so dass es kaum möglich scheint, die Eigenheiten aller erdenklichen Verfahren in einer einzigen Repräsentation zu berücksichtigen.

4.4.3 Domänenübergreifendes Gesamtmodell

Um den oben beschriebenen Widerspruch zu lösen, sieht das Konzept des Entwurfswerkzeugs die Verwendung unterschiedlicher Modellierebenen vor (siehe auch [RB09, RDB07]). Bei der ersten Ebene handelt es sich um ein domänenübergreifendes Gesamtmodell. Dieses beschreibt die Produktstruktur als Graph von Komponenten und deren Beziehungen untereinander. Dabei vermeidet es die Abbildung des Entwurfsobjektes in simulations- und berechnungsspezifischen Repräsentationsformen, die sich lediglich für die Modellierung bestimmter Anwendungsfälle eignen. Auf diese Weise ist es möglich, ein Modell zu erstellen, das ein heterogenes System über alle beteiligten Domänen hinweg dokumentiert (Abbildung 4.6) und das aufgrund seines geringen Spezialisierungsgrades jederzeit die Definition neuer Modellelemente (Komponenten) erlaubt. Beispiele für Modellelemente aus unterschiedlichen Domänen und in ihnen enthaltene Informationen sind in Tabelle 4.1 aufgeführt.

Tabelle 4.1: Beispiele für Modellelemente unterschiedlicher Anwendungsdomänen

Domäne	Modellelemente
Mechanik	Getriebeglied, Drehgelenk, Kurvenscheibe
Optik	optische Linse, Spiegel, Prisma, Lichtquelle
Automatisierungstechnik	Sensor, Regler, Aktor

Die zweite Ebene der Modellierung besteht aus einer Reihe solverspezifischer Teilmodelle, die bestimmten Aspekten des domänenübergreifenden Gesamtmodells zugeordnet sind und aufgabenabhängig aus diesem erzeugt werden. Die solverspezifischen Modelle bleiben mit dem Gesamtmodell bidirektional verknüpft, so dass ihre stetige Aktualisierung und eine Rückführung von Berechnungsergebnissen in das Gesamtmodell möglich ist. Der folgende Abschnitt beschreibt eine Infrastruktur zur Erzeugung der solverspezifischen Modelle und ihrer Kopplung mit dem Gesamtmodell.

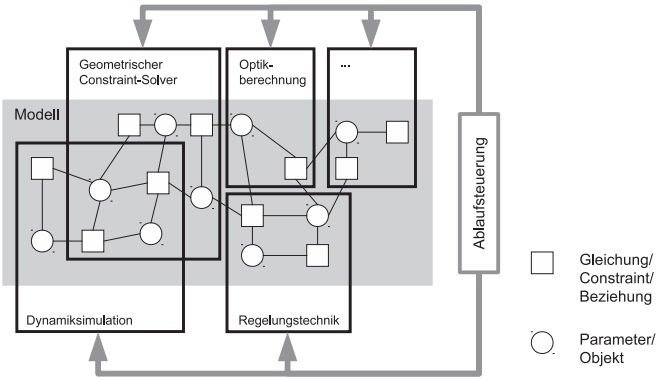


Abbildung 4.6: Domänenübergreifendes Gesamtmodell als Graph und Zuordnung von Teilmodellen zu Berechnungsmodulen

4.4.4 Modellelemente und Transformationselemente

Modellelemente bilden die Grundbausteine des domänenübergreifenden Gesamtmodells. Sie umfassen ihre Semantik, funktionsrelevante Parameter und Beziehungen zu anderen Modellelementen. Die Beschreibung jedes Modellelements besteht

aus einfachen mathematischen und informationstechnischen Konstrukten wie z. B. Skalaren, Vektoren und Referenzen oder Listen aus diesen. Zusammen mit semantischen Bezeichnern erhalten sie im Kontext der Domänen ihre Bedeutung. In Tabelle 4.2 sind Beispiele für Modellelemente und die zur Beschreibung notwendigen Informationen aufgeführt.

Tabelle 4.2: Beispiele für Modellelemente und enthaltene Informationen

Modellelement	enthaltene Informationen
Drehgelenk	Position (Punkt), Rotationsachse (Vektor), verbundene Getriebeglieder/Körper (Referenzen)
Antrieb	Position (Punkt), Kraftrichtung (Vektor), angetriebenes Element (Referenz, über Koppelstelle), Ansteuerung (Skalar)
Optisches Prisma	Position (Punkt), Orientierung (Vektoren), Mediengrenzen (Liste begrenzter Flächen), Brechungsindex (Skalar), Kopplung an andere Elemente (Referenz)

Das domänenübergreifende Modell und seine Elemente sind unabhängig von den Datenstrukturen formuliert, die von den unterschiedlichen Berechnungsverfahren als Eingabedaten gefordert werden. Somit ist keine direkte Verarbeitung des Gesamtmodells durch diese Verfahren möglich. Zur Simulation des Systemverhaltens ist daher eine Transformation der solverunabhängigen Repräsentation in die vom jeweiligen Berechnungsverfahren verwendete Darstellung notwendig. Das Entwurfswerkzeug sieht zu diesem Zweck Transformationselemente vor, die aus einzelnen oder mehreren Modellelementen verfahrensspezifische Modellbestandteile generieren und als Brücke für einen bidirektionalen Informationsaustausch zwischen Gesamtmodell und Simulationsmodellen fungieren (Abbildung 4.7). Im Simulationsmodell entstehende Zustandsänderungen werden, soweit relevant, in das Gesamtmodell zurückgeführt.
Zu den Vorteilen solcher Transformationselemente gehört, dass sie unabhängig voneinander für unterschiedlichste Berechnungsmodule implementiert werden können und somit ein einmal definiertes (solverunabhängiges) Modellelement für verschiedene mathematische Verfahren zur Verfügung steht. Hinzu kommt, dass die Berechnungsmodule ihren Informationen mittels des solverunabhängigen Gesamtmodells synchronisieren und somit auf einem einheitlichen Datenbestand arbeiten.

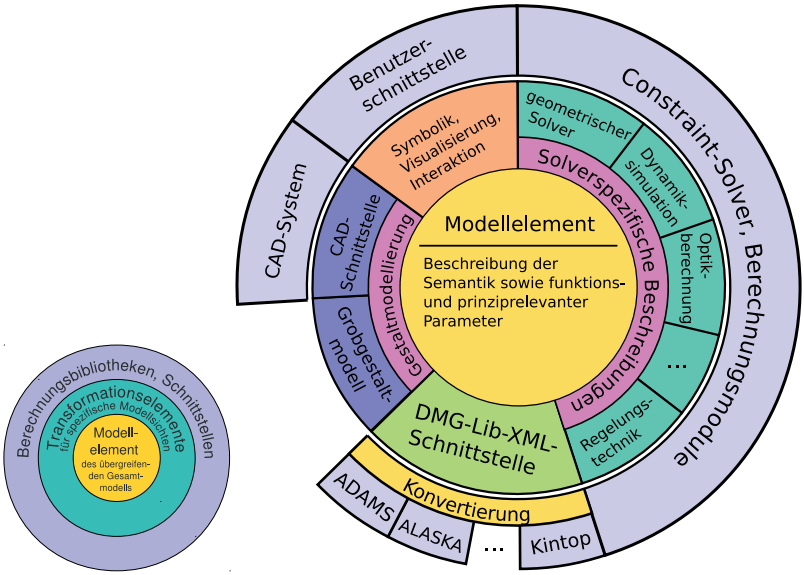


Abbildung 4.7: Modellelemente und Transformationselemente. Grundaufbau (links) und Anwendungsbeispiel (rechts).

Das Einsatzgebiet von Transformationselementen lässt sich auf andere Aspekte wie Visualisierung und Interaktion oder die Kopplung mit CAD-Systemen ausdehnen. In diesen Fällen existieren transformierte Versionen der Modellelemente z. B. als grafische Darstellung für den Nutzer oder als Zuordnungselement für Gestaltmerkmale (siehe Abschnitt 3.4.3).
Diese Vorgehensweise reduziert die Anzahl der notwendigen Modellkonvertierungen und sorgt für die Unabhängigkeit der Softwarekomponenten untereinander, da nicht zwischen allen aufgabenspezifischen Modellsichten Transformationen durchgeführt werden müssen (Abbildung 4.8).

4.4.5 Beispiel für ein Modellelement mit Transformationselementen

Abbildung 4.9 zeigt die Verwendung von Transformationselementen am Beispiel eines binären Getriebegliedes. Das Modellelement speichert als Datenstruktur ein Referenzsystem, zwei Ankerpunkte und einen Längenparameter. Zugeordnet sind zwei Transformationselemente zur Visualisierung und zwei zur Simulation. Die

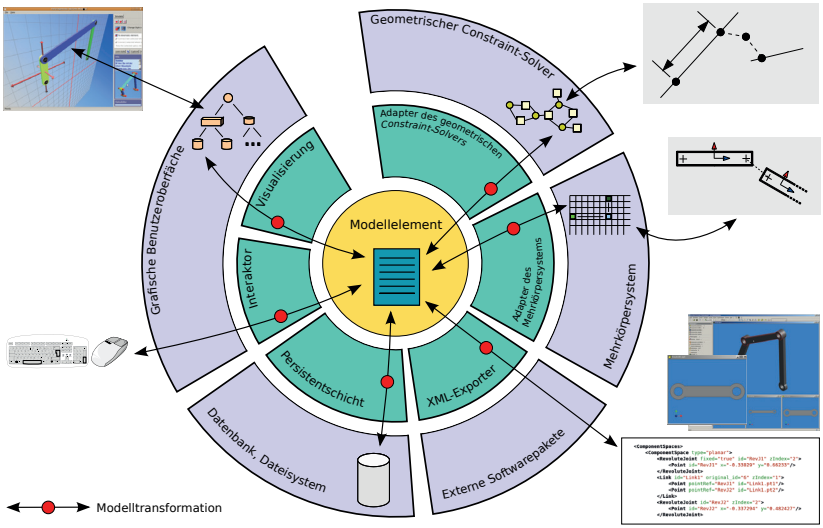


Abbildung 4.8: Reduktion der Anzahl an Konvertierungen durch die Synchronisierung spezifischer Modellsichten über ein Gesamtmodell als gemeinsame Datenbasis

beiden (alternativ einsetzbaren) Visualisierungselemente interpretieren die Informationen des Modellelementes und zeichnen eine im Raum entsprechend ausgerichtete grafische Repräsentation, bzw. ordnen Volumenkörper in einem Szenengraph an. Das Transformationselement für den geometrischen Constraint-Solver erzeugt in dessen spezifischen Modell geometrische Grundelemente (eine Linie und zwei Punkte). Zwischen der Linie und den Punkten wird jeweils ein Punkt-auf-Linie-Constraint vereinbart. Die Punkte erhalten außerdem ein Punkt-Abstand-Punkt-Constraint. Ähnlich verfährt das Transformationselement für die Mehrkörpersimulation. Hier werden entsprechende Strukturen für einen starren Körper als Bestandteile der Jacobi-Matrix des Mehrkörpersystems angelegt. Dessen Ersatzmasse und Masseverteilung ergeben sich durch eine Abschätzung aus der Länge des Getriebegliedes, können aber bei Bedarf auch vom Nutzer festgelegt werden.
Während die beiden Visualisierungselemente lediglich lesend auf das Modellelement zugreifen, schreiben die Simulationselemente unter Umständen Berechnungsergebnisse (z. B. Zustandsinformationen) in das Modellelement zurück.

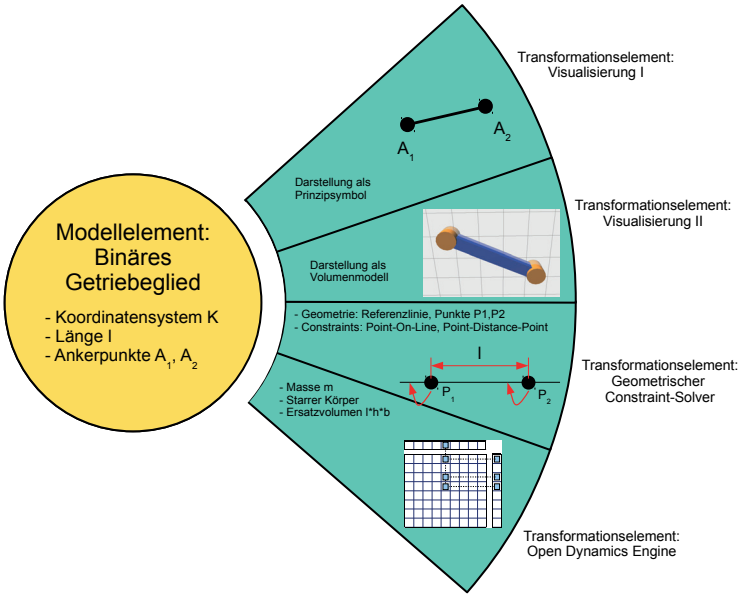


Abbildung 4.9: Beispiel für ein Modellelement mit Transformationselementen

4.5 Modellbildung in ausgewählten Domänen

Der vorliegende Abschnitt beschreibt den Aufbau typischer Modellelemente der im Rahmen dieser Arbeit betrachteten Anwendungsdomänen Mechanik, Optik und Regelungstechnik.

4.5.1 Mechanische Modellelemente

Die im Folgenden beispielhaft beschriebenen mechanischen Modellelemente wurden hauptsächlich zur Modellierung auf der Prinzipebene konzipiert. Dennoch schließen sie die Definition weiterer mechanischer Modellbestandteile, z. B. zur Abbildung von Funktionsstrukturen, keineswegs aus.

Beschreibungsmittel für Modelle der Mechanik

Das solverunabhängige Gesamtmodell setzt sich, wie beschrieben, aus Modellelementen zusammen. Diese repräsentieren vordefinierte Komponenten oder Bau-

steine, die dem Nutzer die Modellierung einer Vielzahl technischer Systeme erlauben. Die Beschreibung der Merkmale der Modellelemente selbst unterliegt wenigen Einschränkungen. Sie erfolgt mittels frei definierbarer, mit Semantik belegter Datenstrukturen. Dennoch erweist es sich aus informationstechnischer Sicht als zweckmäßig, häufig wiederkehrende Konstrukte zu vereinheitlichen und somit ihre Wiederverwendung zu ermöglichen. Beispielsweise könnte das Modellelement „Drehgelenk“ einen Vektor mit der Semantik „Rotationsachse“ enthalten. Auch die Beschreibung des Modellelementes „Rad“ umfasst einen Vektor mit gleicher Funktion. Es liegt daher nahe, beide Vektoren mit einer einheitlichen Semantik zu kennzeichnen.
Die Definition eines Satzes von Basiskonzepten ist nicht zwingend erforderlich, führt aber zu einer Vereinfachung der Verarbeitung, z. B. durch Transformationselemente. Für die Beschreibung mechanischer Modellelemente im Prototyp des Entwurfswerkzeugs erwiesen sich unter anderem folgende Konstrukte als geeignet:

- Koordinatensystem (Transformationsmatrix)
- Rotationsachse, Rotationspunkt, Translationsebene, Translationsvektor
- Anker (Hilfsmittel zur Verknüpfung mit anderen Elementen)
- sonstige Parameter (Skalare, Vektoren, Zeichenketten), z. B. Radius, Federkonstante, Bezeichnung

Aufbau ausgewählter mechanischer Modellelemente im Entwurfswerkzeug Hesym

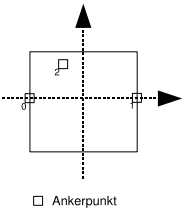
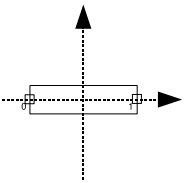
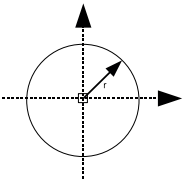
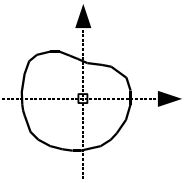
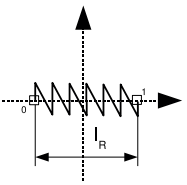
Mechanische Modellelemente weisen gemeinsame Eigenschaften auf und stellen in vielen Fällen Spezialisierungen anderer (mechanischer) Modellelemente dar. Es bietet sich daher an, bei ihrer Formulierung Konzepte der objektorientierten Informationsverarbeitung, wie z. B. Klassenbildung und Vererbung, anzuwenden. Auch aus softwaretechnischer Sicht liegt dies nahe, da die Umsetzung des Prototyps ohnehin in einer objektorientierten Programmiersprache geschieht.
Zunächst soll eine Basisklasse definiert werden, die jene Informationen umfasst, die allen mechanischen Modellelementen gemein sind. Diese Informationen betreffen hauptsächlich Aspekte der Verwaltung, wie beispielsweise die Bezeichnung eines Elementes oder seine Einordnung in eine Bauteilhierarchie (siehe Tabelle 4.3).
Von der Basisklasse leiten sich zwei weitere grundlegende Klassen ab, die die mechanischen Elemente in Körper und Kopplungen aufteilen. Diese Unterscheidung erwies sich bereits in anderen Vorhaben zur Modellierung mechanischer Elemente als zweckmäßig [Bri01].

Tabelle 4.3: In der Basisklasse mechanischer Modellelemente enthaltene Informationen

Eigenschaft	Beschreibung
Semantik	Vereinbarte Bezeichnung eines Konzepts, welches von allen beteiligten Simulations-, Visualisierungs- und Interaktionsmodulen zur Kommunikation und Verarbeitung verwendet wird.
Bezeichner (ID)	Eindeutige Identifikation der Instanz eines Elementes.
Name	Nutzervergebene Bezeichnung.
Elternelement	Sofern das Element Teil einer Aggregation ist, verweist es auf sein übergeordnetes Element.
Koordinatensystem	Relative Lage zum Weltkoordinatensystem oder zum Koordinatensystem des Elternelementes.
Attributliste	Liste von Parametern, die weitere Merkmale eines Elementes beschreiben (z. B. Maße). Der Inhalt der Liste wird jeweils von den spezialisierten Modellelementen festgelegt.

Zum besseren Verständnis sind in Tabelle 4.4 einige typische mechanische Elemente abgebildet. Eines der wichtigsten Modellelemente ist der starre Körper. Er beschreibt ein Koordinatensystem, innerhalb dessen weitere Koordinatensysteme (hier als Anker bezeichnet) positioniert sind. Diese Anker dienen der Verknüpfung des Körpers mit Kopplungen. Dabei entscheidet der Typ der Kopplung über die genaue Interpretation des Ankerpunktes (Beispiel Kugelgelenk: der Anker ist ein Punkt, Drehgelenk: der Anker setzt sich aus einem Punkt und einer Rotationsachse zusammen). Das *Binäre Getriebeglied*, das *Rad* und die *Kurvenscheibe* stellen Beispiele für Spezialisierungen des allgemeinen starren Körpers dar, die bestimmte Einschränkungen und zusätzliche Parameter umfassen. So muss ein binäres Getriebeglied genau zwei Anker besitzen, während ein Rad eine Rotationsachse und einen Radius definiert. Ein Beispiel für verformbare Elemente ist die Zugfeder (Tabelle 4.4, unten).

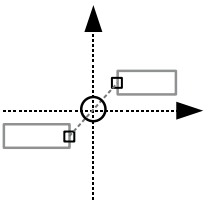
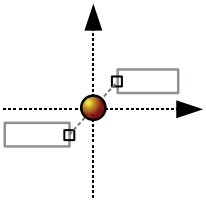
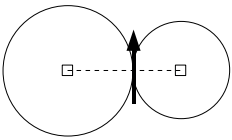
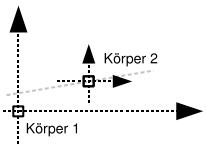
Tabelle 4.4: Beispiele für die Modellierung mechanischer Modellelemente

Element	Skizze
Starrer Körper • Semantik: Körper oder n-äres Getriebeglied • besitzt Eigenschaften des mechanischen Basiselementes (ID, Name, Elternelement, Koordinatensystem, Attributliste) • Liste von Ankerpunkten zur Kopplung mit anderen Elementen Bemerkung: Basis für alle Getriebeglieder, Räder und Kurvenscheiben	
Binäres Getriebeglied • Semantik: binäres Getriebeglied • Spezialisierung der Klasse Starrer Körper • Regel: genau zwei Ankerpunkte auf X-Achse Bemerkung: Typischer Spezialfall für Getriebeglieder	
Reib- oder Zahnrad • Semantik: Rad • Radius r • Ankerpunkt mit Rotationsachse	
Kurvenscheibe • Semantik: Kurvenscheibe als Teil eines Kurvengetriebes • Kurvenkontur (Polygon, Spline, ...) • Ankerpunkt mit Rotationsachse	
Feder • Semantik: Element zur Erzeugung einer Federkraft zwischen zwei Ankerpunkten • Ruhelänge l_R, Federkonstante k_F • besitzt genau zwei Anker	

Mechanische Kopplungen

Mechanische Kopplungen beschreiben Beziehungen zur Einschränkung der relativen Beweglichkeit zwischen mechanischen Körpern. Zusätzlich zu den Merkmalen des mechanischen Basiselements verfügt die Klasse der Kopplungen über eine Liste der von einer Instanz gekoppelten mechanischen Körper. Die einzelnen Spezialisierungen dieser Klasse repräsentieren die unterschiedlichen Kopplungstypen und bestimmen die einzuschränkenden Freiheitsgrade. Tabelle 4.5 führt einige Beispiele für mechanische Kopplungen auf.

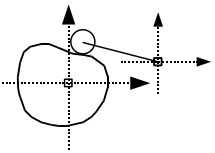
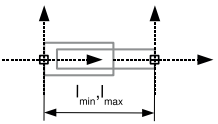
Tabelle 4.5: Beispiele mechanischer Kopplungen

Kopplung	Skizze
Drehgelenk • Semantik: Verbindung mit einem rotatorischen Freiheitsgrad • Rotationsachse	
Kugelgelenk • Semantik: Verbindung mit drei rotatorischen Freiheitsgraden • Rotationspunkt	
Abrollbeziehung • Semantik: Rotationswinkelübertragung zwischen Reib- oder Zahnrädern • koppelt genau zwei Räder	
Schubgelenk / prismatische Kopplung • Semantik: Verbindung mit einem translatorischen Freiheitsgrad • Translationsachse	

Zusammengesetzte mechanische Modellelemente

Oft besitzen bestimmte Gruppen mechanischer Modellelemente eine gemeinsame Bedeutung oder bilden eine logische Einheit. Beispielsweise erhält man aus Kurvenscheibe, Rad, Getriebeglied und Drehgelenken ein Kurvengetriebe mit Rollenhebelabtastung. Es liegt nahe, diese Elemente zusammenzufassen. Dabei kann der Nutzen solcher Verbünde über den einer reinen Gruppierung hinausgehen. Häufig enthält das Gesamtkonstrukt neue Aspekte, die sich in den Bestandteilen nur implizit widerspiegeln. Im Falle des Kurvengetriebes realisieren die beteiligten Bauteile mit ihren Eigenschaften (Kurvenkontur, Lage und Parameter des Kurveneingriffsglieds) eine bestimmte Übertragungsfunktion, die anhand der Einzelteile jedoch nur als Ergebnis beobachtet werden kann. Betrachtet man das Kurvengetriebe als Ganzes, kann durch die Vorgabe von Typ, Lage und Parametern der Abtastvorrichtung und einer Übertragungsfunktion die Kurvenkontur berechnet werden, was den üblichen Zugang zu einem solchen Problem darstellt (beschrieben in [VDI11] und [VDI87]).
Ein anderer Aspekt zusammengesetzter Elemente besteht darin, dass komplexere Konstrukte entstehen, zu deren Beschreibung existierende Elementtypen wiederverwendet werden. Darüber hinaus ist es möglich, dem Anwender spezialisierte Hilfsmittel zur Vereinfachung der Eingabe und Auslegung solcher Baugruppen zur Verfügung zu stellen. Tabelle 4.6 zeigt Beispiele für zusammengesetzte Modellelemente.

Tabelle 4.6: Beispiele zusammengesetzter mechanischer Modellelemente

Element	Skizze
Kurvengetriebe mit Rollenhebelabtastung • Semantik: Kurvengetriebe bestehend aus Kurvenscheibe und Abtastglied Rollenhebel • Liste von Subelementen (Kurvenscheibe, Rad, Getriebeglied, Drehgelenke) • Parameter: Übertragungsfunktion, Typ der Abtastung, Lageparameter	
Linearantrieb • Semantik: Antreibendes Element mit minimaler und maximaler Ausdehnung • Liste von Subelementen • Parameter: minimale und maximale Ausdehnung, Kraftrichtung	

4.5.2 Modellelemente der geometrischen Optik

Dieser Abschnitt stellt die für das Entwurfswerkzeug *Hesym* definierten Basiskonzepte zur Beschreibung optischer Eigenschaften von Systembestandteilen vor und demonstriert ihre Verwendung anhand der Definition typischer Modellelemente.

Beschreibungsmittel für Modelle der geometrischen Optik

Im Bereich der geometrischen Optik ist die Beschreibung von Systemen in Form optischer Schemata bereits mit einer kleinen Menge von Grundelementen möglich. Aus diesen Grundkonstrukten werden Bauelemente zusammengesetzt, die wiederum zu Baugruppen zusammengefügt werden.

Optische Wirkfläche

Optische Wirkflächen bilden die Basis für Körper mit optischen Eigenschaften. Sie beschreiben im Allgemeinen die Grenze zwischen zwei Medien mit folgenden Parametern (Abbildung 4.10):

- Lage und Form – Koordinatensystem und Geometrie der Grenzfläche (Ebene, Kugel, Freiformfläche, Flächenbegrenzung)
- Optische Eigenschaften – Absorption, Diffusion, Reflexion, Refraktion, Dämpfung, Dispersion, Polarisation

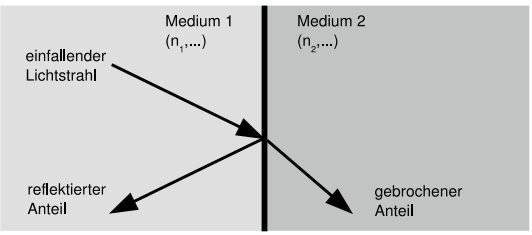


Abbildung 4.10: Mediengrenze als optische Wirkfläche

Es ist durchaus möglich, optische Wirkflächen zu räumlich unsinnigen oder unmöglichen Anordnungen zusammenzufügen. Die Verantwortung für den Aufbau korrekter optischer Körper (Innen-/Außenseite) liegt auf Bauteilebene. Hier wird sichergestellt, dass der Endanwender ausschließlich gültige Anordnungen realisieren kann.

Lichtquellen und Strahlenbündel

Modellierte Lichtquellen senden Lichtstrahlen aus, die in ihrem Verlauf von den optischen Wirkflächen beeinflusst werden. An einer Mediengrenze können aus jedem eintreffenden Strahl (z. B. durch Reflexion oder Brechung) weitere Strahlen entstehen. Es bildet sich eine Baumstruktur mit der Lichtquelle als Wurzelknoten aus. Die Strahlensegmente in diesem Baum besitzen eine Reihe von Zustandseigenschaften, z. B.

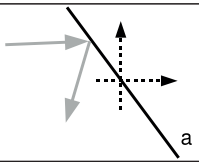
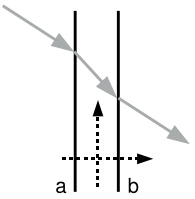
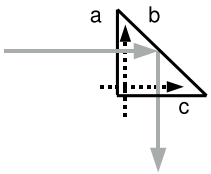
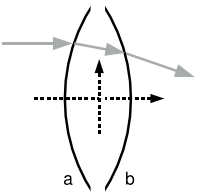
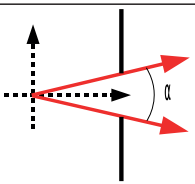
- Wellenlänge
- Intensität
- Polarisierung

Strahlenbündel werden durch eine Reihe charakteristischer Strahlen repräsentiert, die beispielsweise ein Volumen begrenzen oder eine Fläche abrastern.

Aufbau ausgewählter optischer Modellelemente

Tabelle 4.7 führt Beispiele optischer Modellelemente auf, die für den Prototyp des Entwurfswerkzeugs *Hesym* definiert wurden. Im Modell können sie eigenständig existieren oder als Teil mechanischer oder anderer Modellelemente auftreten. Optische Modellelemente lassen sich in aktive und passive Bauelemente unterscheiden. Passive Elemente enthalten eine oder mehrere Mediengrenzen, die sie ihres Typs entsprechend anordnen und parametrisieren (z. B. Spiegel, Linse mit Brennweite). Dabei entstehen nicht zwangsläufig geschlossene Körper, da nur für die optische Funktion notwendige Wirkflächen abgebildet werden. Beispielsweise definiert das Prisma in Tabelle 4.7 lediglich drei Grenzflächen (a,b,c).
Die wichtigsten Vertreter aktiver Bauelemente sind Lichtquellen. Ihre Modellierung erfolgt mittels einer Reihe von Lichtstrahlen, die das Abstrahlverhalten der Quelle charakterisieren. Die Strahlen werden dabei entweder als räumliches Raster angegeben oder dienen zur Begrenzung eines beleuchteten Volumens. Eine andere Gruppe aktiver Bauelemente sind Lichtsensoren, die empfangende Lichtstrahlen melden.

Tabelle 4.7: Beispiele für optische Modellelemente

Element mit optischen Eigenschaften	Skizze
Planspiegel • Lage im Raum • geometrische Ausdehnung • optische Wirkfläche a: 100% Reflexion	
Planparallele Platte • Lage im Raum • geometrische Ausdehnung, Abstand a,b • Mediengrenzen a,b: Refraktion	
Reflexionsprisma • Lage im Raum • geometrische Ausdehnung • Mediengrenzen a,b,c: Refraktion • an Mediengrenze b kann Totalreflexion auftreten	
Bikonvexe sphärische Linse • Lage im Raum, optische Achse • Festlegung der Brennweite über Radius, Abstand und Begrenzung der Kugelsegmente • Mediengrenzen a und b: Refraktion	
Lichtquelle mit Rechteckblende • Lage im Raum, Strahlrichtung • Öffnungswinkel • Grenzstrahlen des Strahlenbündels	

4.5.3 Regelungstechnische Modellelemente

Ein wichtiger Anwendungsfall der Regelungstechnik ist die Modellierung von Regelkreisen. Zu deren Beschreibung findet man verbreitet Blockdiagramme, in denen Komponenten (z. B. Regelstrecke, Sensor, Regler) als Blöcke repräsentiert sind, die

über Signalflüsse miteinander in Beziehung stehen. Eine Darstellung, die bezüglich ihres Konkretisierungsgrades mit dem technischen Prinzip des klassischen Maschinenbaus vergleichbar ist, kommt in der Regel lediglich für einzelne Teilbereiche, wie z. B. das Messprinzip von Sensoren, zur Anwendung.
Die Darstellung der Blockschaltbilder ist nach DIN EN 60027-6 [DIN06] genormt, erfährt in der Praxis allerdings viele Abweichungen hinsichtlich Anordnung und Aufteilung der Blöcke sowie der Bezeichnung der beteiligten Größen. Allen Varianten gemein ist jedoch, dass sie Eigenschaften einer Funktionsstruktur aufweisen, deren Teilblöcke mittels detaillierterer Blockschaltbilder beschrieben werden können. Ein solcher Ansatz soll auch hier verfolgt werden. Dabei liegt der Fokus auf Sensor, Regler und Aktor, da die Regelstrecke selbst meist anderweitig (domänenspezifisch) beschrieben ist (z. B. als technisches Prinzip einer mechanischen Struktur).
Sofern der eigentliche Regler ausgelegt werden soll, erschwert das so ausgelagerte, implizit vorhandene Streckenmodell den Einsatz regelungstechnischer Entwurfsverfahren, die ein explizites Modell der Strecke, beispielsweise in Form von Differentialgleichungen, fordern. Neben der Formulierung eines zusätzlichen Hilfsmodells zum Zwecke des Entwurfs bietet sich hier vor allem die Nutzung von Entwurfsverfahren an, die eine experimentelle Modellbildung zulassen (z. B. durch Auswertung der Systemantwort auf Testsignale, [Lun08]).

Beschreibungsmittel für regelungstechnische Modellelemente

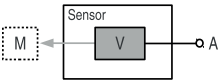
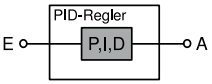
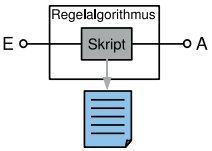
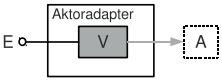
Der Grundbaustein zur Beschreibung regelungstechnischer Strukturen ist der Funktionsblock. Dieser besitzt Schnittstellen zu seiner Umgebung in Form von Ein- und Ausgangskanälen (Größen als Skalare oder Vektoren), innere Zustandsgrößen und eine Übertragungsfunktion. *Hesym* stellt sowohl vordefinierte Blöcke, wie PID-Regler oder Integratoren, als auch benutzerdefinierte Bausteine zur Verfügung. Letztere stellen eine beliebige Kombination aus Eingabe-, Ausgabe- und Zustandsgrößen sowie einer Übertragungsfunktion in Form von Skriptcode dar, was die Realisierung spezieller Regelalgorithmen oder komplexer Konstrukte wie speicherprogrammierbarer Steuerungen (SPS/PLC) erlaubt.

Aufbau ausgewählter regelungstechnischer Elemente in Hesym

In Tabelle 4.8 sind Beispiele für regelungstechnische Modellelemente in *Hesym* aufgeführt. Hierzu zählen die bereits erwähnten, parametrisierbaren Standardelemente wie PID-Regler und lineare Zustandsregler sowie benutzerdefinierte Bausteine auf Basis der Skriptsprache LuaScript [IdC06]. Eine Besonderheit stellen die Adapterelemente für Sensoren und Aktoren dar. Sie dienen zur Kopplung regelungstechni-

scher Strukturen mit Modellelementen anderer Domänen. Beispielsweise könnte ein Sensoradapter ein Signal aus einem optischen Messsystem ermitteln, das als optisches Schema in *Hesym* modelliert wurde. Das Signal würde vorverarbeitet und zur Nutzung durch das regelungstechnische Modell in einen Informationskanal umgesetzt. Das Gegenstück zum Sensoradapter stellt der Aktoradapter dar, der einen regelungstechnischen Informationskanal in das Ansteuersignal eines Aktors konvertiert. Mit den Adaptern wird den sehr unterschiedlichen Abstraktionsmöglichkeiten bei der Modellierung Rechnung getragen. So könnte ein elektrischer Antrieb sowohl konkret über eine elektrische Spannung, andererseits über abstrakte Größen wie Drehzahl oder Drehwinkel angesteuert werden. Mit der Möglichkeit zur Vorverarbeitung in Sensor- und Aktoradapter lassen sich diese konzeptionellen Unterschiede der Abstraktionsniveaus überbrücken.

Tabelle 4.8: Beispiele für regelungstechnische Elemente

Regelungstechnisches Element	Skizze
Sensoradapter • Referenz der abzufragenden Messgröße M • Vorverarbeitungsalgorithmus V / Übertragungsfunktion • Ausgabekanal A	
PID-Regler • Eingangskanal E • Faktoren für P, I und D • Ausgabekanal A	
Benutzerdefinierter Baustein • Eingangskanal E • Algorithmus in Skriptsprache • Ausgabekanal A	
Aktoradapter • Eingangskanal E • Vorverarbeitungsalgorithmus V • Referenz des anzusteuernden Aktors A	

4.6 Integration von Berechnungsverfahren

Dieser Abschnitt setzt sich mit der Frage auseinander, wie das Verhalten des domänenübergreifenden Gesamtmodells simuliert werden kann. Der Grundidee des Entwurfswerkzeugs als Simulationsplattform folgend findet a priori keine Festlegung auf bestimmte Berechnungsverfahren statt. Vielmehr sollen Wege aufgezeigt werden, mit denen eine Integration nahezu beliebiger Simulationsmethoden gelingt. Dies erweist sich unter anderem auch deshalb als notwendig, da das Werkzeug eine Erweiterbarkeit um zusätzliche Modellelemente oder weitere Domänen vorsieht, die wiederum anwendungsspezifische Simulationsverfahren erfordern. Im Folgenden werden Anforderungen bezüglich der Eignung von Berechnungsverfahren aufgestellt, der Ablauf der Transformation des übergreifenden Modells in solverspezifische Darstellungen beschrieben, ein Konzept zur vereinheitlichten Kommunikation des Entwurfswerkzeugs mit den Berechnungsmodulen erarbeitet und Verfahren zur Koordination von Berechnungsmodulen entworfen.

4.6.1 Begriffe Simulation, Berechnungsmodul und Berechnungsverfahren

Der Begriff „Simulation“ wird in der wissenschaftlichen Literatur im Zusammenhang mit unterschiedlichsten Sachverhalten verwendet [Spa09]. Im Rahmen dieser Arbeit soll Simulation als ein Hilfsmittel zur computergestützten Abschätzung des Verhaltens oder der Eigenschaften eines entworfenen Produktes auf der Basis der im Modell festgelegten Merkmale betrachtet werden.
Die häufig betrachtete „Simulationspipeline“ umfasst neben der eigentlichen Berechnung auch die Begriffe Modellierung, Implementierung, Visualisierung und Validierung (z. B. [BZB09]). Der vorliegende Abschnitt konzentriert sich auf den Aspekt der Berechnung, da die verbleibenden Aufgaben von anderen Teilen des Entwurfswerkzeugs abgedeckt werden. So existiert bereits ein Produktmodell, das zum Zwecke der Simulation lediglich anwendungsspezifisch formuliert und gegebenenfalls erweitert wird. Auch die Visualisierung des Modells und daraus erzeugter Informationen erfolgt getrennt von deren Berechnung.
Die zur Berechnung eingesetzten mathematischen Verfahren (Berechnungsverfahren) werden in Form von Berechnungsmodulen als Software umgesetzt.

4.6.2 Auswahl von Simulationsverfahren zum Einsatz in frühen Phasen der Produktentwicklung

Bei der Auswahl zu integrierender Berechnungsverfahren stellen sich zunächst zwei Fragen: Welche Informationen oder Berechnungsergebnisse sind in frühen Phasen

von Interesse? Und: Welche Verfahren sind in diesen Phasen aus technischer Sicht geeignet?
Es ist kaum möglich, beide Fragen allgemeingültig zu beantworten, da sie jeweils im Kontext eines bestimmten Anwendungsfalls betrachtet werden müssen. Grundsätzlich lässt sich jedoch festhalten, dass Berechnungen in den frühen Phasen dazu dienen, eine grundlegende Abschätzung des Systemverhaltens zu ermöglichen, auch um Entscheidungskriterien für den Vergleich von Lösungsvarianten zu schaffen. Zu den relevanten Informationen für die Domänen Mechanik, Optik und Regelungstechnik gehören z. B. kinematische Größen wie Beschleunigungen, Geschwindigkeiten, Bewegungsbahnen, aber auch Eigenschaften wie Bauräume und auftretende Kräfte, der Verlauf optischer Strahlengänge sowie Aspekte der Reglerauslegung, der Einfluss von Störgrößen oder die Stabilität geregelter Systeme.
Auch aus technischer Sicht existieren wenige Einschränkungen bezüglich der einzusetzenden Verfahren. Ein grundsätzlicher Gedanke ist, dass die Informationsmenge des recht abstrakten domänenübergreifenden Gesamtmodells als Eingabedaten für das jeweilige Berechnungsverfahren ausreichend sein muss, bzw. die Generierung notwendiger Informationen mit möglichst geringem manuellen Aufwand zulässt. Umgekehrt sollte die Rücktransformation von Berechnungsergebnissen oder eine Zuordnung zu den Elementen des solverunabhängigen Gesamtmodells möglich sein. Weiterhin werden echtzeitfähige Verfahren bevorzugt, um dem Nutzer eine zeitnahe Rückkopplung über die Auswirkungen seiner Eingaben zu geben und somit dem Anspruch eines interaktiven Simulationssystems gerecht zu werden.
Typische Beispiele für eingesetzte Berechnungsverfahren sind

- ein geometrischer Constraint-Solver zur Berechnung geometrischer Optik und kinematischer Problemstellungen,
- ein Mehrkörpersystem-Solver zur Simulation der Dynamik mechanischer Systeme,
- ein Raytracer zur Berechnung optischer Strahlengänge,
- die Finite-Elemente-Methode zur Bestimmung von Kräften (beispielsweise in Tragwerken) oder
- ein Skriptinterpreter zur Ausführung von Algorithmen der Regelungstechnik und sonstiger Informationsverarbeitung.

4.6.3 Das domänenübergreifende Gesamtmodell und solverspezifische Modelle

Da das Entwurfswerkzeug als Plattform für den Einsatz unterschiedlichster Berechnungsverfahren konzipiert ist, orientiert sich die Formulierung des domänenübergreifenden Modells nicht an den mathematischen Modellen eines bestimm-

ten Berechnungsverfahrens. Das Gesamtmodell ist ein Graph, dessen Knoten die Modellelemente repräsentieren. Sie besitzen eine bestimmte Semantik sowie eine Reihe von Attributen, mit denen sie funktionsrelevante Eigenschaften der Elemente abbilden. Die Kanten des Graphen beschreiben Beziehungen zwischen den Modellelementen.
Das domänenübergreifende Gesamtmodell wird als solches nicht direkt von den Berechnungsverfahren verarbeitet. Es werden daher solverspezifische Modelle generiert, die aus dem solver-unabhängigen Gesamtmodell abgeleitet werden und in der Folge mit diesem in Beziehung stehen. Abbildung 4.11 verdeutlicht diesen Zusammenhang am Beispiel eines optomechanischen Systems, des Verstellmechanismus für einen Spiegel.

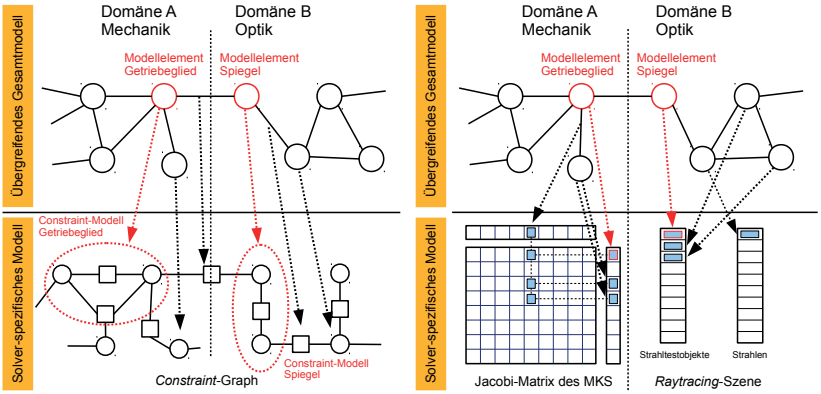


Abbildung 4.11: Domänenübergreifendes Gesamtmodell und solverspezifische Modelle

Der linke Teil des Bildes demonstriert die Modelltransformation bei Verwendung eines einzelnen Berechnungsmoduls für beide Domänen. Die Modellelemente des mechanischen und des optischen Teilsystems werden mit den Beschreibungsmitteln eines geometrischen Constraint-Solvers abgebildet. Dieser ist in der Lage, das transformierte Modell zu verarbeiten. Berechnete Modellzustände werden in die Repräsentation des domänenübergreifenden Gesamtmodells zurücktransformiert, so dass sie anderen Modellsichten (Transformationselementen) zur Verfügung stehen, beispielsweise der Visualisierung.
Das zweite Beispiel in Abbildung 4.11 (rechts) beschreibt einen Fall, in dem zwei unterschiedliche Berechnungsmodule zum Einsatz kommen. Das mechanische Teilmodell wird in die Datenstrukturen eines Mehrkörpersystem-Solvers transformiert,

während anhand des optischen Teilmodells eine Reihe von Kollisionsgeometrien für einen Raytracer erzeugt werden. Die Auswertung beider Modelle erfolgt separat, so dass eine Synchronisierung an den Schnittstellen notwendig wird. Auch hierfür findet das domänenübergreifende Gesamtmodell Verwendung, wobei die korrekte Abfolge der Berechnungs- und Synchronisierschritte gewährleistet werden muss (Abschnitt 4.6.5).
Art und Umfang der im solverspezifischen Modell erzeugten Informationen (z. B. Datenstrukturen) variieren in Abhängigkeit von Modellelement und Anwendungsfall. Dies liegt zum einen in der Verschiedenheit der möglichen Berechnungsverfahren begründet. Zum anderen ist jeder Knoten in der Lage, Transformationselemente von sehr unterschiedlicher Komplexität zu definieren. So könnte beispielsweise aus einem Modellelement „Kugelgelenk“ in einem geometrischen Constraint-Netz nur ein einziges geometrisches Element (Punkt-inzident-Punkt-Constraint) erzeugt werden, während das Modellelement „optisches Prisma“ zur Erstellung dreier Grenzflächenbeschreibungen als Kollisionsobjekte eines Raytracers führt.

4.6.4 Transformation des solverunabhängigen Modells

Dem Ansatz aus Abschnitt 4.4.4 folgend, verwendet das Entwurfswerkzeug Transformationselemente zur Umsetzung des domänenübergreifenden Gesamtmodells in ein solverspezifisches Modell. Während die Transformationselemente die Verbindung zwischen den Modellelementen und dem Berechnungsmodul herstellen, bildet das Berechnungsmodul eine vereinheitlichende Schnittstelle zu dem als Programmbibliothek implementierten Berechnungsverfahren (Abbildung 4.12). Ein Transformationselement repräsentiert somit den Teil des solverspezifischen Modells, der einem bestimmten domänenunabhängigen Modellelement entspricht. Für jedes Modellelement können beliebig viele solcher Transformationselemente definiert werden, um eine Auswertung des Modells durch eine Vielzahl unterschiedlicher Berechnungsverfahren zu ermöglichen. Auf diese Weise wird das Modellelement um berechnungsspezifische Sichten oder Aspekte erweitert.

Zu den Aufgaben des Transformationselementes gehören

- die Erzeugung solverspezifischer Informationen und Datenstrukturen,
- die Speicherung der Referenzen zwischen den Datenstrukturen des Berechnungsmoduls und denen des Modellelementes und
- die Synchronisierung der Daten des Modellelements und des Berechnungsmoduls.

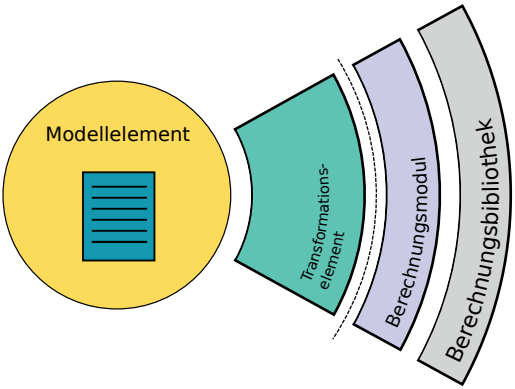


Abbildung 4.12: Transformationselement als Schnittstelle zwischen Modellelement und Berechnungsbibliothek

Der grundlegende Aufbau eines Transformationselementes ist in Abbildung 4.13 dargestellt. Transformationselemente referenzieren das Modellelement, für dessen Transformation sie eingesetzt werden und die von ihm erzeugten Datenstrukturen im internen Modell des Berechnungsmoduls. Weiterhin besitzen sie eine Transformationsvorschrift, die einerseits aus den Informationen des Modellelementes für das Berechnungsmodul verarbeitbare Daten erzeugt und andererseits eine Rücktransformation von Berechnungsergebnissen in das domänenübergreifende Modell ermöglicht.
Da die Transformationsvorschrift zwar meist recht einfach, aber dennoch individuell ist, muss für jede Kombination von Modellelement und Berechnungsmodul ein Transformationselement definiert werden. Dies geschieht zum Zeitpunkt der Implementierung der Software und erfordert später keine weitere Einflussnahme seitens des Anwenders.

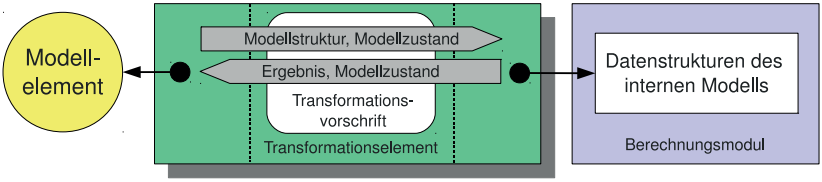


Abbildung 4.13: Aufbau des Transformationselementes

4.6.5 Koordination von Berechnungsmodulen

Berechnungsmodule weisen deutliche Unterschiede hinsichtlich ihrer internen Arbeitsweise, der zugrundeliegenden mathematischen Verfahren und des Formates der Ein- und Ausgabedaten auf. Zu ihrer Integration in eine Software-Plattform ist eine gewisse Anpassung notwendig, da die Module später über eine einheitliche Schnittstelle mit dem Entwurfswerkzeug kommunizieren sollen.

Vereinheitlichung der Schnittstellen von Berechnungsmodulen

Für die Koordination von Berechnungsvorgängen in *Hesym* ist eine Reihe von Konzepten notwendig, die von allen Berechnungsmodulen unterstützt werden. Dazu gehören

- ein Zeitkonzept — z. B. zur Definition von Simulationsschritten und Zeitspannen, zur zeitlichen Zuordnung von Ereignissen,
- ein Konsistenzbegriff — das Modell oder Teile davon befinden sich in einem gültigen Zustand,
- die Unterscheidung von Topologieänderung, Parameterwertänderung und Zustandsänderung und
- die Beschreibung von Abhängigkeiten bezüglich der Berechnungsreihenfolge.

Nicht alle Berechnungsverfahren erfüllen diese Anforderungen. Daher werden solverspezifische Adapter geschaffen, die den jeweiligen Berechnungsmodulen eine einheitliche Schnittstelle vermitteln (Abbildung 4.14). Beispielsweise besitzt der eingesetzte geometrische Constraint-Solver kein Zeitkonzept, da er jeden Modellzustand innerhalb eines Berechnungsschrittes erreichen kann. Es ist somit die Aufgabe des Adapters, geometrische Änderungen auf eine Zeitbasis abzubilden, um Geschwindigkeiten, Beschleunigungen u.ä. darstellen zu können.
Das Konzept des Entwurfswerkzeugs setzt voraus, dass die Berechnungsverfahren als Programmbibliotheken vorliegen. Sollten sich Änderungen in diesen Bibliotheken ergeben, muss lediglich eine Anpassung des Adapters (und ggf. der Transformationselemente) vorgenommen werden. Die Simulationsplattform selbst bleibt unberührt.

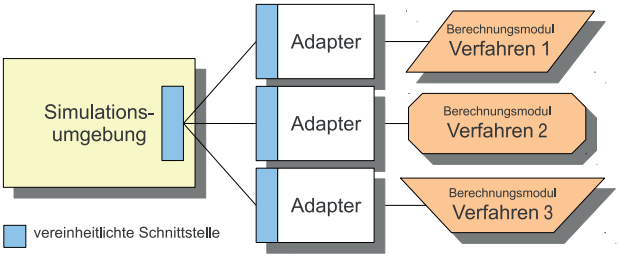


Abbildung 4.14: Adapter zur Vereinheitlichung der Schnittstellen von Berechnungsverfahren

Koordination des Berechnungsvorgangs

Jede Änderung eines Modellparameters kann grundsätzlich einen inkonsistenten Modellzustand erzeugen. Solche Veränderungen können beispielsweise durch Nutzerinteraktionen hervorgerufen werden, aber auch in Folge bestimmter Simulationsprozesse (z. B. Vorgabe von Randbedingungen) auftreten. Damit sind Maßnahmen zur Wiederherstellung der Modellkonsistenz notwendig.
Es lassen sich zwei Fälle unterscheiden:

- Das gesamte Modell kann von einem einzigen Berechnungsverfahren behandelt werden (z. B. mechanische und optische Komponenten des Systems werden von einem geometrischen Constraint-Solver behandelt).
- Das Gesamtmodell wird in mehrere solverspezifische Teilmodelle zerlegt (z. B. mechanische Komponenten mittels MKS und regelungstechnische Bestandteile durch einen Skriptinterpreter behandelt).

Der erste Fall gestaltet sich vergleichsweise unkompliziert, da alle Abhängigkeiten zwischen Modellelementen innerhalb des transformierten Modells eines einzigen Berechnungsmoduls bestehen und dort lokal behandelt werden können. Findet hingegen eine Aufteilung des Gesamtmodells in unterschiedliche solverspezifische Teilmodelle statt, entsteht zusätzlicher Koordinationsaufwand, da alle Teilmodelle separat ausgewertet und synchronisiert werden müssen. Die Schnittstellen zwischen den modellierten Domänen werden hier ausschließlich über das Gesamtmodell abgebildet und stellen somit Randbedingungen für die Behandlung der Teilmodelle dar (siehe auch Abbildung 4.11). Darüber hinaus muss eine geeignete Reihenfolge von Berechnungsschritten zur Auswertung der Teilmodelle gefunden werden. Hierzu sind mehrere Strategien denkbar, die sich in Aufwand und Leistungsfähigkeit unterscheiden.

Eine erste und einfache Variante ist eine fest vorgegebene oder vom Nutzer gewählte Reihenfolge von Berechnungen. Eine Vorschrift dieser Art für das Beispiel in Abbildung 4.11 könnte folgendermaßen aussehen:

> Berechne für jeden Zeitschritt zuerst mittels MKS das Verhalten des Bewegungssystems, übernimm diese Ergebnisse als Randbedingungen und bestimme ausgehend davon mit dem Raytracer den Strahlengang des optischen Systems.

Der Vorteil dieser Herangehensweise liegt im geringen Aufwand seiner Umsetzung. Als nachteilig erweist sich, dass auf diese Weise keine zyklischen Abhängigkeiten zwischen den Teilmodellen behandelt werden können. Andererseits zeigen Praxisbeispiele, dass die Teilmodelle der beteiligten Domänen in vielen heterogenen Systemen nur über eine geringe Zahl von Parametern in Verbindung stehen und diese Variante somit oft ausreichend ist (Beispiele dafür siehe Kapitel 6).
Eine Erweiterung des Ansatzes der Berechnung mit einer festgelegten Abfolge ist die automatische Bestimmung der Reihenfolge notwendiger Berechnungsschritte. Hierbei wird ausgehend von der Quelle einer Modelländerung (z. B. Nutzerinteraktion) eine Folge von Solveroperationen gesucht, die zur Herstellung der Modellkonsistenz führen. Als Grundlage kann ein Graph dienen (z. B. domänenübergreifendes Gesamtmodell), der Abhängigkeiten zwischen den Teilmodellen und deren Modellbestandteilen (zumindest qualitativ) beschreibt. Diese Lösung besitzt zwar ähnliche Einschränkungen bezüglich der Behandlung zyklischer Abhängigkeiten wie die manuelle Festlegung der Berechnungsreihenfolge, entbindet den Nutzer aber von Eingriffen in die Organisation des Berechnungsprozesses.
Darüber hinaus sind weitere Varianten denkbar, die mittels aufwendigerer Koordinationsverfahren in der Lage sind, auch Systeme mit zyklischen Abhängigkeiten zwischen den solverspezifischen Teilmodellen zu verarbeiten. Diese erfordern allerdings eine detailliertere Beschreibung und Auswertung der Zusammenhänge zwischen den Modellbestandteilen, auch unter Berücksichtigung der Eigenschaften der eingesetzten Berechnungsverfahren, und sollen im Rahmen dieser Arbeit nicht genauer untersucht werden. Dennoch bietet sich hier ein umfangreiches Arbeitsgebiet zur Weiterentwicklung des Entwurfswerkzeugs für heterogene Systeme.

4.7 Visualisierung und Interaktion

Dieser Abschnitt entwickelt ein Konzept, das es dem Anwender erlaubt, mit dem Entwurfswerkzeug zu interagieren. Dieser Aspekt ist von wesentlicher Bedeutung für die Gebrauchstauglichkeit (Usability) und somit für den Nutzen des Werkzeugs.

4.7.1 Ziel und Bedeutung eines Interaktionskonzeptes

Die Benutzerschnittstelle bildet den Kommunikationskanal zwischen Mensch und Software. Sie soll es dem Anwender ermöglichen, seine Ziele, Vorhaben, Aktionen und Anfragen dem Rechner mitzuteilen, während dieser Anfrageergebnisse und Informationen über seinen inneren Zustand kommuniziert [HH93]. Benutzerschnittstellen werden häufig als wichtigster Aspekt eines Softwaresystems betrachtet, da sie als sichtbarer Teil des Systems in der Wahrnehmung vieler Anwender mit dem Softwaresystem gleichgesetzt werden [Gal07].
Auch aus Sicht eines Entwurfswerkzeugs für den Einsatz in der Konzeptphase ist den Interaktionsmöglichkeiten der Benutzerschnittstelle eine große Bedeutung beizumessen, da zu den Hauptaufgaben eines solchen Werkzeugs die Eingabe und Manipulation von Modellen sowie die Präsentation von Simulationsergebnissen gehören. Die volle Leistungsfähigkeit der dem Softwarewerkzeug zugrundeliegenden Algorithmen und Modelle eröffnet sich dem Anwender erst mit dem Einsatz einer durchdachten und problemangepassten Benutzerschnittstelle. Der Anwender wird so in die Lage versetzt, eine Vielzahl von Lösungsvarianten in vertretbarer Zeit zu konzipieren und miteinander zu vergleichen.

4.7.2 Anforderungen an das Interaktionssystem des Entwurfswerkzeugs für räumliche heterogene Systeme

Die Wahl eines Interaktionskonzeptes ist in hohem Maße anwendungsabhängig. Anforderungen leiten sich aus der zu lösenden Aufgabe, den beteiligten Nutzern sowie den zur Verfügung stehenden Interaktionsgeräten ab.
Im Falle von *Hesym* besteht die Aufgabe in der Modellierung räumlicher technischer Systeme, die zu einem Großteil durch symbolhafte geometrische Strukturen dargestellt werden, deren Lage, Dimensionen und andere Parameter vom Nutzer einzugeben sind. Dementsprechend sind für ein derartiges Werkzeug Konzepte erforderlich, die eine Arbeit im dreidimensionalen Raum erlauben. Solche Konzepte lassen sich nach [Bow05] in die vier Grundaufgaben räumlicher Interaktion unterteilen:

- Navigation
- Selektion
- Manipulation und
- Steuerung des Systemzustandes.

Diese Einteilung wurde ursprünglich in Bezug auf Anwendungen der Virtuellen Realität vorgenommen, gilt aber mit veränderter Gewichtung auch für Werkzeuge

auf dem Gebiet des rechnerunterstützten Entwurfs. So gestalten sich Navigation und Selektion im Entwurfswerkzeug *Hesym* verhältnismäßig unkompliziert, während Manipulation und Steuerung des Systemzustandes eine größere Bedeutung zukommt. Dies ergibt sich aus der Anforderung, eine große Zahl unterschiedlicher und teilweise komplexer Interaktionsaufgaben zu koordinieren, die zudem häufig die exakte Spezifikation von Anordnungsrelationen oder Maßen erfordern.
Auch der Kenntnisstand und die Gewohnheiten potenzieller Nutzer beeinflussen die Auslegung des Interaktionskonzepts. Der gelegentlich angeführte Begriff der „intuitiven Benutzeroberfläche" erscheint zunächst wenig sinntragend, erhält aber im Zusammenhang mit der Berücksichtigung der Erfahrungswerte bestimmter Nutzer eine gewisse Substanz. Es liegt nahe, dass wiedererkannte Eigenschaften oder Interaktionsmuster aus weit verbreiteter Software den Anwendern einer neuen Software gewohnt und plausibel erscheinen. Eine weitere Strategie, einem Nutzer die Arbeit mit einem Softwarewerkzeug zu erleichtern, zielt auf dessen Begriffswelt ab. Für Visualisierung und Interaktion sollten gewohnte, fachspezifische Darstellungsformen gewählt werden, die dem Anwender einen vertrauten Zugang zu den präsentierten Inhalten ermöglichen.
Die verfügbaren Eingabegeräte stellen die dritte Quelle von Anforderungen dar. [Bow09] empfiehlt, Interaktionstechniken an die verwendeten Geräte anzupassen. Dies betrifft insbesondere die Anzahl der Freiheitsgrade, die von Eingabegeräten gebunden und für Interaktionen benötigt werden. Da *Hesym* vornehmlich zur Verwendung an herkömmlichen PC-Arbeitsplätzen vorgesehen ist und somit lediglich auf zweidimensionale Ein- und Ausgabegeräte zurückgreift (Maus, Bildschirm), sind Verfahren zur Interpretation zweidimensionaler Eingaben im dreidimensionalen Raum erforderlich.
Eine weitere, vorwiegend technisch bedingte Anforderung ergibt sich aus der modularen Softwarearchitektur des Werkzeugs. Im Gegensatz zu einer monolithisch aufgebauten Software, bei der alle Modellbestandteile und die darauf anwendbaren Operationen im Vorfeld bekannt sind, kann für ein erweiterbares Werkzeug kein ganzheitliches Interaktionskonzept gefunden werden. Jedes hinzukommende Modellelement oder Simulationsverfahren benötigt spezialisierte Darstellungs- und Interaktionsmethoden. Das Interaktionskonzept sollte daher das Hinzufügen und die Koordination zusätzlicher Applikationslogik vorsehen, die nicht zu Änderungen am Programmcode der gesamten Software führen [Brü10].
Mittlerweile existiert eine Vielzahl von Softwarewerkzeugen, in denen unterschiedlichste Interaktionskonzepte umgesetzt und erprobt wurden. Obwohl hierbei gelegentlich auch neue Lösungen zum Einsatz kommen, stellen diese Konzepte selten vollständig neue Verfahren dar und greifen stattdessen häufig auf bekannte Teilkonzepte zurück. Die Gründe hierfür könnten in wiederkehrenden oder ähnlichen

Anforderungen und Nutzergruppen sowie unveränderten Ein- und Ausgabegeräten zu suchen sein. Dementsprechend besteht die Hauptaufgabe bei der Entwicklung eines Interaktionskonzeptes in der zweckmäßigen Kombination und Anpassung bewährter Teillösungen.
Für *Hesym* bedeutet dies, Interaktionsverfahren aus Simulationswerkzeugen wie *MASP* (Baukastensystem, semantikbehaftete Symbole; siehe Abschnitt 3.3.1) mit dreidimensionalen Interaktionsverfahren aus der Welt der VR- und CAD-Anwendungen zu kombinieren und in ein Gesamtkonzept zu überführen. Dies soll auch unter Berücksichtigung der Richtlinien zur Ergonomie der Mensch-System-Interaktion nach DIN EN ISO 9241 ([ISO95] Teile 11-17 und 110) sowie einer Usability-Studie zu *MASP* [KF06] geschehen.

4.7.3 Interaktionskonzept von Hesym

Das Gesamtkonzept zur Interaktion umfasst eine Reihe von Aspekten und Teilkonzepten, die hier als Liste aufgeführt und im Folgenden detailliert beschrieben werden sollen. Da *Hesym* eine Software-Plattform zur Entwicklung domänenübergreifender Entwurfswerkzeuge darstellt, sollten diese Punkte auch als Handlungsempfehlung für die Implementierung zukünftiger Interaktionskomponenten verstanden werden.
In *Hesym* eingesetzte Teilkonzepte der Interaktion sind

- die Nutzung der geometrischen Eigenschaften der Modellelemente zur räumlichen Darstellung (keine Reduktion auf Funktionsblöcke oder mathematische Darstellung, gegebenenfalls auch geometrische Darstellung nichtgeometrischer Eigenschaften),
- die Verwendung der Semantik der Modellelemente in Verbindung mit Vereinbarungen und Zusatzwissen zur Bindung von Freiheitsgraden und damit Vereinfachung der Interaktion,
- interaktive Simulation mit sofortiger Rückmeldung und der Möglichkeit, Konsequenzen von Modellieroperationen direkt zu beobachten,
- Hilfestellung über kontextabhängige Instruktionen und damit ständige Dokumentation des Interaktionszustandes,
- Interaktoren als Kontexte zur Interpretation von Nutzereingaben und hierarchische Wiederverwendung von Interaktoren,
- objektbezogene Aktionen (als Kontextmenü oder Liste von Handlungsmöglichkeiten),

- Reduktion von Freiheitsgraden zur Interaktion mit dreidimensionalen Strukturen unter Nutzung zweidimensionaler Ein- und Ausgabegeräte,
- Unterstützung unterschiedlicher Eingabegeräte, allerdings mit Hauptaugenmerk auf konventionelle PC-Arbeitsplätze (Verbreitungsgrad).

Nutzung geometrischer Eigenschaften für die Modellvisualisierung

Ein grundlegender Gedanke des Entwurfswerkzeugs für heterogene Systeme ist die grafisch-interaktive Kommunikation mit dem Benutzer. Dieser bevorzugt die Interaktion mit sichtbaren Objekten [Stü09]. Dementsprechend sollte die Darstellung des Modells gestaltet sein. Sofern modellierte Strukturen eine räumliche Ausdehnung und Lage besitzen, sollten diese Eigenschaften visualisiert und so für die Interaktion nutzbar gemacht werden. Dies ist insbesondere dann von Bedeutung, wenn Lage und Form funktionsrelevant sind. Gegenüber z. B. einer rein mathematischen Repräsentation oder einer Blockdiagrammdarstellung wird in diesem Fall eine deutlich bessere Anschaulichkeit erreicht.
Sehr deutlich wird dies unter anderem bei Problemstellungen aus den Bereichen der Kinematik oder der Optik. Demgegenüber erscheint eine räumliche Modellvisualisierung z. B. bei der Darstellung elektronischer Schaltungen oder von Regelkreisen weniger wichtig, da hier Lage und Form der Bauteile nicht in besonderem Maße funktionsbestimmend sind. Eine Visualisierung als Graph oder Algorithmus ist daher ausreichend, wobei im Kontext heterogener Systeme auch die Darstellung räumlicher Aspekte sinnvoll sein kann, z. B. die Lage von Sensoren in Relation zu den von ihnen gemessenen Objekten.

Semantik der Modellelemente

Die Modellierung im dreidimensionalen Raum erfordert die Festlegung vieler Freiheitsgrade, insbesondere bei der Arbeit mit geometrischen oder mathematischen Grundelementen. Die Folge ist ein hoher manueller Aufwand, was gerade bei einem Werkzeug zur Unterstützung der Konzeptphase unerwünscht ist. Eine mögliche Lösung ist die Zusammenfassung von Grundelementen und Parametern zu Prinzipelementen mit einer definierten Semantik und einem standardisierten Inhalt. Auf diesen können komplexere Operationen durchgeführt werden, die mehrere, immer wieder gleiche Modellierschritte zu einer Einheit zusammenfügen. Der Modellierprozess wird somit auf eine abstraktere Ebene verlagert, die der Philosophie des technischen Prinzips besser entspricht.

Es entsteht ein Baukasten aus Prinzipelementen, der für den Anwender zwar mit gewissen Einschränkungen verbunden ist, insgesamt aber zu einer deutlichen Vereinfachung der Modellierung führt. Die im Modell enthaltenen Informationen müssen nur noch zu einem gewissen Teil vom Nutzer spezifiziert werden, die verbleibenden Fakten sind durch die Semantik impliziert.
Beispiel zur Modellierung von Koppelgetrieben: Während in einem 3D-CAD-System eine große Zahl teilweise nicht prinziprelevanter Parameter eingegeben werden muss (z. B. Volumen), erfolgt die Modellierung auf der Prinzipebene anhand einer geringen Anzahl von Positionsangaben und Vereinbarungen darüber, auf welche Weise Drehgelenke und Getriebeglieder miteinander gekoppelt werden.

- Modellierschritte in einem CAD-System für drei Einzelteile und Constraints im Zusammenbau (siehe Abbildung 4.15)

  1. Skizze 1: Rechteck $l_1 * b_1$, Sweep Höhe $h_1 \rightarrow$ Volumenkörper K_1
  2. Skizze 2: Rechteck $l_2 * b_2$, Sweep Höhe $h_2 \rightarrow$ Volumenkörper K_2
  3. Bohrung B_1 auf Oberfläche von K_1, $\varnothing d_1$
  4. Bohrung B_2 auf Oberfläche von K_2, $\varnothing d_2$
  5. Zylinder Z_1 $\varnothing d_z$, Länge l_z
  6. Assembly: K_1, K_2 und Z_1 positionieren, Oberflächen K_1 und K_2 parallel mit Abstand, von Kreise von Z_1 und B_1 konzentrisch , Kreise von Z_1 und B_2 konzentrisch

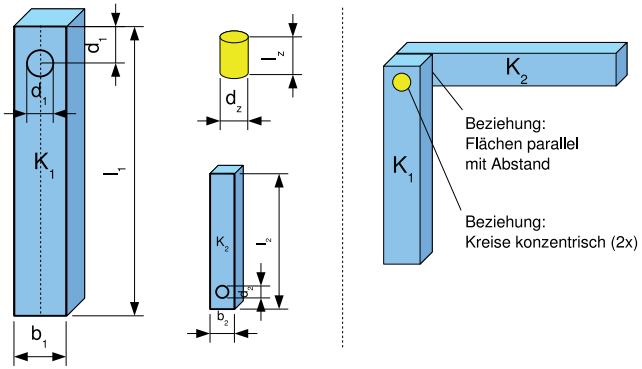


Abbildung 4.15: Modellierung von Getriebegliedern aus Volumenkörpern in einem CAD-System

- Modellierschritte im Entwurfswerkzeug für heterogene Systeme (siehe Abbildung 4.16)

  1. Binäres Getriebeglied von P_1 nach P_2 (je ein Mausklick bei Anfangs- und Endpunkt)
  2. Drehgelenk auf P_2 (automatische Verbindung mit Getriebeglied)
  3. Binäres Getriebeglied von P_2 nach P_3 (automatische Verbindung mit Drehgelenk anhand der Semantik der Koordinatenachsen und Ankerpunkte)

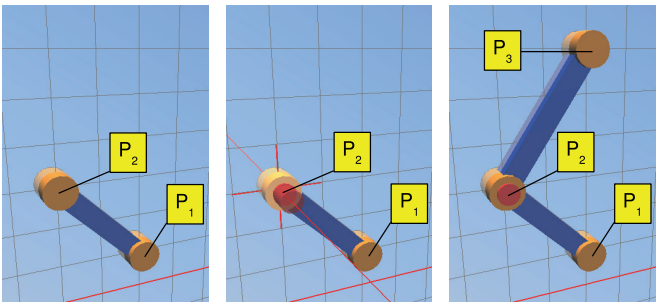


Abbildung 4.16: Modellierung von Getriebegliedern mit vorgefertigten Bauelementen in *Hesym*

Interaktive Simulation

Eine zentrale Forderung bei der Umsetzung von Interaktionskonzepten betrifft die zeitnahe Rückmeldung von Auswirkungen durchgeführter Operationen [Bow05]. Als nützlich erweist sich hierzu die Verschmelzung von Entwurfs- und Simulationswerkzeug. Bereits während der Modellierung wird automatisch ein simulierbares Modell generiert. Zu dessen Verarbeitung werden bevorzugt echtzeitfähige Simulationsverfahren eingesetzt, die zudem während des Simulationsvorgangs Nutzereingaben akzeptieren (interaktive Simulation). Der Nutzer erhält somit eine ständige Rückkopplung über die Auswirkungen seiner Modellieroperationen. Das Verstehen und Erkennen des Systemverhaltens wird durch dieses interaktive „Ausprobieren“ erleichtert.
Ein Beispiel für die enge Kopplung von Modellierung und Simulation ist in Abbildung 4.17 dargestellt. Hier wird das Licht einer Lichtquelle von einer konvexen Linse parallelisiert und anschließend über einen Planspiegel abgelenkt. Weiterhin

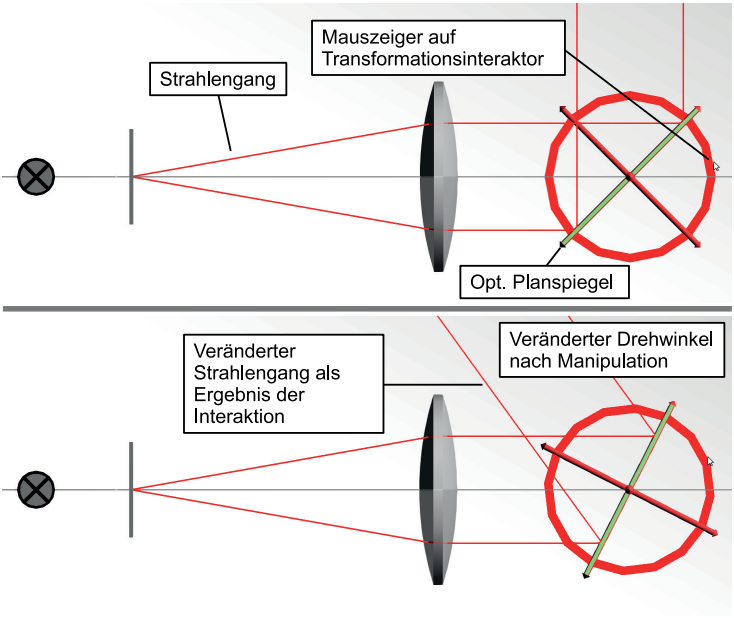


Abbildung 4.17: Interaktives „Ausprobieren“

zeigt das Bildschirmfoto einen Interaktor, der zur räumlichen Transformation des Spiegels dient. Während der Nutzer den Interaktor mit der Maus steuert und somit den Spiegel rotiert, aktualisiert die Optiksimulation fortlaufend den resultierenden Strahlengang.

Dokumentation des Interaktionszustandes

Interaktionsprozesse erfordern häufig eine Abfolge von Eingaben. Das Entwurfswerkzeug erwartet in einem bestimmten Zustand eine festgelegte Reihenfolge von Nutzeraktionen. Besonders in der Lernphase oder bei selten angewandten Interaktionen fehlt dem Anwender das Wissen über diese Abfolge. In solchen Fällen erweist es sich als sinnvoll, den Interaktionszustand zu dokumentieren. Ein einfacher Ansatz ist es, in einen Bereich der Nutzerschnittstelle eine textliche oder grafische Beschreibung darüber zu liefern, welcher Interaktor gerade aktiv ist und welche Eingaben er erwartet („Instruktor“-Fenster).
Beispiel: Der Anwender hat das Interaktionswerkzeug zur Erstellung von Rädergetrieben ausgewählt. Er soll nun zuerst den Mittelpunkt des ersten Rades positionie-

ren, den Mittelpunkt des zweiten Rades eingeben und anschließend auf der Strecke zwischen beiden Mittelpunkten das Übersetzungsverhältnis angeben. Abbildung 4.18 zeigt die entsprechenden Hinweise des Instruktorfensters.

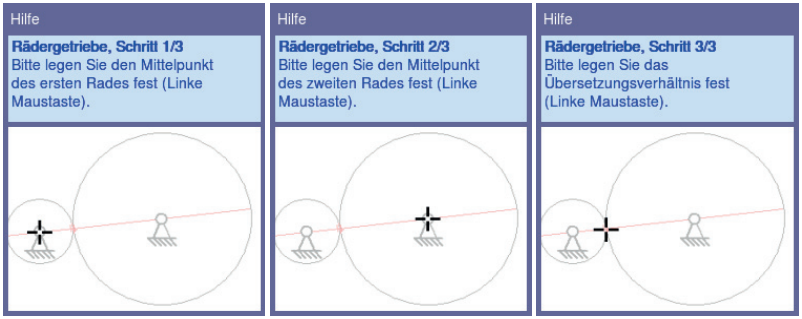


Abbildung 4.18: Ausgaben des „Instruktors“ bei der Modellierung eines Rädergetriebes

Objektbezogene Aktionen

Zur Manipulation vorhandener Objekte (Modellbestandteile) eignet sich in grafischen Umgebungen das *Select-then-Operate*-Paradigma [Brü10]. Der Nutzer wählt ein Objekt oder eine Gruppe von Objekten aus und wendet eine Operation darauf an. Einer der Vorteile dieses Verfahrens liegt darin, dass das Interaktionssystem in Abhängigkeit von der Auswahl eine (meist überschaubare) Anzahl möglicher Operationen zusammenstellen kann. Diese werden als Kontextmenü oder als Liste von Handlungsoptionen präsentiert. Der Anwender kann seine Interaktionswünsche also durch die direkte Auswahl sichtbarer Objekte benennen und muss daher nicht über ein ausgeprägtes Hintergrundwissen bezüglich möglicher Werkzeuge und Operationen sowie deren kontextabhängige Anwendbarkeit verfügen.
Da in einem Werkzeug mit modularer Architektur keine statische Benutzerschnittstelle erstellt werden kann, erweist sich dieser Ansatz auch aus softwaretechnischer Sicht als günstig. Wurden Objekte ausgewählt, wird diese Information an die *Objektbehandler* (Object Action Handler) in den Plug-Ins übermittelt. Als Reaktion liefern diese eine Liste von Aktionen zurück, die auf einzelnen Objekten oder auf deren Kombinationen ausgeführt werden können. Für jede Änderung der Auswahl wird diese Liste neu erstellt. Wählt der Nutzer eine Aktion aus, wird der zugehörige *Objektbehandler* mit ihrer Ausführung beauftragt.

Beispiel: Der Anwender positioniert zwei Räder in einer Ebene. Anschließend wählt er unterschiedliche Gruppen von Objekten.

1. Auswahl: Eines der Räder wurde markiert.
   Optionsliste: Löschen / Radius ändern / Verschieben
2. Auswahl: Beide Räder wurden markiert.
   Optionsliste: Durch Welle verbinden / Abrollbeziehung für Stirnräder erstellen / Zugmittelgetriebe erstellen

Interaktionskontexte (Interaktoren)

Im Entwurfswerkzeug für heterogene Systeme erfordert ein großer Teil der Interaktionsvorgänge die Transformation von Nutzereingaben in Modellieroperationen. Insbesondere bei komplexen räumlichen Interaktionen ist für diese Aufgabe oft eine erhebliche Menge an Applikationslogik notwendig. Es ist allerdings nicht möglich, diese in ihrer Gesamtheit fest zu kodieren, da in einem modularen Softwaresystem jederzeit neue Aspekte (Modellbestandteile, domänenspezifische Berechnungsverfahren u.ä.) hinzukommen können, deren spezielle Eigenschaften auch bei der Interaktion berücksichtigt werden müssen. Daher besteht die Notwendigkeit, zusätzliche spezialisierte Interaktionslogik in das Werkzeug einzufügen. Dies geschieht in Form sogenannter Interaktoren, die jeweils eine bestimmte Menge anwendungsabhängiger Interaktionsfunktionalität zusammenfassen (Interaktionskontext). Umfang und Ausrichtung dieser Zusatzmodule können nahezu beliebig gestaltet sein. Beispielsweise könnte ein Interaktor für die Eingabe lediglich eines einzigen Prinzipelementes zuständig sein, während ein anderer die Manipulation einer bestimmten Eigenschaft für eine ganze Gruppe ähnlicher Modellelemente übernimmt. Weitere Aspekte des Interaktorkonzeptes sind in Abschnitt 4.7.4 beschrieben.

Reduktion von Freiheitsgraden

Ein grundsätzliches Problem bei der Eingabe dreidimensionaler Strukturen an einem herkömmlichen PC-Arbeitsplatz ist das Fehlen von Eingabegeräten, die eine hinreichend große Zahl von Freiheitsgraden definieren. Beispielsweise ist die direkte Eingabe eines dreidimensionalen Punktes mit einer Computermaus (zwei Koordinaten) nicht auf direkte Weise möglich. Folglich werden Verfahren benötigt, die die Zahl der für die Eingabe notwendigen Freiheitsgrade reduzieren und somit eine eindeutige räumliche Interpretation zweidimensionaler Interaktionen erlauben. Ein bewährter Ansatz hierzu besteht in der Aufteilung der Freiheitsgrade bei der Eingabe (*separating degrees of control* [Bow05]). Eine solche Aufteilung kann durch unterschiedliche Techniken erreicht werden.

Zu den für *Hesym* relevanten Vorgehensweisen gehören:

- die Verwendung von Raster- und Führungsgeometrien (guides), mit denen Position und Orientierung einzugebender Elemente „gefangen“ werden können („snapping“),
- die Projektion zweidimensionaler Eingaben auf Hilfsgeometrien (z. B. auf Ebenen oder Kugeloberflächen),
- die Verwendung grafischer Interaktionshilfen (3D-Widgets), die eine getrennte Bearbeitung der Freiheitsgrade erlauben (z. B. *ARCBALL* [Sho92]).

Geräteabstraktion

Das Interaktionskonzept von *Hesym* sieht die Verallgemeinerung der Hardware-Eigenschaften von Eingabegeräten mittels Informationskanälen vor [SWR+02]. Obwohl hauptsächlich aus softwaretechnischen Gründen umgesetzt (siehe Abschnitt 4.8.4), bietet diese Geräteabstraktion auch aus Anwendersicht Vorteile. So erlaubt sie die beliebige Verknüpfung der Ausgangskanäle der Eingabegeräte mit den Eingangskanälen der Interaktoren entsprechend den Vorstellungen und Gewohnheiten des Anwenders.
Die Abstraktionsschnittstelle ermöglicht außerdem die Verlagerung der Funktionalität konventioneller Maus- und Tastaturinteraktionen auf eventuell vorhandene spezielle Eingabegeräte (z. B. SpaceMouse, Tracking-Systeme bei Verwendung in VR-Umgebungen) oder simulierter Geräte (z. B. Dialogfenster mit Eingabefeldern und Schaltflächen). Darüber hinaus lassen sich die von den Eingabegeräten ausgegebenen Informationen anwendungsgerecht umformen, um beispielsweise eine Skalierung oder Akkumulation der Ausgabewerte zu erreichen.

4.7.4 Interaktorkonzept im Detail

Aufbau und Funktionsweise

Interaktoren fassen Bereiche von Anwendungslogik zusammen, die in einem bestimmten Kontext Nutzereingaben in Modellieroperationen transformieren. Sie sind über festgelegte Schnittstellen in die Softwarearchitektur des Entwurfswerkzeugs eingebettet. In Abbildung 4.19 ist der Grundaufbau der Interaktorinfrastruktur dargestellt. Ein Interaktor erhält von den Eingabegeräten Ereignisse (events). Diese werden von der Applikationslogik verarbeitet. Als Reaktion wird der Zustand des Interaktors beeinflusst und gegebenenfalls finden Operationen auf dem Datenmodell statt.

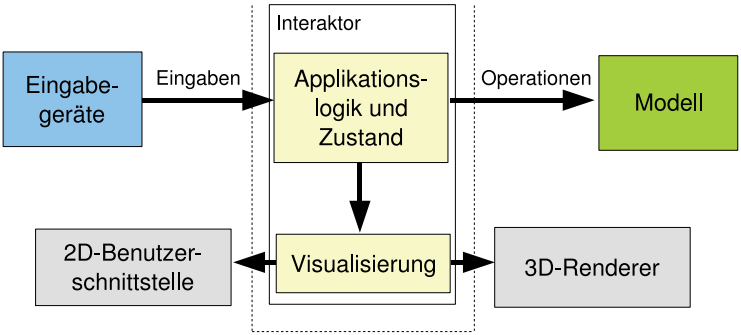


Abbildung 4.19: Grundaufbau und Umfeld eines Interaktors

Seinen Zustand stellt der Interaktor über zwei Visualisierungen dar. Er verfügt einerseits über eine zweidimensionale Repräsentation im Rahmen des Menü- und Fenstersystems des Anwendungsprogramms. Hier werden Parameter, Optionen usw. angezeigt, von denen ein Teil manipuliert werden kann (erzeugt Eingabeereignisse). Andererseits existiert eine dreidimensionale Darstellung des Interaktors parallel zur Modellvisualisierung, die mittels grafischer Eingabehilfen (Handles, „Griffe“) verfügbare Interaktionsmöglichkeiten visualisiert.

Kombination und Wiederverwendung von Interaktoren

Die Verteilung von Anwendungslogik auf unterschiedliche Interaktoren besitzt nicht nur bezüglich der Modularisierung des Entwurfswerkzeugs Vorteile. Sie fördert gleichzeitig die Wiederverwendung häufig benötigter Funktionalitäten, da diese nicht in jedem Interaktor neu implementiert werden müssen. Dies setzt allerdings voraus, dass ein gegenseitiger Aufruf von Interaktoren möglich ist.
Das Entwurfswerkzeug für heterogene Systeme definiert Schnittstellen zur Über- und Rückgabe des Kontrollflusses zwischen Interaktoren sowie zu deren anwendungsabhängiger Konfiguration und Übergabe von Zustandsinformationen. Jeder Interaktor kann sowohl Dienstanbieter als auch Dienstnutzer sein. Dieses Konzept stellt eine Weiterentwicklung des in [FKB03] vorgeschlagenen Modells dar.
Abbildung 4.20 demonstriert den gegenseitigen Aufruf von Interaktoren am Beispiel der Eingabe eines binären Getriebegliedes unter Verwendung des Interaktors *LinkCreator*. Zur Positionierung und Ausrichtung des Getriebegliedes benötigt *LinkCreator* zwei Koordinatensysteme, die er von einem weiteren Interaktor, *CoordinateSequence*, anfordert. Dieser bietet als Dienst die Ermittlung einer beliebigen

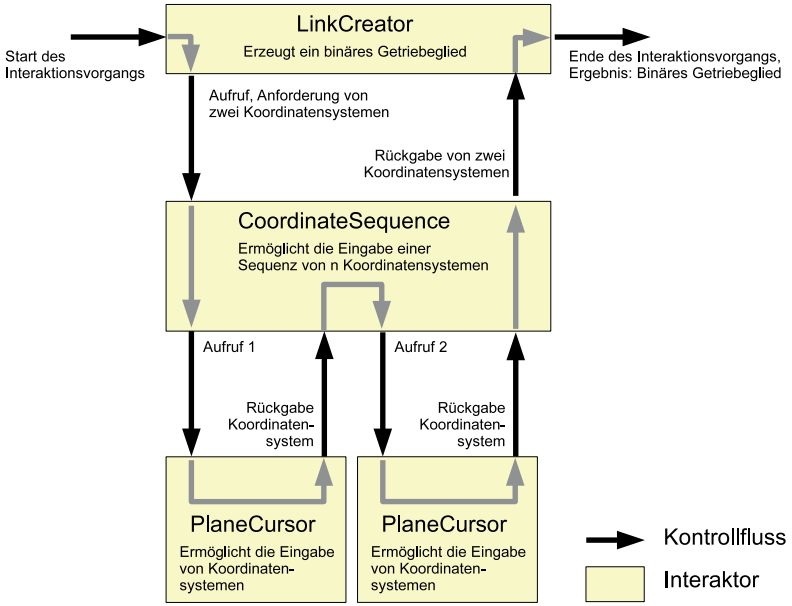


Abbildung 4.20: Beispiel für die Wiederverwendung von Interaktoren

Anzahl von Koordinatensystemen an. *CoordinateSequence* selbst greift wiederum auf andere Interaktoren zurück, um jedes einzelne Koordinatensystem einzugeben oder zu konstruieren. Im Beispiel gibt er den Kontrollfluss an *PlaneCursor* weiter, der durch Projektion des Mauscursors auf eine Raumebene ein Koordinatensystem bestimmt und an *CoordinateSequence* zurückgibt.
Dieser Schritt wird wiederholt, um das zweite Koordinatensystem zu erhalten. Nach der zweiten Rückgabe des Kontrollflusses an *CoordinateSequence* meldet dieser beide Koordinatensysteme an *LinkCreator*, der zwischen ihnen das Getriebeglied positioniert und damit den Interaktionsvorgang abschließt. Sollen weitere Interaktoren implementiert werden, so können diese auf die existierenden Bausteine zurückgreifen. So kann ein Interaktor zur Eingabe von Polygonen ebenfalls *CoordinateSequence* verwenden, während ein Interaktor zur Erzeugung eines Drehgelenks dessen Lage direkt mit *PlaneCursor* bestimmt.
Das Beispiel zeigt, wie unter Verwendung einfacher Interaktoren komplexe Interaktionsvorgänge realisiert werden können. Aufgrund der Spezialisierung der einzelnen Interaktoren bleibt deren Programmlogik überschaubar und beschränkt sich in vielen Fällen auf die Implementierung der Ablaufsteuerung.

4.7.5 Visualisierung von Modellelementen

Die Visualisierung der modellierten Lösung ist ein unverzichtbarer Bestandteil der Interaktion zwischen dem Entwurfswerkzeug und dem Nutzer. Die Lösungselemente selbst sind nicht sichtbar, da sie lediglich die mathematische Repräsentation eines Konzeptes darstellen. Daher muss eine grafische Darstellung zur Veranschaulichung dieser Konzepte vorgenommen werden. Jedes Modell (und damit jedes Modellelement) kann auf unterschiedliche Arten visualisiert werden, z. B. um Eigenschaften hervorzuheben oder einem bestimmten Darstellungsstil zu genügen. Abbildung 4.21 zeigt unterschiedliche Varianten eines Drehgelenkes, das zwei Getriebeglieder verbindet.

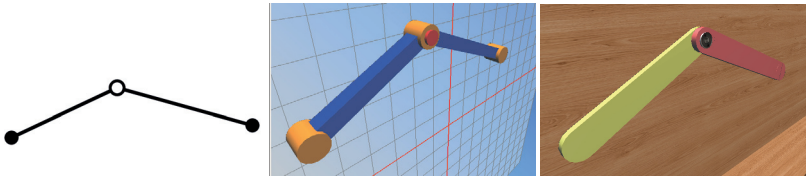


Abbildung 4.21: Unterschiedliche Visualisierungsstile von Modellelementen

Die Darstellung als ebenes technisches Prinzip (links) besitzt den Vorteil, dass hier eine weithin verbreitete und bekannte Symbolik zum Einsatz kommt. Weniger geeignet ist diese Variante für räumliche Anordnungen, da es bei unterschiedlichen Betrachtungswinkeln zu unübersichtlichen oder mehrdeutigen Darstellungen kommen kann. In diesem Falle eignet sich eine dreidimensionale Symbolik (mittlere Abbildung). Die dabei eingesetzten „Scheinvolumen“ sind für das technische Prinzip zwar irrelevant, unterstützen aber das räumliche Vorstellungsvermögen sehr. Der rechte Teil der Abbildung zeigt eine Visualisierung als einfaches Gestaltmodell, die an den Stil getriebetechnischer Demonstrationsmodelle angelehnt ist und z. B. für die Erzeugung von Animationen innerhalb der Digitalen Mechanismen- und Getriebebibliothek DMG-Lib Verwendung findet [DMG13].
Vor dem Hintergrund der vielseitigen Visualisierungsvarianten erscheint die Integration der grafischen Repräsentation in das Prinzipelement nicht sinnvoll. Zweckmäßiger ist eine Entkopplung entsprechend dem in Abschnitt 4.4 vorgestellten Datenmodell. Visualisierungen werden in Erweiterungselemente (Visualisierungselemente, Abbildung 4.22) ausgelagert, existieren parallel und können aufgabenabhängig umgeschaltet werden. Es ist somit möglich, weitere zweckangepasste Visualisierungen zu erstellen, ohne das visualisierte Prinzipelement oder bereits existierende Visualisierungen anpassen oder verändern zu müssen.

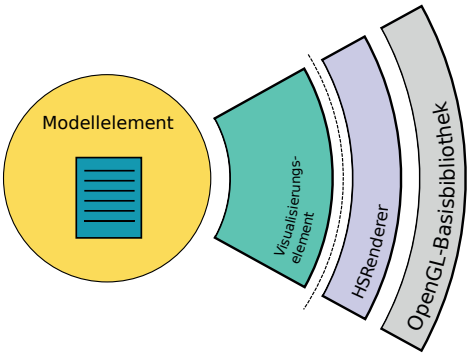


Abbildung 4.22: Visualisierungselement als Brücke zwischen Modellelement und Renderer

4.7.6 Visualisierungselemente

Die Aufgabe eines Visualisierungselementes besteht in der Transformation der Informationen des Prinzipelementes in eine grafische Repräsentation, bzw. in eine Reihe grafischer Darstellungsmittel. Es fungiert als Bindeglied zwischen dem Modellelement und einer Visualisierungsumgebung, z. B. einer Grafikbibliothek, die einen Szenengraphen zur Darstellung nutzt. Ein Visualisierungselement kann auf unterschiedliche Weise realisiert werden, wobei einige Gemeinsamkeiten existieren. Der Grundaufbau ist in Abbildung 4.23 dargestellt.

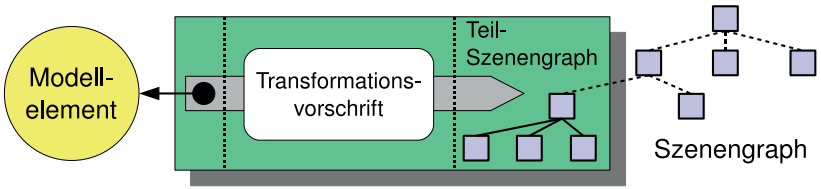


Abbildung 4.23: Aufbau eines Visualisierungselementes bei Verwendung eines Szenengraphen

Visualisierungselemente besitzen folgende Eigenschaften:

- Referenz der visualisierten Elemente – Ein Visualisierungselement ist einem oder mehreren zu visualisierenden Prinzipelementen zugeordnet.

- Transformationsvorschrift – Es existiert eine Vorschrift zur Umsetzung der Parameter des Prinzipelementes auf die Parameter der grafischen Darstellungsmittel. Diese meist einfachen Berechnungen werden entweder fest kodiert oder in Form von Regeln oder Skriptcode angegeben.

- Zugriff auf eine Grafikschnittstelle – Die berechneten grafischen Darstellungsmittel werden über eine Schnittstelle ausgegeben. Dabei kann es sich um eine Grafikbibliothek auf niedriger Ebene (z. B. direktes Zeichnen in OpenGL) oder um eine komplexere Grafikschnittstelle (z. B. Manipulation eines Szenengraphen) handeln. Auch der Fernzugriff über eine Netzwerkschnittstelle, beispielsweise auf den Szenengraphen in einem VR-System, kann realisiert werden.

- Globale Eigenschaften – Stileigenschaften oder Visualisierungsparameter.

Änderungen am Prinzipelement können Änderungen der Visualisierung zur Folge haben. Abhängig vom Umfang der Änderung werden unterschiedliche Teile der Transformationsvorschrift ausgeführt. Die möglichen Auswirkungen reichen von einfachen Transformationsänderungen bis hin zur Auswahl und Initialisierung anderer oder zusätzlicher grafischer Beschreibungselemente.

4.7.7 Interaktionsbeispiel

Das Zusammenwirken der einzeln beschriebenen Teilkonzepte zur Visualisierung und Interaktion soll anhand eines Beispiels aus der Getriebetechnik demonstriert werden. Bei dem dafür ausgewählten Entwurfsobjekt handelt es sich um eine räumliche Schubkurbel, deren Ausführung sich an dem in Abbildung 4.24 dargestellten körperlichen Modell orientiert, wobei allerdings auf die Übertragung der farblichen Kodierung der Vorlage (Antrieb, Abtrieb usw.) verzichtet wird. Die folgenden Absätze und Abbildungen geben die notwendigen Interaktionsschritte in der Reihenfolge ihres Auftretens während der Modellierung wieder.
Die Eingabe des Modells beginnt mit dem Drehgelenk der Antriebskurbel, für das zunächst eine Referenzebene gewählt wird. Anschließend aktiviert der Nutzer den Interaktor zur Erzeugung von Drehgelenken, der wiederum den Interaktor Ebenen-Cursor aufruft, um eine Raumposition zu erhalten, die aus der Projektion des Mauszeigers auf die Referenzebene bestimmt wird. Ist die Position gewählt, fügt der Drehgelenk-Interaktor dem Modell ein Modellelement Drehgelenk hinzu. Zusätzlich erfolgt die automatische Erzeugung einer entsprechenden Visualisierung (Abbildung 4.25).
Der Vorlage entsprechend soll das Drehgelenk der Kurbel gestellfest ausgeführt werden. Daher wählt der Nutzer das visualisierte Drehgelenk aus und ruft das

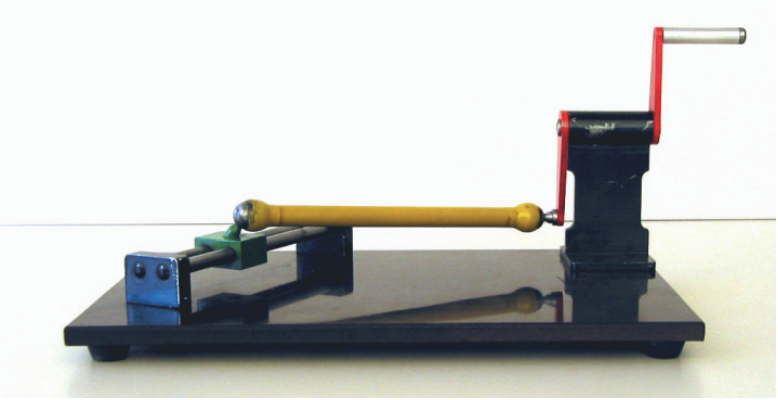


Abbildung 4.24: Körperliches Modell einer räumlichen Schubkurbel, Quelle: Getriebesammlung der Technischen Universität Dresden, Modell N7

Kontextmenü auf. In diesem führt das Entwurfswerkzeug alle objektbezogenen Aktionen auf, die für das ausgewählte Objekt von den Plug-Ins zur Verfügung gestellt werden. Da das Drehgelenk noch nicht gestellfest ist, bietet das Mechanik-Modul die Aktion „Fixieren“ an. Nach Ausführung dieser Aktion passt sich die Visualisierung des nun gestellfesten Drehgelenks an (Abbildung 4.26).
Anschließend soll das mit der Kurbel verbundene Kugelgelenk in der bereits existierenden Referenzebene relativ zum Drehgelenk positioniert werden. Dazu aktiviert der Nutzer den Interaktor zur Erzeugung von Kugelgelenken, der wiederum auf den Ebenen-Cursor zurückgreift. In Abhängigkeit von seiner Position zeigt der Cursor entsprechende Führungshilfslinien (Guides, gestrichelte Linie in Abbildung 4.27 links) an, die ein Fangen (Snapping) und somit eine exakte Relativausrichtung an bereits vorliegenden Modellelementen ermöglicht. Mit diesem Hilfsmittel gelingt beispielsweise die Positionierung des Kugelgelenkes auf einer vom Drehgelenk ausgehenden, horizontalen Linie (Abbildung 4.27 rechts). Bei Bedarf kann außerdem ein Abstandsmaß des Cursors zu anderen Modellelementen angezeigt werden.
Zur Eingabe des Schubgelenks muss zunächst eine weitere Hilfsebene spezifiziert werden. Zu diesem Zweck kommt der Ebenenauswahl-Interaktor zum Einsatz. Mit diesem erfolgt die Auswahl eines Modellelementes, aus dessen lokalen Koordinatensystem anschließend eine Ebene abgeleitet wird. In Abbildung 4.28 handelt es sich dabei um die X-Z-Ebene des Drehgelenkes, die in negativer Y-Richtung verschoben und mit dem Mauszeiger auf die benötigte Größe aufgezogen wird. Diese Operationen geschehen in direkter Interaktion mit dem 3D-Modell, ohne

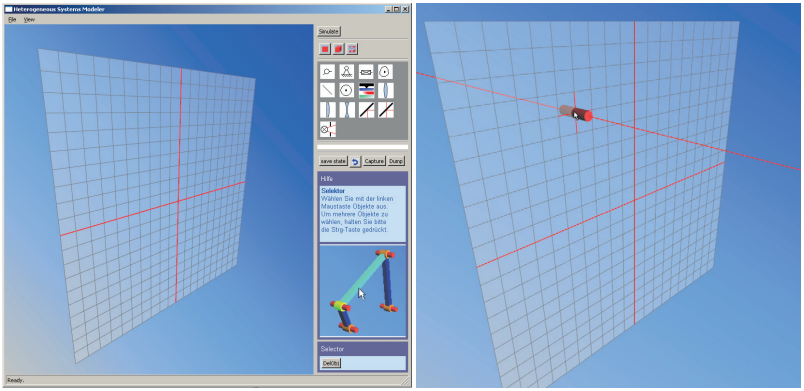


Abbildung 4.25: Positionierung des Drehgelenks auf einer Referenzebene

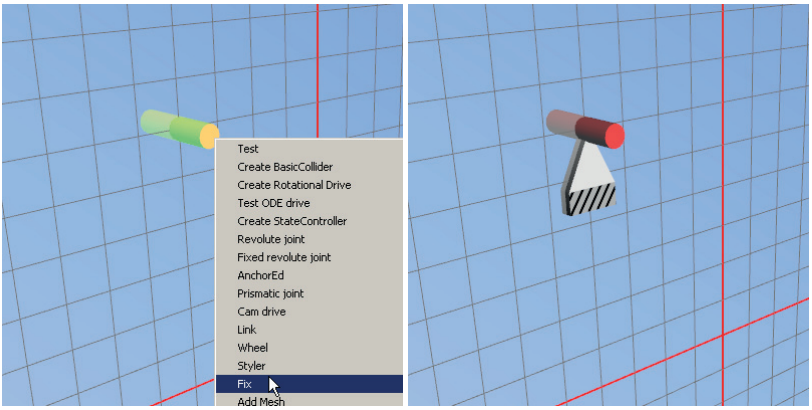


Abbildung 4.26: Umwandlung des Drehgelenks in ein gestellfestes Lager

dass dazu auf die in CAD-Software verbreiteten Baumdarstellungen des Modells oder sonstige Menüs zurückgegriffen werden muss.
Wie schon bei der Erzeugung von Dreh- und Kugelgelenk greift auch der Interaktor zur Erstellung von Schubgelenken auf den Ebenen-Cursor zurück, um drei Raumpositionen zu ermitteln (Start- und Endpunkte der Führung und die Lage des Gleitsteins, Abbildung 4.29). Der Cursor wird jeweils auf die Ebene projiziert, auf die der Mauszeiger gerichtet ist. Ein explizites Umschalten ist nicht notwendig. Dies ermöglicht eine zügige Arbeitsweise und bedeutet darüber hinaus, dass Start- und

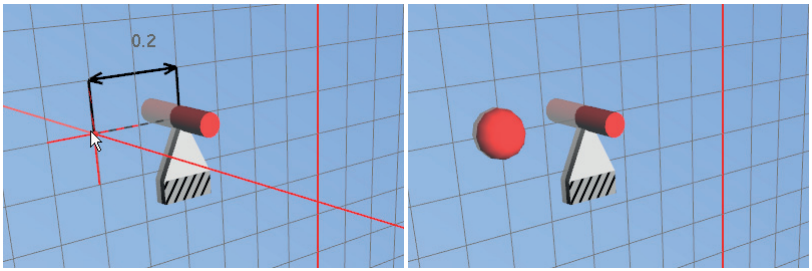


Abbildung 4.27: Positionierung des Kugelgelenks relativ zum Drehgelenk mittels Führungshilfslinien

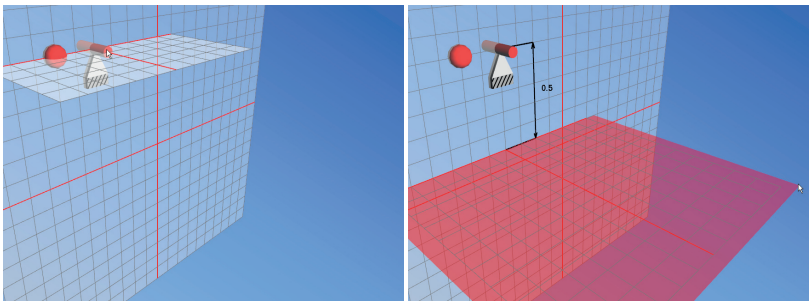


Abbildung 4.28: Ableitung einer Hilfebene aus dem Koordinatensystem des Drehgelenks

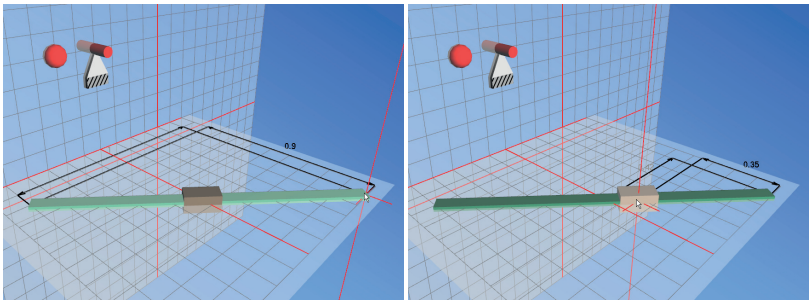


Abbildung 4.29: Eingabe des Schubgelenkes

Endpunkte der Schubgelenkführung auf unterschiedlichen Ebenen liegen könnten. Um den Nutzer über den Verlauf des Interaktionsvorgangs zu informieren und das jeweilige Ziel der drei aufeinanderfolgenden Anwendungen des Ebenen-Cursors zu verdeutlichen, dokumentiert der Interaktor seinen aktuellen Zustand einschließlich der geforderten Eingaben in einem Hilfefenster (Abbildung 4.30). Vor allem in der Lernphase erleichtert dies die Bedienung der Software.

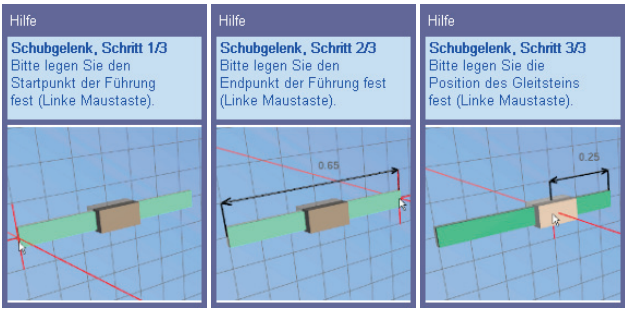


Abbildung 4.30: Dokumentation des Interaktionszustandes für den Interaktor zur Erzeugung von Schubgelenken

Das mit dem Schubgelenk verbundene Kugelgelenk soll oberhalb des Gleitsteins positioniert werden. Um eine Hilfsebene aus der lokalen X-Y-Ebene des Schubgelenks abzuleiten, findet ein weiteres Mal der Ebenenauswahl-Interaktor Verwendung. Anschließend wird das Kugelgelenk mit Hilfe der vom Gleitstein ausgehenden Führungshilfslinie positioniert. Zur Verbesserung der Übersichtlichkeit kann die Hilfsebene nun gelöscht werden. (Abbildungen 4.31 und 4.32).

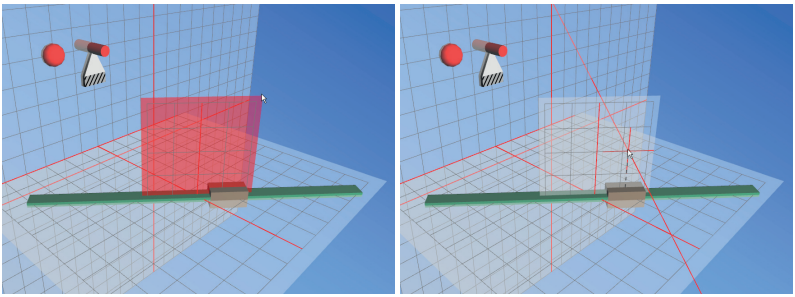


Abbildung 4.31: Ableitung einer Hilfsebene aus dem Schubgelenk

An dieser Stelle ist ein wichtiges Zwischenergebnis erreicht. Alle für das Getriebe relevanten Punkte sind über die Lage der Gelenke definiert. Die Eingabe der Getriebeglieder vereinfacht sich somit erheblich, da sie auf die Auswahl der durch sie verbundenen Gelenke reduziert wird. Dieser Umstand ist insbesondere bei der Positionierung der windschief zum globalen Koordinatensystem liegenden Koppel von Bedeutung.

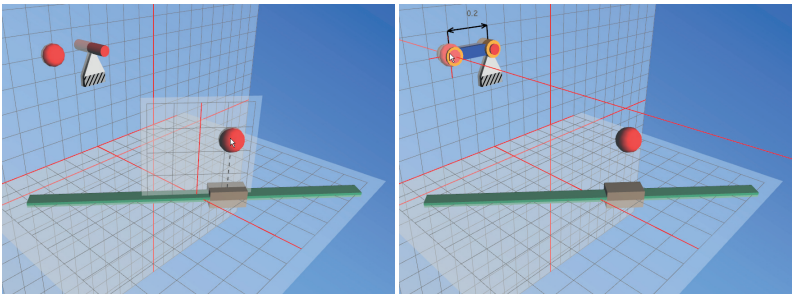


Abbildung 4.32: Positionierung des zweiten Kugelgelenkes

Der Interaktor zur Eingabe binärer Getriebeglieder erstellt entsprechende Modellelemente zwischen zwei Raumpositionen, deren Lage entweder per Ebenen-Cursor oder durch die Auswahl existierender Gelenke bestimmt werden kann. Hierbei kommt ein weiterer Aspekt des Interaktionskonzeptes, die Semantik der Modellelemente, zum Tragen. Sie ermöglicht dem Getriebeglied-Interaktor, alle benötigten Modellbestandteile, wie Ankerpunkte oder Rotationsachsen, zu identifizieren und korrekt miteinander in Beziehung zu setzen (Abbildungen 4.32 rechts und 4.33).

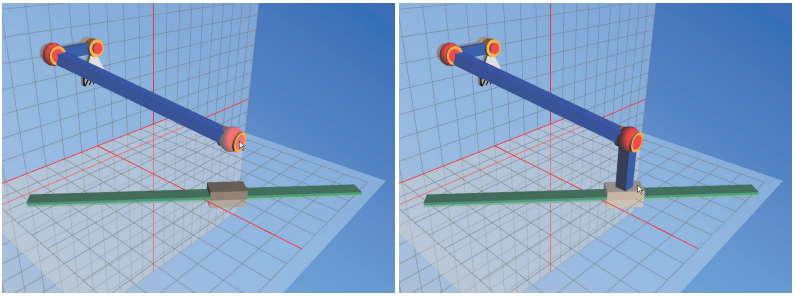


Abbildung 4.33: Eingabe der Getriebeglieder

Damit ist die interaktive Modellierung der räumlichen Schubkurbel abgeschlossen. Für die Durchführung der beschriebenen Modellierschritte benötigt ein geübter Benutzer eine Arbeitszeit von weniger als einer Minute. Anschließend kann das Verhalten des Getriebes mit Hilfe des Bewegungsinteraktors erprobt werden. Dieser ermöglicht das Greifen und Bewegen von Modellelementen mit dem Mauszeiger, während ein entsprechendes Simulationsmodul den Bewegungszustand des Modells aktualisiert (Abbildung 4.34). Die Änderung der Modellparameter ist jederzeit möglich.

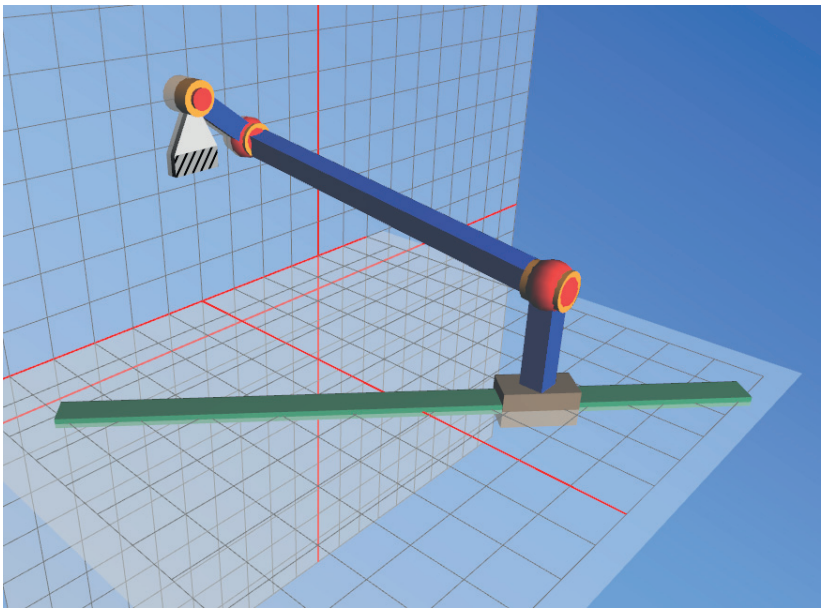


Abbildung 4.34: Veränderter Modellzustand durch interaktive Bewegungssimulation

4.8 Softwaretechnische Aspekte

Jeder Software liegt eine Architektur zugrunde. Dabei ist es möglich, dass diese implizit oder zufällig bei der Programmierung entstanden ist oder das Ergebnis eines softwaretechnischen Entwicklungsprozesses darstellt [VAC+08]. Bedenkt man den Einfluss einer Softwarearchitektur auf Aspekte wie Leistungsfähigkeit, Wartbarkeit oder die Beherrschung von Komplexität, so erscheint der Entwurf einer durchdachten, bewusst erschaffenen Architektur für den Erfolg eines Softwareprojektes unerlässlich.
Dennoch soll der vorliegende Abschnitt nicht die gesamte Architektur von *Hesym* beschreiben, sondern den Blick des Lesers auf bestimmte, für die Entwicklung eines derartigen Werkzeugs relevante Probleme und deren Lösungen lenken.
Zunächst werden grundlegende Anforderungen an die Softwareplattform und ihre Architektur aufgestellt. Anschließend erfolgt die Diskussion besonderer softwaretechnischer Aspekte, auch in Hinblick auf die in den vorangegangenen Abschnitten beschriebenen Teilkonzepte des Werkzeugs zum Entwurf heterogener Systeme.

4.8.1 Allgemeine Anforderungen an die Softwarearchitektur von Hesym

Hesym bildet kein in sich abgeschlossenes Softwareprodukt, sondern stellt als Plattform Infrastrukturen zur Schaffung domänenübergreifender Entwurfswerkzeuge zur Verfügung. Auf dieser Basis sollen anwendungsabhängige Funktionalitäten (z. B. Modellbestandteile, Berechnungs- und Visualisierungsverfahren) erstellt werden und insbesondere zur Laufzeit kombinierbar sein. Daraus ergibt sich eine Kernforderung nach Flexibilität und Erweiterbarkeit, die nicht nur durch einen modularen Aufbau der Plattform selbst, sondern auch ihrer Bestandteile (z. B. des Produktmodells oder des Interaktionssystems) zu gewährleisten ist.
Um den Anspruch eines interaktiven Simulationswerkzeugs zu erfüllen, das dem Anwender die Ergebnisse seiner Eingaben anhand direkter Rückkopplung aufzeigt, ist die Umsetzung einer echtzeitfähigen Simulationsumgebung erforderlich. Hierbei ist die Echtzeitfähigkeit sowohl in den eigentlichen Berechnungsbibliotheken zu realisieren, als auch bezüglich deren Einbindung in die Softwareplattform (Schnittstellen, Zugriff auf das Produktmodell, Visualisierung usw.).
Die in den vorangegangenen Abschnitten beschriebenen Gedanken zu Flexibilität, Erweiterbarkeit und Echtzeitfähigkeit des Entwurfswerkzeugs müssen sich auch in den Betrachtungen zur Softwaretechnik widerspiegeln. Dabei ist anzumerken, dass die Konzepte, die hier in Software abgebildet werden sollen, in Teilen bereits mit Hinblick auf die softwaretechnische Realisierbarkeit entwickelt wurden. Dies ver-

deutlicht ein weiteres Mal, dass die Betrachtung der Teilaspekte bei der Entwicklung eines Software-Entwurfswerkzeugs gesamtheitlich und nicht voneinander getrennt erfolgen darf.
Hesym entstand im Rahmen von Forschungsarbeiten. Daher war eine vollständige und detaillierte Spezifikation seiner Funktionalität und der Mittel zu deren Umsetzung im Vorfeld nicht möglich. Lösungsmöglichkeiten mussten gefunden, getestet und gegebenenfalls wieder verworfen oder mit anderen Lösungen kombiniert werden. Unter diesen Umständen erwies sich ein agiler, iterativer Entwicklungsprozess als vorteilhaft [MNK12].

4.8.2 Infrastruktur und Produktmodell

Die Forderung nach Erweiterbarkeit kann aus mehreren Blickwinkeln betrachtet werden. Einer davon betrifft die Möglichkeit der Aufgliederung von Erweiterungen. Um bestimmte Teilaspekte des Entwurfswerkzeugs getrennt voneinander erweitern zu können, wurde gemäß dem *Separation-of-Concerns*-Architekturprinzip eine Trennung von Modellbildung, Berechnung, Visualisierung und Interaktion vorgenommen. Die einzelnen Funktionalitäten werden im Rahmen separater Klassenstrukturen realisiert, die später anwendungsabhängig miteinander in Beziehung gesetzt werden (Zuordnung).
Diese Trennung wird auch durch den Einsatz des *Model-View-Controller*-Architekturmusters (MVC) getragen. MVC wurde schon vor längerer Zeit beschrieben [Ree79] und lässt aufgrund seines allgemeinen Charakters einen großen Gestaltungsspielraum bei seiner Umsetzung. Besonders im Zusammenhang mit interaktiven Simulationssystemen existieren unterschiedliche Ansichten über die Frage, ob die Simulation über einen *Controller* auf das Modell zugreift, selbst einen *Controller* darstellt oder ein eigenständiger Bestandteil der Architektur ist (z. B. *Model-Simulator-View-Controller (MSVC)*, [NH04]).
Bei der Entwicklung von *Hesym* wurden Simulationssystem und Interaktionssystem als getrennte Komponenten realisiert, die als *Controller* fungieren. Das Interaktionssystem ist in der Lage, das Produktmodell (*Model*) direkt oder indirekt über die Steuerung des (interaktiven) Simulationssystems zu beeinflussen. Das Visualisierungssystem (*View*) ist für die graphische Darstellung des Produktmodells dem Nutzer gegenüber zuständig und leitet dessen Eingaben an das Interaktionssystem weiter (Abbildung 4.35). Die Model-, View- und Controller-Komponenten sind modularisiert, so dass weite Teile ihrer Funktionalität im Rahmen von Plug-Ins erbracht und erweitert werden können.
Ein weiteres Problemfeld der Erweiterbarkeit betrifft das Hinzufügen domänenspezifischer Modellbestandteile zum Produktmodell (domänenübergreifendes Gesamt-

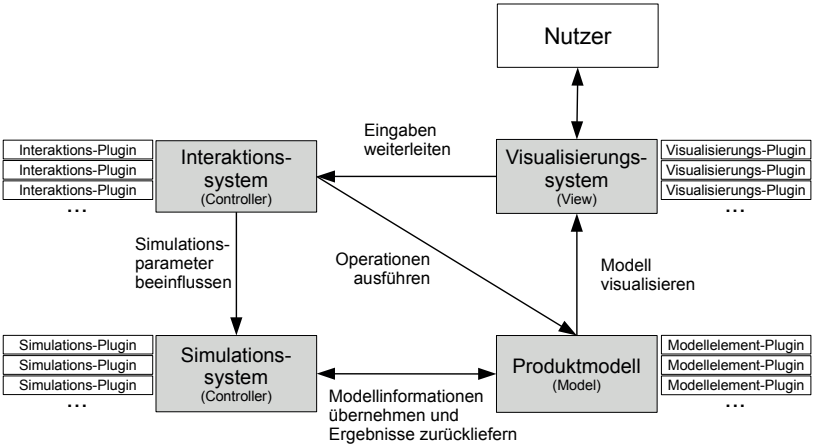


Abbildung 4.35: Grundaufbau des Entwurfswerkzeugs

modell). Das Modell muss daher Basisklassen und -schnittstellen definieren, auf denen Modellelemente aller Domänen aufbauen und die somit eine gemeinsame Behandlung (Verwaltung, Verknüpfung, Serialisierung) dieser Elemente erlauben. Die Basisklasse für Modellelemente in *Hesym* enthält

- ein eindeutiges Identifikationsmerkmal (ID),
- eine Bezeichnung,
- die Semantik,
- der Zustand (z. B. deaktiviert, gewählt),
- eine Liste zugeordneter Attribute und
- eine Liste von Beziehungen zu anderen Modellelementen.

Zur Schnittstelle der Basisklasse gehört außerdem eine Reihe von Operationen, von denen Methoden zum Erzeugen und Klonen (*Factory*-Muster [GHJV11]), zur Speicherung des Produktmodells (Serialisierung) und zum leistungssteigernden Zwischenspeichern von Objektreferenzen (Caching) hervorzuheben wären.
Ausgehend von der Basisklasse erfolgt die Definition von Modellelementen der abzubildenden Domänen mittels des Vererbungsprinzips der objektorientierten Programmierung [BME+07]. Die entstehende Vererbungshierarchie spiegelt die

Spezialisierungsbeziehungen der Modellelemente wider und geht mit der Festlegung von deren Semantik einher. Ein Beispiel ist in Abbildung 4.36 dargestellt: Aus der Basisklasse für Modellelemente geht durch Spezialisierung die Klasse *Mechanik::StarrerKörper* hervor, aus der wiederum die Klassen *Mechanik::Getriebeglied* und *Mechanik::Rad* abgeleitet werden.

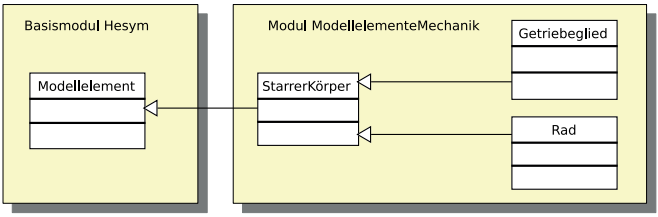


Abbildung 4.36: Bildung von Modellelementen nach dem Vererbungsprinzip

Die so definierten Klassen entsprechen den Elementen zum Aufbau des domänenübergreifenden Gesamtmodells (siehe Abschnitt 4.4.3). Zur Verarbeitung der Modellelemente im Rahmen konkreter Berechnungs- oder Visualisierungsverfahren müssen ihnen zusätzliche Informationen zugewiesen werden. Dies lässt sich mit dem Prinzip der Vererbung nur schwierig realisieren, da an dieser Stelle keine Spezialisierung in eine bestimmte Entwicklungsrichtung, sondern die Ausstattung der Elemente mit meist mehreren speziellen Eigenschaften oder Funktionalitäten gewünscht ist.
Vor diesem Hintergrund wurde bereits bei der Formulierung des Konzepts zur Modellbildung (Abschnitt 4.4.4) eine Aufteilung in Modellelemente und Erweiterungselemente vorgenommen. Aus softwaretechnischer Sicht werden Erweiterungselemente als eigenständige Klassen definiert und mit den Klassen der Modellelemente assoziiert (Abbildung 4.37). Diese Zuweisung kann auf unterschiedliche Weise erfolgen. Umfasst ein Erweiterungselement lediglich Informationen, die von mehreren anderen Erweiterungselementen genutzt werden (z. B. Definition physikalischer Eigenschaften), so erweist sich eine Einbindung in das Modellelement als Attribut als zweckmäßig (Aggregation [BME+07]). Für Erweiterungselemente, die auf Zustandsänderungen des Modellelementes reagieren sollen (z. B. Visualisierungselement), bietet sich der Einsatz des *Beobachter*-Musters (*Observer* [GHJV11]) an. Sofern Erweiterungselemente den Zustand des zugeordneten Modellelementes beeinflussen (z. B. Simulationselement), sollte eine Kopplungsschnittstelle geschaffen werden, die gegebenenfalls auch notwendige Reaktionen auf Änderungen auslöst. Auf diese Weise entstehen Modellelemente, die anwendungsabhängig um mehrere zusätzliche Aspekte erweitert werden können. Der Einsatz separater Erweiterungs-

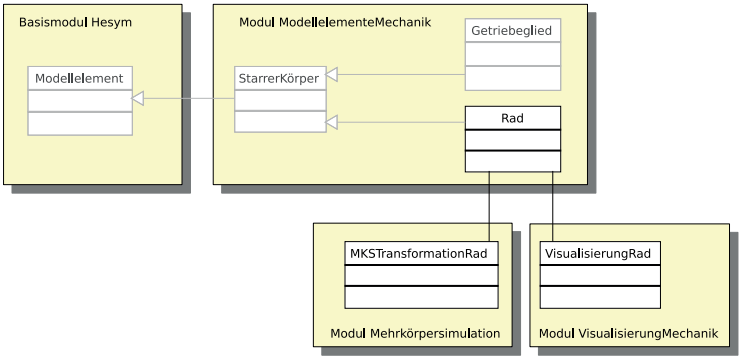


Abbildung 4.37: Assoziation zwischen Modellelementen und Erweiterungselementen

elemente ermöglicht die Zusammensetzung solcher Konstrukte zur Laufzeit, vermeidet Mehrfachvererbung und die Abhängigkeit der allgemeinen Modellelemente von speziellen Softwarebibliotheken.

4.8.3 Integration von Berechnungsverfahren und Echtzeitfähigkeit

Das domänenübergreifende Gesamtmodell bildet gleichzeitig die Grundlage für einen anderen Aspekt der Erweiterbarkeit. Bei der Einbindung von Berechnungsverfahren kann es als zentraler Kommunikationspunkt verwendet werden. Im Sinne des *Blackboard*-Architekturmusters [Nii86b, Nii86a] beziehen Berechnungsbibliotheken ihre Eingabedaten aus dem Gesamtmodell und legen Berechnungsergebnisse wieder dort ab. Die (externen) Berechnungsbibliotheken sind darüber hinaus mittels des *Adapter*-Musters [GHJV11] angebunden, das zur Vereinheitlichung der spezifischen Schnittstellen der Bibliotheken dient.
Solverspezifische Datenstrukturen werden ausschließlich in den dafür vorgesehenen Erweiterungselementen gehalten oder referenziert. Auch die Transformation der Informationen aus den Elementen des solverunabhängigen Gesamtmodells erfolgt hier. In der Regel werden die Adapter und Erweiterungselemente für ein bestimmtes Berechnungsverfahren in einem Modul (Plug-In) zusammengefasst, so dass nur dieses vom Code der Berechnungsbibliothek abhängig ist.
Um dem Anspruch eines interaktiven Simulationswerkzeugs genügen zu können, müssen Berechnungen in hoher Geschwindigkeit ausgeführt werden. Dabei handelt es sich nicht um die Forderung nach Echtzeitfähigkeit im strengeren Sinne. Es besteht lediglich das Ziel, Eingaben rechtzeitig in die Simulation einzubeziehen und

zeitnah Ergebnisse bereitzustellen, so dass die simulierte Zeit einem der Echtzeit ähnlichen Verlauf unterliegt und dem Anwender der Eindruck von Interaktivität vermittelt wird. Hieraus ergeben sich Effizienzforderungen an die Implementierung von *Hesym* und an die eingesetzten Softwarebibliotheken zur Simulation. Der diesbezügliche Einfluss auf die Bibliotheken beschränkt sich hauptsächlich auf ihre Auswahl, teilweise aber auch auf die Art ihrer Einbindung in die Infrastruktur des Entwurfswerkzeugs.
Erweiterungselemente ermöglichen dem Konzept von *Hesym* entsprechend eine bidirektionale Transformation zwischen den Informationen der Modellelemente des domänenübergreifenden Gesamtmodells und den spezialisierten Datenstrukturen zur Verwendung mit Berechnungs- und Simulationsbibliotheken.
Diese Transformation ist potenziell in jedem Simulationsschritt durchzuführen. Um den dafür notwendigen Zeit- und Speicheraufwand gering zu halten, sollten dabei so selten wie möglich Datenstrukturen erzeugt und wieder gelöscht werden. Eine strikte Unterscheidung zwischen der eigentlichen Erzeugung spezialisierter Datenstrukturen, z. B. zu Beginn eines Simulationsvorgangs oder bei strukturellen Änderungen des Modells, und der Aktualisierung der Strukturen bei reinen Zustandsänderungen, erweist sich als zweckmäßig. Auch eine Prüfung, ob überhaupt zu transformierende Änderungen vorliegen (*„dirty flag“* oder Versionszähler) sowie die Nutzung direkter Referenzen auf die verwendeten Datenstrukturen senken den Zeitbedarf der Transformation erheblich.
Diese Gedanken gelten nicht nur für Erweiterungselemente zur Simulation, sondern auch im Zusammenhang mit anderen, häufig ausgeführten Transformationen, beispielsweise bei der Visualisierung von Modellelementen mit Hilfe entsprechender Transformationselemente.

4.8.4 Interaktionssystem aus softwaretechnischer Sicht

Interaktion stellt in einem Modellierwerkzeug einen großen Teil der Anwendungslogik dar. Die Interaktionsfunktionalität ist domänenspezifisch und muss daher dem Gesamtkonzept entsprechend per Plug-Ins in das Entwurfswerkzeug eingebracht werden. Um die notwendige Flexibilität zu gewährleisten, kommen drei Teilkonzepte zum Einsatz: Abstraktion von Eingabegeräten, Interaktionskontexte und objektbezogene Aktionen.

Abstraktion von Eingabegeräten

Bei der Einbindung von Eingabegeräten in das Entwurfswerkzeug *Hesym* sind zwei Teilprobleme zu lösen. Zum einen soll die Verwendung beliebiger Eingabegeräte (z. B. Maus, Tastatur, *SpaceMouse*, *Tracker*) ermöglicht werden, zum anderen kann

die Zuordnung von Eingaben zu Operationen erst zur Laufzeit erfolgen. Dies ist notwendig, da bei der Erstellung von Plug-Ins nicht alle jemals erdenklichen Eingabegeräte bekannt sind und unabhängig voneinander arbeitende Plug-In-Entwickler sehr wahrscheinlich identische Eingabeereignisse, z. B. bestimmte Tastenkombinationen, für unterschiedliche Operationen verwenden würden, wodurch bei statischen Zuordnungen Konflikte entstünden.
Eine Lösung für beide Probleme besteht in einer Weiterentwicklung der Konzepte aus [FKB03] und [SWR+02], die eine Abstraktion und Vereinheitlichung der Eigenschaften von Eingabegeräten mittels eines Systems aus Schaltern und Kanälen (*triggers/channels*) vorschlagen. Schalter erzeugen Schaltereignisse (Aus → Ein, Ein → Aus), während Kanäle zur Übermittlung von Zahlenwerten (z. B. Ausschlag der Achse einer *SpaceMouse*) dienen. Für jedes Eingabegerät wird eine Adapterklasse definiert, die verfügbare Hardware-Eigenschaften als Schalter und Kanäle abbildet. Diese Schalter und Kanäle erhalten eindeutige Bezeichnungen innerhalb der Gerätedefinitionen (z. B. *Tastatur::Taste_A* oder *SpaceMouse::Translation_X*). Auf der anderen Seite definieren Interaktoren eine individuelle Schnittstelle aus Eingabekanälen und -schaltern, mit denen Operationen ausgelöst und parametrisiert werden (z. B. *CursorInteractor::SelectPosition*, *CameraInteractor::Yaw*).
Schalter und Kanäle der Eingabegeräte und Interaktoren können zur Laufzeit einander zugeordnet werden, wobei auch Kombinationen von Eingabeereignissen möglich sind (z. B. *Tastatur::Strg Tastatur::Taste_S* → *CursorInteractor::SelectPosition*). Die Auswertung und Übermittlung der so entstehenden dynamischen Zuordnungen übernimmt ein Ereignisverteiler (Abbildung 4.38). Dieser arbeitet anhand von Zuordnungstabellen, von denen für jeden Interaktor ein Exemplar generiert wird. Somit beschränkt sich die Suche nach Zuordnungen für eintreffende Ereignisse auf die Tabelle des gerade aktiven Interaktors.
Darüber hinaus erfolgt eine Übersetzung der Ereignisnamen in Zahlencodes, um bei der Suche zeitaufwändige Zeichenkettenanalysen (parsing) zu vermeiden. In Anbetracht großer Mengen auftretender Ereignisse (z. B. bei Mausbewegungen) erweisen sich derartige Optimierungen als vorteilhaft.

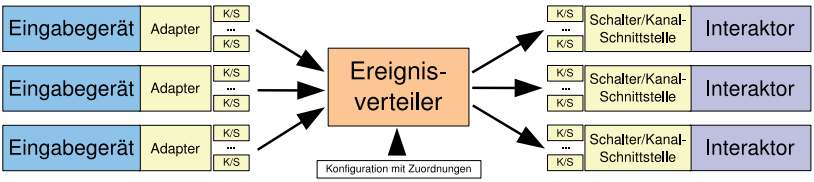


Abbildung 4.38: Schalter-/Kanal-Kommunikation zwischen Eingabegeräten und Interaktoren

Interaktoren

Im Interaktionskonzept von *Hesym* (Abschnitt 4.7.4) stellen Interaktoren Objekte dar, die einen Kontext zur Wandlung von Nutzereingaben in Modellieroperationen bilden. Sie kapseln eine bestimmte Menge anwendungsabhängiger Funktionalität hinter einer einheitlichen Schnittstelle. Interaktoren werden beim Laden der entsprechenden Plug-Ins im Interaktionssystem registriert und bei Bedarf anhand eines eindeutigen Bezeichners aktiviert (zusammengesetzt aus Plug-In-Name und Interaktorname).
Neben einer Aktivierung seitens des Nutzers ist ein gegenseitiger Aufruf von Interaktoren möglich, so dass häufig genutzte Funktionalität wiederverwendet werden kann (z. B. Selektion von Modellelementen, 3D-Cursor). Eine solche Aufrufkette lässt sich zweckmäßig als Stapel (stack) implementieren, wobei der oberste Interaktor im Stapel aktiv ist, Eingaben erhält und den Kontrollfluss besitzt. Ist die Interaktion eines Interaktors abgeschlossen, gibt er den Kontrollfluss an seinen Vorgänger im Stapel zurück. Dabei können außerdem Ergebnisse der Interaktion übermittelt werden (z. B. eingegebene Koordinaten) sowie Informationen darüber, ob die Interaktion erfolgreich abgeschlossen oder abgebrochen wurde.
Im Zusammenhang mit dem Interaktorstapel sind folgende Methoden der Interaktor-Schnittstelle von Bedeutung:

- *activate()* - teilt dem Interaktor mit, dass er aktiviert wurde,
- *reactivate()* - teilt dem Interaktor mit, dass ein von ihm aufgerufener Interaktor den Kontrollfluss zurückgegeben hat,
- *configure()* - übermittelt dem Interaktor Konfigurationsparameter,
- *handleTrigger(), handleChannel()* - übermittelt Eingabeereignisse an den Interaktor.

Neben dem beschriebenen Hauptinteraktorstapel sollte das Interaktionssystem die parallele Aktivierung spezieller Interaktoren erlauben. Ein Beispiel hierfür ist der Kamera-Interaktor, mit dem eine Steuerung der Kamera während anderer Eingaben ermöglicht wird. Um solche zusätzlichen Interaktoren mit Eingabeergebnissen zu versorgen, bietet sich eine Reservierung bestimmter Eingabegeräte für diesen Interaktor oder eine Filterung aller Ereignisse an. Der Interaktor konsumiert dazu benötigte Ereignisse und leitet die verbleibenden an den Hauptinteraktorstapel weiter.

Die Interaktor-Basisklasse umfasst weiterhin

- eine Referenz auf das Interaktionssystem, das bei Bedarf Zugriff auf das Produktmodell erlaubt,
- eine Referenz auf ein Instruktorobjekt, mit dem der Zustand der Interaktion dokumentiert wird und das dem Nutzer mit zusätzlichen Hinweisen versorgt,
- Hilfsmittel zur Definition einer Zustandsmaschine, die den Interaktionsprozess abbildet,
- Methoden zur Definition von Kanälen und Schaltern sowie
- eine Visualisierung.

Die Visualisierung eines Interaktors besteht aus zwei optionalen Teilen. Dabei handelt es sich zum einen um zweidimensionale Bestandteile der Benutzeroberfläche von *Hesym* (Dialoge, Menüs). Mit diesen können Möglichkeiten zur Konfiguration des Interaktors sowie zur Darstellung und Beeinflussung seines Zustands geschaffen werden. Getätigte Eingaben generieren Ereignisse für das Schalter-/Kanal-System. Aus softwaretechnischer Sicht sind diese Widgets vom verwendeten 2D-GUI-System unabhängige Schnittstellen und müssen für ein solches implementiert werden. Somit ist das Interaktionssystem nicht auf eine bestimmte GUI-Bibliothek festgelegt.
Der zweite Teil der Visualisierung findet im dreidimensionalen Raum neben der Darstellung des Produktmodells statt. Hierzu stellt das Interaktionssystem von *Hesym* einen Baukasten von 3D-Widgets (Handles, Griffe) zur Verfügung. Typische Ausprägungen solcher Widgets sind Pfeile, Kugeln, Quader oder Kreislinien, die entlang bestimmter Freiheitsgrade und Bewegungsbahnen im Raum transformiert werden können. Zusätzlich übernehmen die Widget-Klassen einen Teil der Auswertung der Eingaben, insbesondere zweidimensionaler Mausbewegungen, die auf dreidimensionale Bewegungen abgebildet werden. Die eigentliche Interpretation dieser Eingaben findet in der individuellen Anwendungslogik des Interaktors statt.

Objektbezogene Aktionen

Eine Besonderheit bei der Umsetzung einer Plug-In-basierten Software ist die Erstellung kontextabhängiger Menüs. Im Gegensatz zu monolithisch aufgebauten Anwendungen kann der Menüinhalt nicht statisch vorprogrammiert sein, da mit neuen Plug-Ins bis dahin unbekannte Funktionalität erscheint. Zur Lösung des Problems verfügt das Entwurfswerkzeug über eine Einrichtung, die für eine bestimmte Zusammenstellung von Modellbestandteilen alle durchführbaren objektbezogenen Aktionen sammelt.

Jedes Plug-In kann über einen sogenannten Objektaktionsbehandler verfügen. Dieser ist eine Klasse, deren Schnittstelle zur Ermittlung und Verarbeitung möglicher Aktionen dient. Diese Behandler werden mit der Methode *„createObjectActions(elements, actions)“* über die aktuelle Auswahl von Modellelementen (*elements*) informiert. Als Reaktion tragen sie die aus ihrer Sicht durchführbaren Aktionen für diese Auswahl in eine Liste ein. Ein solcher Eintrag enthält eine Verweis auf den zuständigen Behandler, die Zielobjekte, eine Bezeichnung und eine Beschreibung. Die Aktionen werden dem Nutzer präsentiert (z. B. per Kontextmenü). Wählt der Nutzer eine Aktion aus, so wird beim jeweiligen Behandler die Methode *„executeAction(action)“* aufgerufen, worauf dieser die entsprechende Operation ausführt.

4.8.5 Einbindung von Funktionalität zur Laufzeit (Plug-Ins)

Der Begriff Plug-In (to plug in, anschließen/einstecken) bezeichnet eine Softwarekomponente, die der Erweiterung eines anderen Programms dient. Plug-Ins fügen dem Hauptprogramm Funktionalität hinzu, ohne dieses selbst zu verändern. Die Einbindung solcher Softwarekomponenten kann erfolgen, während das Hauptprogramm läuft.

Zweck von Plug-Ins in Bezug auf Hesym

Eine Kernforderung an ein Entwurfswerkzeug für heterogene Systeme besteht in seiner einfachen Erweiterbarkeit. Es liegt nahe, dass im Zuge des Ausbaus einer solchen Software, z. B. bei der Einbeziehung zusätzlicher Domänen, immer wieder neue Datenstrukturen und Programmfunktionen hinzugefügt werden müssen. Auch die in Abschnitt 4.4 vorgeschlagene Beschreibung heterogener Systeme mittels erweiterbarer Modellelemente unterstreicht die Notwendigkeit einer modularen Softwarearchitektur. Als zusätzlicher Vorteil einer Plug-In-Schnittstelle sei herausgestellt, dass sie die Erweiterbarkeit der Software durch Dritte erleichtert. Es muss keine Koordination zwischen den Autoren des Hauptprogramms und des Plug-Ins erfolgen, da der Quelltext des Hauptprogramms nicht verändert wird.

Beispiele für die Verwendung von Plug-Ins in *Hesym* sind

- das Hinzufügen neuer Modellelemente,
- die Erstellung eines Dateiexporters,
- die Einbindung eines Berechnungsmoduls,
- das Hinzufügen einer Visualisierung von Modellelementen.

Aufbau der Plug-In-Schnittstelle

Das Plug-In-System definiert zwei Arten von Schnittstellen. Eine davon ist im Plug-In selbst enthalten und dient zur Abfrage und Übergabe der enthaltenen Funktionalität. Über diese Schnittstelle wird das Plug-In aufgefordert, seine Algorithmen und Datenstrukturen beim Hauptprogramm zu registrieren.
Auf der anderen Seite stellt die Hauptapplikation eine Reihe von Schnittstellen und Basisklassen zur Verfügung. Diese definieren den Rahmen, in dem das Hauptprogramm erweitert werden kann. Die Klassen des Plug-Ins sind üblicherweise von den Basisklassen der Hauptapplikation abgeleitet und werden mittels *Inversion of Control* [JF88] angesprochen (z. B. Aktivierung eines registrierten Interaktors durch die Hauptapplikation).
Die in Tabelle 4.9 aufgeführten Schnittstellen verdeutlichen die Vielseitigkeit der Erweiterungsmöglichkeiten von *Hesym*. Jedes Plug-In kann eine beliebige Untermenge dieser Schnittstellen unterstützen. Werden beispielsweise zusätzliche Modellelemente in das System eingebracht, ist es ebenfalls erforderlich, alle Funktionen und Daten bereitzustellen, die zur Simulation, Interaktion und Visualisierung dieser Modellelemente notwendig sind.

Tabelle 4.9: Von *Hesym* bereitgestellte Schnittstellen für Plug-Ins

Schnittstellentyp	Beschreibung
Modellelementschnittstelle	Verwaltung von Modellelementen
Visualisiererschnittstelle	Erweiterungselemente zur Darstellung von Modellelementen
Solver-Schnittstelle	Berechnungsmodule oder Erweiterungselemente für solverspezifische Modelltransformationen
Interaktor-Schnittstelle	Umsetzung von GUI- und Eingabegerät-Ereignissen in Editier- oder Modellieroperationen
Objektaktionsschnittstelle	Kontextabhängige, objektbezogene Operationen
Import-/Export-Schnittstelle	Transformation des Produktmodells in oder aus Dateien
Kommunikationschnittstelle	Kopplung z. B. an CAD- oder VR-System, DMG-Lib-Lösungsdatenbank für Mechanismen

Auf welche Anzahl von Plug-Ins diese Funktionalitäten verteilt werden, ist frei wählbar. Allerdings sind bestimmte Kombinationen zweckmäßig (z. B. Visualisierung und Interaktion).

Registrierung der Funktionalität zur Laufzeit

Beim Start von *Hesym* (Hauptprogramm) durchsucht der Plug-In-Lader bestimmte Verzeichnisse nach dynamischen Bibliotheken (*DLL/Shared Object*). Aufgefundene Bibliotheken werden geladen und auf die Verfügbarkeit der Plug-In-Registrierschnittstelle geprüft (Abbildung 4.39). Ist diese Schnittstelle vorhanden, wird die Bibliothek mit der Registrierung ihrer Datentypen und Funktionalitäten beauftragt. Als Ergebnis speichert das Hauptprogramm eine Reihe von Implementatoren der in Tabelle 4.9 aufgeführten Schnittstellen. Sind alle Plug-Ins bearbeitet, wird die Applikation initialisiert, deren Umfang sich aus ihrem Grundsystem (Plattform) und den anwendungsspezifischen Funktionalitäten der Plug-Ins zusammensetzt.

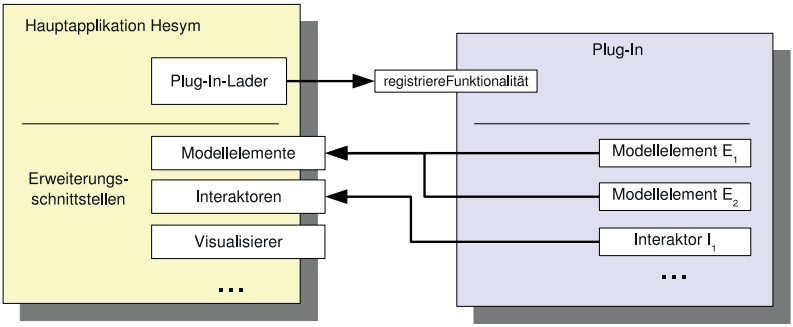


Abbildung 4.39: Plug-In-Schnittstelle von *Hesym*

Kapitel 5

Implementierung des Entwurfswerkzeugs für heterogene Systeme

5.1 Überblick

Dieses Kapitel beschreibt die Implementierung des Entwurfswerkzeugs in Form eines Softwarepakets. Es dient zur Erprobung der in den vorangegangenen Abschnitten beschriebenen Konzepte. Darüber hinaus soll im Zusammenhang mit den Beispielen in Kapitel 6 verdeutlicht werden, dass die Behandlung aller in dieser Arbeit aufgeführten Teilaspekte, sowie deren Abstimmung aufeinander, benötigt werden, um ein funktionsfähiges Werkzeug zusammenzusetzen. Zu diesem Zweck wurde eine Auswahl von Modulen und Plug-Ins erstellt, die die Umsetzung und das Zusammenspiel der wichtigsten Teilkonzepte demonstrieren. Dazu zählen die Definition von Modellelementen aus unterschiedlichen Anwendungsdomänen, die Einbindung von Berechnungsmodulen und die Erstellung entsprechender Transformationselemente, die Visualisierung von Modellelementen in unterschiedlichen Darstellungsstilen, die Schaffung eines Interaktionssystem einschließlich verschiedener Interaktoren sowie die Kombination dieser Module und Plug-Ins zu einer Gesamtanwendung. Abbildung 5.1 zeigt das entstandene Entwurfswerkzeug für heterogene Systeme.
Das Softwarepaket besteht aus 16 Modulen, die ca. 30.000 Codezeilen umfassen. Der Quellcode ist in der Programmiersprache C++ implementiert und zur Kom-

pilierung unter den Betriebssystemen Microsoft Windows und GNU/Linux vorgesehen. Aufgrund der Verwendung von standardisiertem C++ und dem Einsatz plattformübergreifender Programmbibliotheken ist eine Adaption der Software an andere Plattformen mit geringem Aufwand möglich. Lediglich die Programmteile, die die CAD-Schnittstelle betreffen, können ausschließlich unter Microsoft Windows übersetzt werden, da hier die MS-COM-Schnittstelle (*Component Object Model* [Zwi05]) zum Einsatz kommt. Dies stellt keine Einschränkung dar, da die verwendeten CAD-Pakete ohnehin nur für dieses Betriebssystem existieren und keinen zentralen Bestandteil des Entwurfswerkzeugs bilden.
Die Software setzt auf eine Reihe von Programmbibliotheken auf, welche bestimmte Grundfunktionalitäten zur Verfügung stellen (Datenstrukturen, Benutzeroberfläche usw.). Die wichtigsten von ihnen sind in Tabelle 5.1 aufgeführt.

Tabelle 5.1: Verwendete Programmbibliotheken

Bibliothek	Verwendung
OpenGL	Dreidimensionale Grafik [Shr09]
FOX Toolkit	2D-Benutzerschnittstelle (Fenster, Dialoge usw., [van09])
Xerces-C++	XML-Schnittstelle für Modellserialisierung [Xer13]
STL, boost	Hilfsmittel für Datenstrukturen [STL11, boo13]
lua	Skriptsprache für Modellaufbau, Formulierung von Regleralgorithmen u.ä., nutzerdefinierte Visualisierungen [IdC06]
ODE	*Open Dynamics Engine*, Dynamiksimulation mechanischer Elemente [Smi06]
Ficucs	Geometrischer Constraint-Solver, Kinematik, Optik [Dör11]
Raytracer	Berechnung von Strahlengängen optischer Bauelemente
masp-sdk	Import/Export von *MASP*-XML-Dateien [Sto07]

5.2 Modulstruktur

Das Entwurfswerkzeug besteht aus einer Reihe von Modulen, die sich in Basismodule zur Schaffung der Infrastruktur der Software-Plattform und domänenspezifische Erweiterungsmodule aufteilen. Die Erweiterungsmodule wurden so angelegt, dass sie für eine bestimmte Domäne jeweils Modellelemente definieren, Transformationselemente für Simulationsverfahren zur Verfügung stellen oder Visualisierung und Interaktion implementieren. Aus technischer Sicht sind beliebige andere Aufteilungen möglich (z. B. die gesamte Funktionalität einer Domäne in einem einzigen

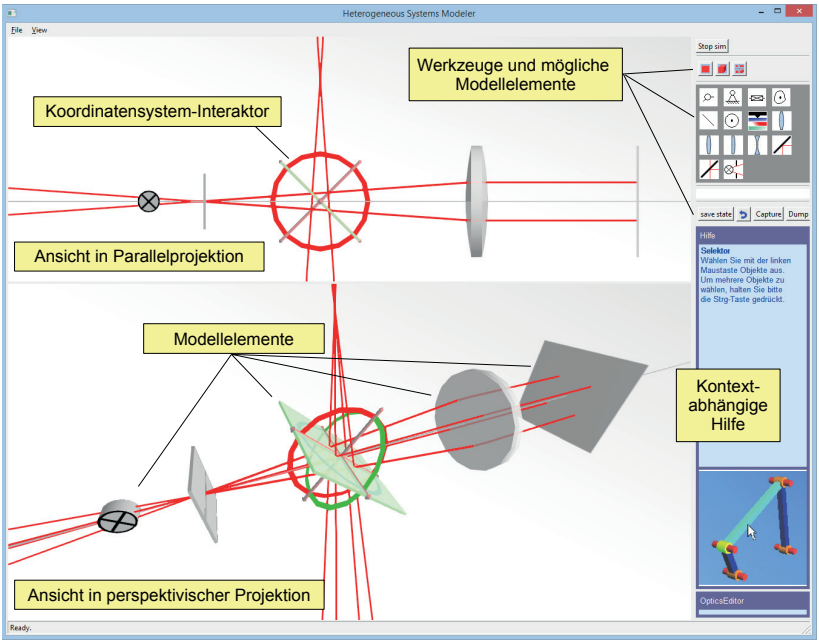


Abbildung 5.1: Entwurfswerkzeug für heterogene Systeme *Hesym*

Modul). Die gewählte Variante ist aus Anwendungssicht allerdings sinnvoll, da zu einem Satz Modellelemente jeweils mehrere Simulationsverfahren (jeweils ein Modul) oder Visualisierungsstile existieren können. Weiterhin lassen sich so auf Basis der Datenstrukturmodule auch nicht-grafische Anwendungen schaffen, wie beispielsweise Konvertierprogramme als Serverprozesse.

Die umgesetzte Modulstruktur ist in Abbildung 5.2 dargestellt. Die Bezeichnungen der Module folgen größtenteils einer Namenskonvention. Sie beginnen mit den Buchstaben „HS“ für *Hesym*, gefolgt von einem Bezeichner der Anwendungsdomäne und einer Abkürzung mit dem Hinweis auf die enthaltene Funktionalität. Das Modul zur Simulation optischer Komponenten mit einem Raytracer trägt folglich den Namen *HSOpticsRT*, während *HSAutomationVI* Funktionalität zur Visualisierung und Interaktion automatisierungstechnischer Modellbestandteile liefert.

Die folgenden Abschnitte beschreiben den Funktionsumfang der implementierten Module.

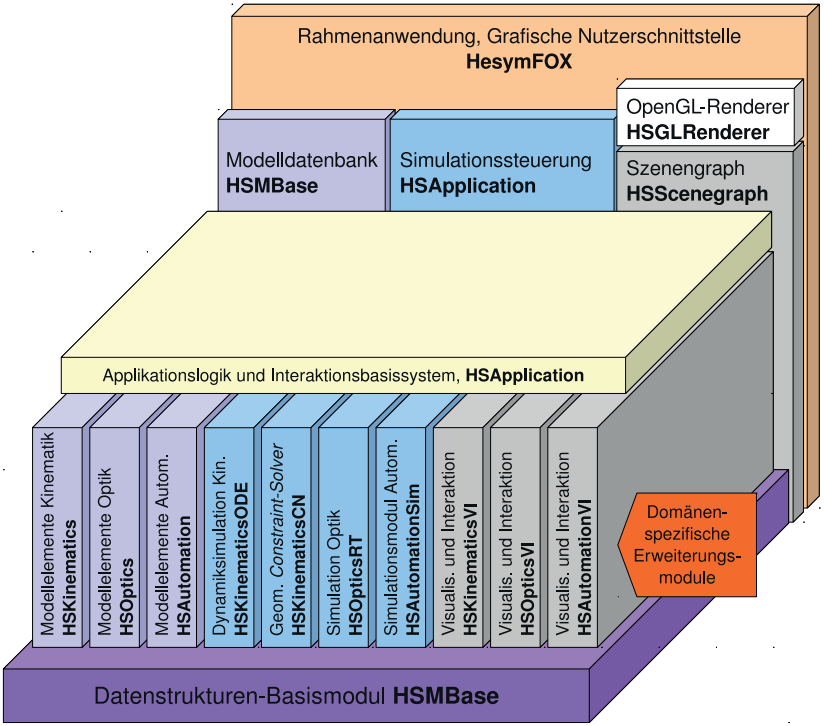


Abbildung 5.2: Modulstruktur des Entwurfswerkzeugs mit Basis- und Erweiterungsmodulen

5.2.1 Basismodule

Die Basismodule stellen die grundlegende Infrastruktur für das Entwurfswerkzeug und die domänenspezifischen Module zur Verfügung.

Basismodul für das Produktmodell HSMBase

HSMBase umfasst die Basisklassen für die Definition des Produktmodells. Mit ihrer Hilfe können sowohl die Elemente des domänenübergreifenden Gesamtmodells als auch Erweiterungselemente implementiert werden. Neben der Modelldatenbank (Modellelemente, Attribute, Serialisierung und Selektion) werden Schnittstellen für Erweiterungsmodule definiert (z. B. Berechnungs- oder Visualisierungsmodule).

Hauptapplikationsmodul HSApplication

Die Aufgabe von *HSApplication* ist es, die grundlegende Applikationslogik zu implementieren. Das Modul organisiert das Laden der Plug-Ins und die Registrierung der darin enthaltenen Funktionalitäten. Weiterhin werden hier die Grundlagen für die Abstraktion der Eingabegeräte mittels eines Trigger-Channel-Konzepts gelegt. Darauf aufbauend definiert *HSApplication* Basisklassen für das Interaktionssystem (*InteractionManager*, Interaktoren, Handles) und selektionsabhängige Objektaktionen. Außerdem stellt das Modul eine Reihe von Basisinteraktoren zur Verfügung (z. B. Kamerasteuerung, Selektor, 3D-Cursor, *PolygonEditor* usw.).

Szenengraph HSSceneGraph

Das Modul *HSSceneGraph* definiert einen einfachen Szenengraph und alle für die Visualisierung verwendbaren grafischen Primitive (z. B. Zylinder, Dreiecksnetz, Pfeil, Linie) sowie die Basisklasse für Visualisierungsobjekte von Modellelementen (siehe Schalenmodell in Abbildung 4.7). Jedes Visualisierungsobjekt besitzt einen (Miniatur-)Szenengraphen mit einer globalen Transformation als Wurzelelement. Im Sinne der unkomplizierten Anwendung existieren auch komplexere Primitive, wie beispielsweise Maßangaben. Folgende Elemente stehen zur Verfügung:

- Kugel/Kugelsegment
- Ebene
- Quader
- Zylinder (konfigurierbar mit und ohne Deck- bzw. Mantelfläche)
- Dreidimensionale Linien
- 2D-Primitive (Linien, Kreise, Rechtecke)
- Dreiecksnetz
- 3D-Pfeil (mit konfigurierbaren Pfeilspitzen)
- Maßangabe (Winkel, Distanz)

Dem Grundgedanken des Entwurfswerkzeugs entsprechend muss *HSSceneGraph* nicht verwendet werden. Es ist möglich, Visualisierungselemente auf der Basis einer beliebigen Grafikbibliothek zu erstellen.

Visualisierungsmodul HSGLRenderer

HSGLRenderer verwendet die Grafikbibliothek *OpenGL*, um die in *HSSceneGraph* definierten grafischen Elemente darzustellen.

Dialogoberfläche HesymFOX

HesymFOX bettet *HSApplication* und *HSRenderer* unter Verwendung der GUI-Bibliothek *FOX Toolkit* in die Benutzeroberfläche des jeweiligen Betriebssystems ein. Es erzeugt Menüs, Dialoge und die Fenster für die OpenGL-Ansichten des Renderers. Gleichzeitig dient es zum Start des Entwurfswerkzeugs (ausführbares Programm als Rahmen für die übrigen Module).

Hilfsgeometriemodul HSMetaGeom

HSMetaGeom beschreibt Geometrien, die nicht zum modellierten Produkt gehören, sondern unterstützenden Charakter besitzen. Beispiele sind Linien zur Darstellung von Maßangaben oder eine Hilfsebene zur räumlichen Eingabe mit zweidimensionalen Interaktionsgeräten. Das dazugehörige Modul *HSMetaGeomVI* enthält Visualisierungselemente für die in *HSMetaGeom* definierten Geometrien.

5.2.2 Domänenspezifische Module

Domänenmodule fügen dem Entwurfswerkzeug Modellelemente, Simulationsverfahren (einschließlich Transformationselemente), Interaktoren oder Visualisierungen für eine bestimmte Domäne hinzu. Dieser Abschnitt gibt einen Überblick über die implementierten Module für die im Rahmen dieser Arbeit ausgewählten Beispieldomänen Mechanik, Optik und Automatisierungstechnik.

Module für Bewegungssysteme

HSKinematics definiert mechanische Modellelemente des domänenübergreifenden Gesamtmodells und deren Beziehungen. Zur Verfügung stehen

- Körper / Getriebeglied,
- Rad,
- Kurvenscheibe,
- Drehgelenk (*Revolute Joint*),
- Kugelgelenk (*Spherical Joint*),
- Schubgelenk (*Prismatic Joint*),
- Rädergetriebe (Abrollbeziehung) und
- Zahnstangengetriebe.

Das Modul *HSKinematicsVI* stellt Visualisierungselemente und Interaktoren für die Modellelemente aus *HSKinematics* zur Verfügung. *HSKinematicsCN* enthält solverspezifische Transformationselemente für den geometrischen Constraint-Solver. Sie dienen zur kinematischen Bewegungssimulation der oben genannten Modellelemente der Mechanik. Die Transformation der in *HSKinematics* definierten mechanischen Modellelemente in Erweiterungselemente zur Verwendung mit der *Open Dynamics Engine* erfolgt in *HSKinematicsODE*. Mit diesem Modul kann das dynamische Verhalten von Bewegungssystemen untersucht werden. In Abbildung 5.3 sind Beispiele für visualisierte mechanische Modellelemente dargestellt.

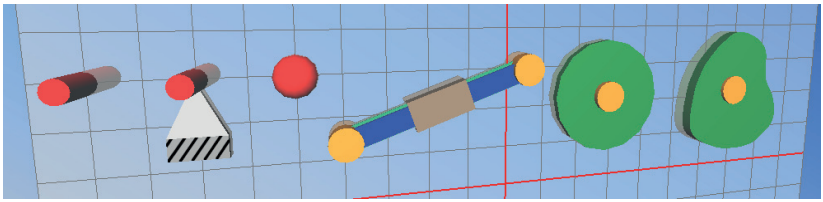


Abbildung 5.3: Beispiele für mechanische Modellelemente (Drehgelenk, gestellfestes Drehgelenk, Kugelgelenk, Getriebeglied mit Schubgelenk, Rad, Kurvenscheibe)

Module für optische Systemkomponenten

Das Modul *HSOptics* enthält die Klassen für optische Bauelemente wie

- sphärische und planare Spiegel,
- sphärische Linsen (Kombinationen aus konvexen, konkaven und planaren Oberflächen),
- Prismen, Planplatten und optische Keile,
- Licht-/Laserquellen,
- lichtempfindliche Sensoren.

In *HSOpticsVI* sind Visualisierungselemente und Interaktoren für optische Bauelemente definiert (Beispiele in Abbildung 5.4). *HSOpticsRT* transformiert die Modellelemente aus *HSOptics* in das solverspezifische Modell des *Raytracing*-Moduls.

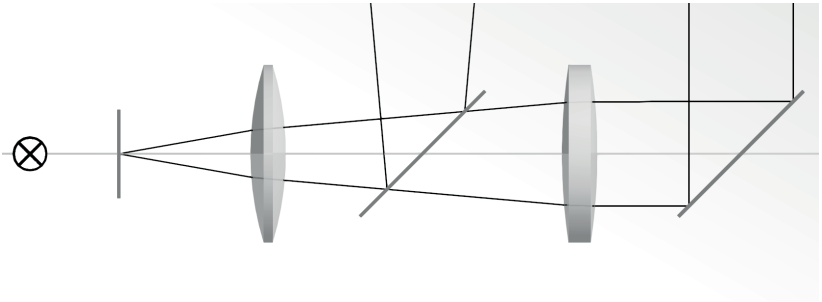


Abbildung 5.4: Beispiele für optische Komponenten (Lichtquelle, Rechteckblende, Linse, halbtransparenter Spiegel, Linse, Spiegel)

Module für Automatisierungstechnik

Die Module *HSAutomation*, *HSAutomationVI* und *HSAutomationSim* umfassen Modelldefinition, Interaktions-, Visualisierungs- und Simulationselemente für folgende automatisierungstechnische Komponenten:

- Sensoren,
- PID-Regler,
- Zustandsregler,
- Regleralgorithmus (skriptbasiert),
- rotatorische und translatorische Antriebe.

Beispiele für solche Komponenten sind in Abbildung 5.5 dargestellt.

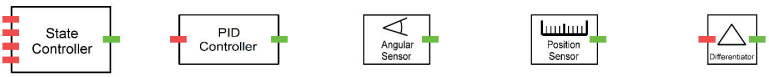


Abbildung 5.5: Beispiele für automatisierungstechnische Komponenten (Zustandsregler, PID-Regler, Winkelsensor, Lagesensor, Differenzierblock)

5.3 Interaktoren

In den oben beschriebenen Interaktionsmodulen wurden folgende Interaktoren implementiert:

- CameraInteractor – dient zur Festlegung der räumlichen Ansicht des 3D-Modells.
- PlaneCursor – Eingabe von Punkten im Raum mittels zweidimensionaler Interaktionsgeräte (z. B. Maus) und einer Hilfsebene.
- CoordinateSequence - Eingabe einer Folge räumlicher Punkte.
- PolygonEditor – Modellierung von Polygonen auf einer Ebene.
- PlaneCreator – Positionierung von Hilfsebenen.
- LinkCreator – Erzeugung von Getriebegliedern.
- CamCreator – Erzeugung von Kurvenscheiben.
- WheelCreator – Erzeugung von Rädern.
- PrismaticJointCreator – Erzeugung von Schubgelenken.
- RevoluteJointCreator – Erzeugung von Drehgelenken.
- SphericalJointCreator – Erzeugung von Kugelgelenken.
- AnchorEditor – Bearbeiten der Ankerpunkte von Körpern.
- OpticsCreator – Erstellung aller optischen Bauelemente des Moduls *HSOptics*.
- OpticsEditor – Interaktive Modifikation der Parameter optischer Bauelemente.
- Styler – Festlegung individueller Darstellungsparameter für Visualisierungselemente.

Interaktoren können mit sehr unterschiedlichem Funktionsumfang ausgestattet sein. Zur Demonstration des möglichen Spektrums wurden während der Implementierung unterschiedliche Varianten von Interaktoren erstellt. Diese reichen von spezialisierten Interaktoren, mit denen jeweils ein einziges Modellelement eingegeben werden kann (z. B. *WheelCreator*, *PrismaticJointCreator* usw.), bis hin zu umfangreichen Interaktoren zur Modifikation der Modellelemente eines gesamten Domänenmoduls (z. B. *OpticsEditor*).
Bei der Aufteilung der Funktionalität von Interaktoren spielen verschiedene Aspekte eine Rolle. Aus Nutzersicht sollte ein Interaktor einen plausibel abgegrenzten Aufgabenbereich umfassen. Dabei können entweder eine bestimmte Operation

für unterschiedliche Modellelemente oder unterschiedliche Operationen zu einem bestimmten Modellelement angeboten werden. Softwaretechnisch betrachtet stehen die Trennung von Zuständigkeiten zur Schaffung wartbaren Quellcodes, aber auch die Wiederverwendung implementierter Funktionalität für ähnliche Modellelemente im Vordergrund.

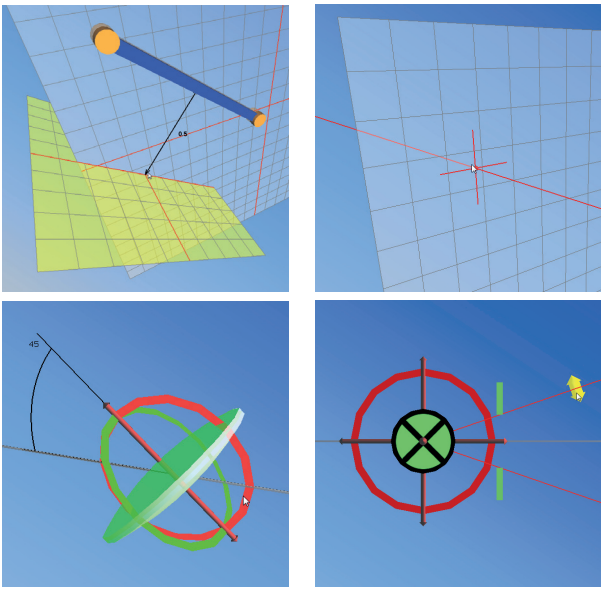


Abbildung 5.6: Beispiele für Interaktoren (Interaktor zur Definition von Hilfsebenen, Oberflächencursor, Arcball, Interaktor zur Manipulation von Lichtquellen)

5.4 Einschätzung

Mit der Implementierung des Softwarepaketes *Hesym* gelang die Umsetzung der wesentlichen Konzepte eines Werkzeugs zur Modellierung heterogener Systeme. Die betrachteten Teilaspekte zur Modellbildung, Simulation, Visualisierung und Softwarearchitektur greifen wie vorgesehen ineinander und erlauben die Demonstration der Funktionsfähigkeit des Werkzeugs im Detail und als Ganzes. Dies wird auch anhand der im folgenden Kapitel beschriebenen Anwendungsbeispiele untermauert.

Kapitel 6

Anwendungsbeispiele

In diesem Abschnitt wird eine Reihe von Modellierbeispielen beschrieben, die einen Überblick über die Möglichkeiten des entstandenen Entwurfswerkzeugs für heterogene Systeme vermitteln sollen. Jedes Beispiel geht auf unterschiedliche Aspekte wie Modellbildung oder Interaktion ein.

6.1 Planplattenmikrometer

Das Planplattenmikrometer stellt ein Beispiel für ein System mit mechanischen und optischen Komponenten dar, das domänenübergreifend modelliert und simuliert werden soll. Beim Planplattenmikrometer handelt es sich um ein optisches Messinstrument. Seine Funktionsweise beruht auf dem Versatz eines Lichtstrahls bezüglich der optischen Achse eines Messfernrohrs, der aus der Verkippung einer planparallelen Platte resultiert. Der Versatz kann aus der (leicht zu messenden) Verkippung der Planplatte bestimmt werden (Abbildung 6.1). Der funktionale Zusammenhang zwischen beiden Größen ist nichtlinear. Da dennoch eine Messtrommel mit linearer Skalenteilung verwendet werden soll, ist ein geeigneter Verstellmechanismus zu finden, der das Übertragungsverhalten des Gesamtsystems linearisiert. Bereits diese vergleichsweise einfache Aufgabe verdeutlicht die Vorteile eines domänenübergreifenden Entwurfswerkzeugs bei der Auswahl und Auslegung von Lösungen.
Im Kontext dieser Arbeit stellt das Planplattenmikrometer eines der initialen Untersuchungsobjekte dar, anhand dessen Anforderungen für ein domänenübergreifendes Entwurfswerkzeug und erste Erfahrungswerte gesammelt wurden. Das im Folgenden vorgestellte Modell wurde mit einer modifizierten Version von *MASP* (*MASP/H*, siehe Abschnitt 4.2.2) erstellt, die die Untersuchung von Ansätzen zur

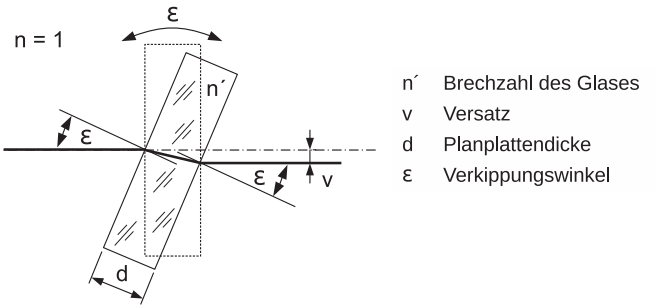


Abbildung 6.1: Strahlenversatz an einer planparallelen Platte

domänenübergreifenden Modellierung ermöglicht. Die Erweiterung bestand in der Beschreibung von Zusammenhängen der geometrischen Optik auf der Basis der in *MASP* verfügbaren Grundelemente (Punkt, Linie, Kreis). Mittels Gleichungs-Constraints wurden Reflexions- und Brechungsgesetz an optischen Wirkflächen umgesetzt. Somit konnten entsprechende Prinzipelemente und -kopplungen gebildet werden, mit denen die Modellierung optischer Baugruppen ähnlich einfach gelang, wie zuvor schon der Aufbau von Bewegungssystemen. Da beide Modelldomänen auf den gleichen Beschreibungsmitteln (des geometrischen Constraint-Solvers) aufbauten, war eine Kopplung und domänenübergreifende Simulation problemlos zu erreichen. Abbildung 6.2 illustriert die Beschreibung optischer Komponenten mittels der Grundelemente des geometrischen Constraint-Solvers.
Ein Lichtstrahlsegment bildet sich aus einer Linie, zwei Punkten und zwei Punkt-auf-Linie-Constraints (links oben). Zusammen mit einer Mediengrenze (Linie) ist die Beschreibung des Brechungs- und des Reflexionsgesetzes möglich. Die Endpunkte der Segmente verbinden sich mit der Mediengrenze über Punkt-auf-Linie-Constraints (rechts oben). Zusätzlich werden die Segmentpunkte per Punkt-gleich-Punkt-Constraint verknüpft (links unten). Im letzten Schritt erfolgt die Beschreibung der Gesetze mittels Gleichungs-Constraints aus dem Brechzahlenverhältnis und den Winkeln der Strahlen gegenüber der Mediengrenze (rechts unten). Die so abgebildeten Modellelemente und Zusammenhänge werden zu Prinzipsymbolen und Kopplungen zusammengefügt. Diese ermöglichen die Erzeugung komplexerer Modelle.
Abbildung 6.3 a zeigt das in *MASP* modellierte Planplattenmikrometer. Der Verstellmechanismus ist als Tangensgetriebe realisiert. Die Kopplung mit der Messtrommel erfolgt über ein zusätzliches Rädergetriebe. Wird diese verstellt, ändern sich der Winkel der Planplatte und der Versatz des Lichtstrahls (b).

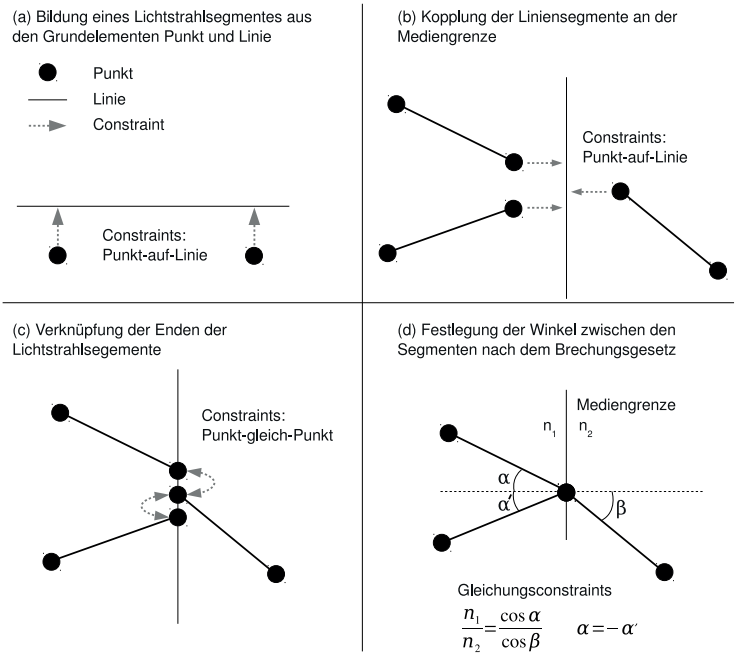


Abbildung 6.2: Beschreibung optischer Zusammenhänge mit Primitiven des geometrischen Constraint-Solvers

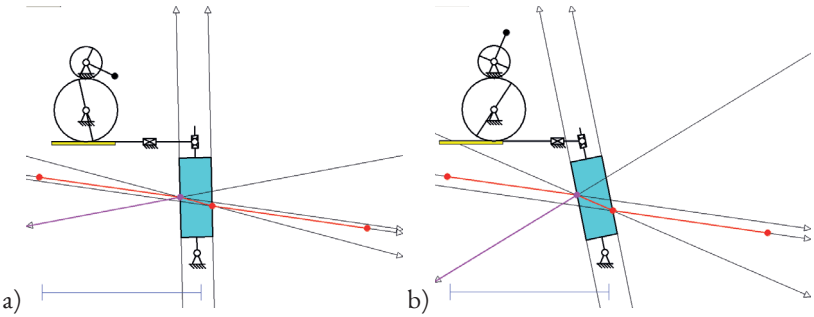


Abbildung 6.3: Domänenübergreifende Simulation eines Planplattenmikrometers. Interaktive Vorgabe des Strahlenganges und Berechnung der Stellung der Messtrommel (a und b).

Das vollparametrische Modell verfügt über einige weitere interessante Eigenschaften. Wird beispielsweise die Messtrommel fixiert, so passt sich bei einer Veränderung des Lichtstrahlversatzes die Dicke der Planplatte entsprechend an (Abbildung 6.4 a). Soll die Dicke konstant bleiben, lassen sich andere Freiheitsgrade zur Realisierung des Versatzes finden. So berechnet in Teilbild b der Constraint-Solver einen anderen Brechungsindex für die Planplatte (vgl. mit Abbildung 6.3 b). Wird schließlich die Parallelitätsbedingung aufgehoben, ist eine Deformation der Platte zu einem optischen Keil möglich (Teilbilder c und d).

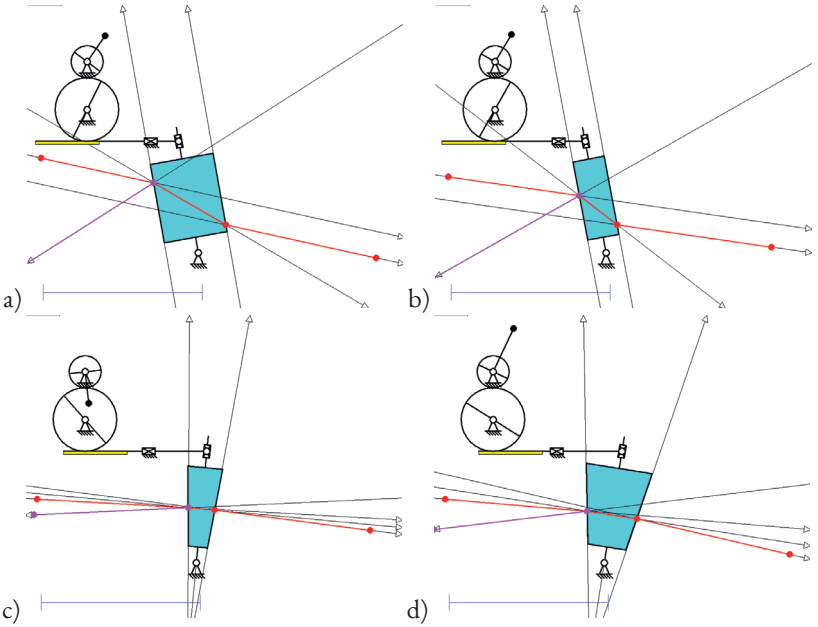


Abbildung 6.4: Domänenübergreifende Simulation eines Planplattenmikrometers. Interaktive Vorgabe des Strahlenganges und Berechnung der Dicke der Planplatte (a) oder des Brechungsindex des Werkstoffes der Platte (b). Deformation der Platte nach Aufgabe der Parallelitätsbedingung (c und d).

Dieser Versuch der Modellierung heterogener Systeme verdeutlichte die Vorteile einer domänenübergreifenden Modellbildung und Simulation. Gleichzeitig zeigte das Beispiel die Grenzen der Erweiterbarkeit einer domänenspezifisch arbeitenden Software wie *MASP* auf. Die Modellierung war hier weitgehend auf die Erfordernisse des geometrischen Constraint-Solvers abgestimmt. Gelang die Abbildung

von Zusammenhängen der Optik noch problemlos, würden andere, weniger geometriezentrierte Domänen schwieriger zu beschreiben sein. Neben der Forderung nach einem solverunabhängigen, domänenübergreifenden Gesamtmodell deutete sich die Notwendigkeit der Integration unterschiedlicher Berechnungsmodule zur Untersuchung einer breiteren Palette von Eigenschaften des Modells an. Diese Forderungen flossen später in das Konzept des Modellierwerkzeugs für heterogene Systeme ein.

6.2 Autokollimator

Ein Autokollimator ist ein optisches Messinstrument zur berührungsfreien Messung kleiner Winkel. Es umfasst sowohl optische als auch mechanische Eigenschaften, die durch das Entwurfswerkzeug entsprechend abzubilden und zu simulieren sind. Anhand dieses Beispiels soll die Vorgehensweise bei der interaktiven Modellierung mit *Hesym* demonstriert werden.

6.2.1 Interaktiver Modelliervorgang

Als Basis für die Modellierung des optischen Teilsystems findet das Modellelement „optische Achse“ Verwendung. Als Interaktionshilfe ist diese nicht nur anschaulich, sondern ermöglicht die Bindung von Freiheitsgraden bei der Positionierung der Prinzipelemente im dreidimensionalen Raum. Auf die optische Achse werden eine Lichtquelle mit Blende und eine sphärische Linse platziert. Dies geschieht, indem der Nutzer den Interaktor zur Erzeugung optischer Bauelemente (*OpticsCreator*) aktiviert und per Mauscursor die gewünschte Stelle auf der optischen Achse wählt (Abbildung 6.5).

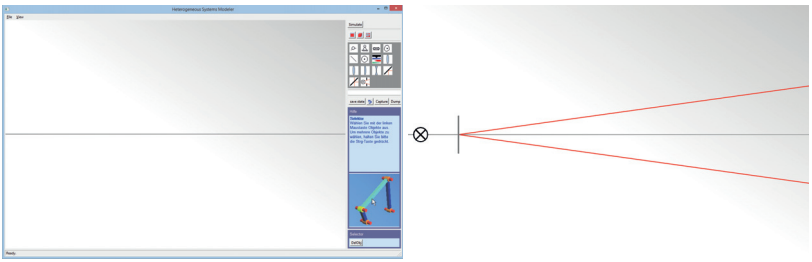


Abbildung 6.5: Optische Achse (links) und positionierte Lichtquelle (rechts)

Bereits während der Modellierung findet die Simulation des Strahlenganges statt, so dass die Auswirkungen von Änderungen an den Bauelementen sofort ersichtlich sind. So kann beispielsweise die Blende der Lichtquelle mittels eines Interaktors so eingestellt werden, dass das ausgesendete Licht vollständig durch die Linse verläuft. Da vorerst nur der generelle Strahlengang modelliert werden soll, kann diese Einstellung „per Augenmaß“ geschehen (Abbildung 6.6).

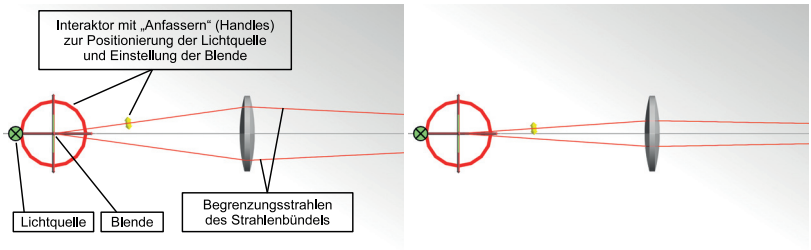


Abbildung 6.6: Positionierung der Linse und Einstellung der Blende

Das Funktionsprinzip des Autokollimators erfordert, dass das von der Lichtquelle ausgesendete Licht hinter der Linse parallel verläuft. Der Nutzer hat nun die Möglichkeit, die Lichtquelle in den Brennpunkt der Linse zu positionieren oder den Brennpunkt in die Lichtquelle zu verschieben. Beide Varianten sind mittels interaktiver Manipulation des 3D-Modells realisierbar. Da die eingesetzten Interaktoren eine Fangen-Funktion (Snapping) implementieren, ist es möglich, bei der Interaktion qualitative Aussagen zu treffen („Brennpunkt der Linse liegt im Zentrum der Punktlichtquelle“). Dies entbindet den Nutzer von der Eingabe (ohnehin nicht genau bekannter) Zahlenwerte für Position oder Brennweite (Abbildung 6.7).

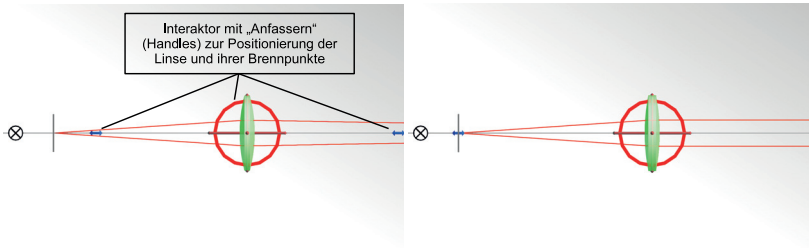


Abbildung 6.7: Anpassung des Strahlengangs durch Verschiebung des Brennpunktes der Linse in das Zentrum der Lichtquelle

Das nun parallelisierte Licht wird von einem ebenfalls auf der optischen Achse positionierten Spiegel in die Lichtquelle zurückgeworfen. Dieser Spiegel repräsentiert das Messobjekt und verändert durch seine Verkippung den Strahlengang des zurückkehrenden Lichtes. Die so entstehende Differenz ermöglicht die Berechnung des Verkippungswinkels und kann beispielsweise mittels einer CCD-Zeile gemessen werden (Abbildung 6.8).

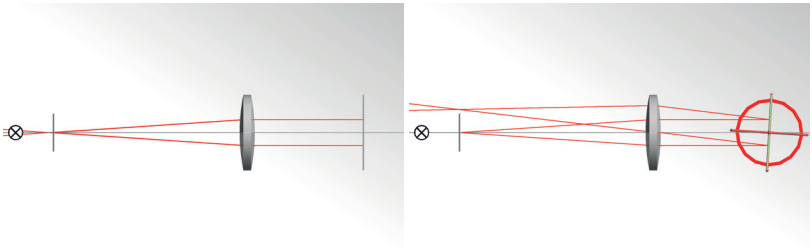


Abbildung 6.8: Einfügen des Messspiegels

Der hierfür notwendige Raum wird allerdings bereits von der Lichtquelle belegt, so dass ein halbtransparenter Spiegel zur Auskopplung und Umlenkung des zurückkehrenden Lichtes zum Einsatz kommt (Abbildung 6.9). Dieser Spiegel wird zunächst auf der optischen Achse platziert und anschließend mit einem Transformationsinteraktor um 45 Grad gedreht. Um eine exakte Eingabe des Winkels mit der Maus zu ermöglichen, kann der Interaktor so konfiguriert werden, dass er in bestimmten Winkellagen einrastet (Fangen, *Snapping*). Wird nun der Messobjektspiegel verkippt, wandert der zu messende Rückkehrpunkt in einer um 90 Grad gedrehten Ebene, in der das CCD-Element positioniert wird (Abbildung 6.10). Die Modellierung des optischen Teils des Autokollimators ist an diesem Punkt abgeschlossen.

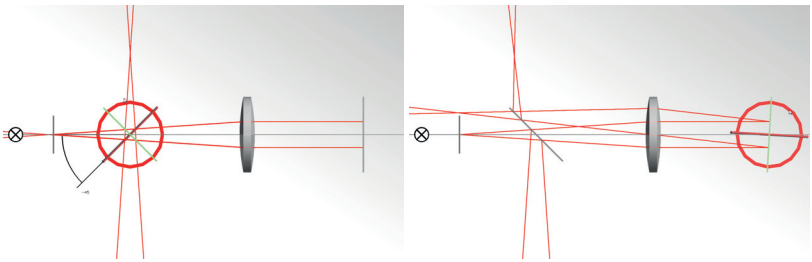


Abbildung 6.9: Auskopplung mittels eines halbtransparenten Spiegel

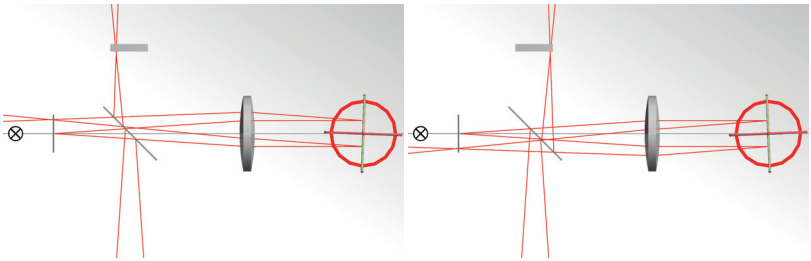


Abbildung 6.10: Positionierung des CCD-Elementes

Da die direkte manuelle Verkippung des Spiegels nicht feinfühlig genug ist, soll ein Verstellmechanismus eingesetzt werden. Zu dessen Modellierung wird mit dem Interaktor *PlaneSelector* eine Ebene relativ zum Spiegel gewählt. Auf dieser werden anschließend kinematische Prinzipelemente positioniert. Für das Modellierbeispiel in Abbildung 6.11 wurde beispielhaft ein Koppelgetriebe mit sehr hohem Übersetzungsverhältnis gewählt. Abschließend werden der Spiegel und das benachbarte Getriebeglied ausgewählt und miteinander verknüpft.

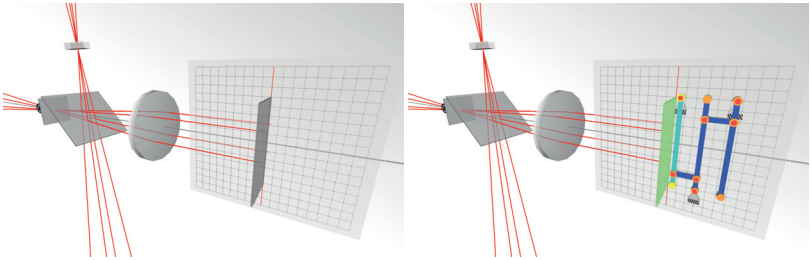


Abbildung 6.11: Modellierung der Verstellmechanik

Bewegungen des Mechanismus werden nun auf den Spiegel übertragen (Abbildung 6.12). Somit wird die Simulation des Verhaltens des Gesamtsystems über die Domänen der Optik und der Mechanik ermöglicht.

6.2.2 Berechnungsvorgang

Für die Simulation des Autokollimators kommen ein Mehrkörpersystem-Solver und ein Raytracer zum Einsatz. Beide Berechnungsmodule leiten ihre Teilmodelle aus dem domänenübergreifenden Gesamtmodell ab. Die Kopplung beider Teilmodelle erfolgt über die Transformationen des Spiegels und des damit verbunde-

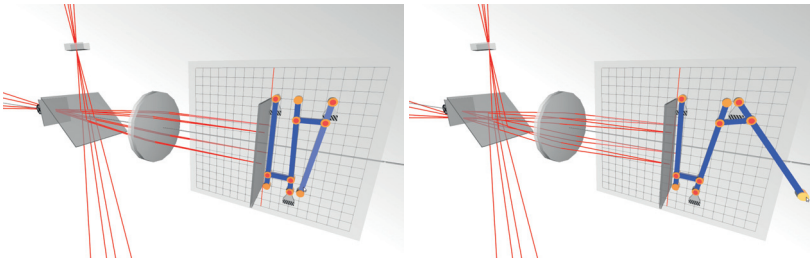


Abbildung 6.12: Feinfühlige Verstellung mittels Koppelgetriebe

nen Getriebegliedes. Hat beispielsweise die Mehrkörpersimulation einen neuen Zustand des mechanischen Teilmodells berechnet und in das domänenübergreifende Gesamtmodell übermittelt, wird anschließend der Raytracer den Zustand der aktualisierten Transformation übernehmen und so die Position des Spiegels in der Berechnung des Strahlenganges berücksichtigen.

6.3 Balanceregelung für ein inverses Pendel

Beim inversen Pendel handelt es sich um ein Standardbeispiel der Regelungstechnik. Es eignet sich als Testfall, da es einerseits mechanische, antriebstechnische sowie regelungstechnische Komponenten vereint und andererseits ein leicht zu verifizierendes Ergebnis erzeugt.

6.3.1 Aufbau und Modellierung in Hesym

Die Anordnung basiert typischerweise auf einem Wagen, der sich auf einer Führungsschiene bewegen kann und der über ein Drehgelenk mit dem Pendel verbunden ist. Der Antrieb greift am Wagen an und beschleunigt diesen in Richtung der Schiene. Gemessen werden der Winkel des Pendels und die Position des Wagens. Mit diesen Informationen kann ein Regler den Antrieb so ansteuern, dass das Pendel in ein labiles Gleichgewicht versetzt wird. Die detaillierte Beschreibung eines solchen Aufbaus befindet sich in [Ber04].
Abbildung 6.13 zeigt die im Entwurfswerkzeug *Hesym* modellierte Struktur als technisches Prinzip. Der mechanische Anteil besteht aus einem Drehgelenk (rot), das ein den Wagen repräsentierendes, gestellfestes Schubgelenk (grau) mit einem als Pendel fungierenden Getriebeglied (blau) verbindet. Der Antrieb (rechts unten) kann entlang der roten Linie Kräfte auf den Gleitstein des Schubgelenkes wirken lassen.

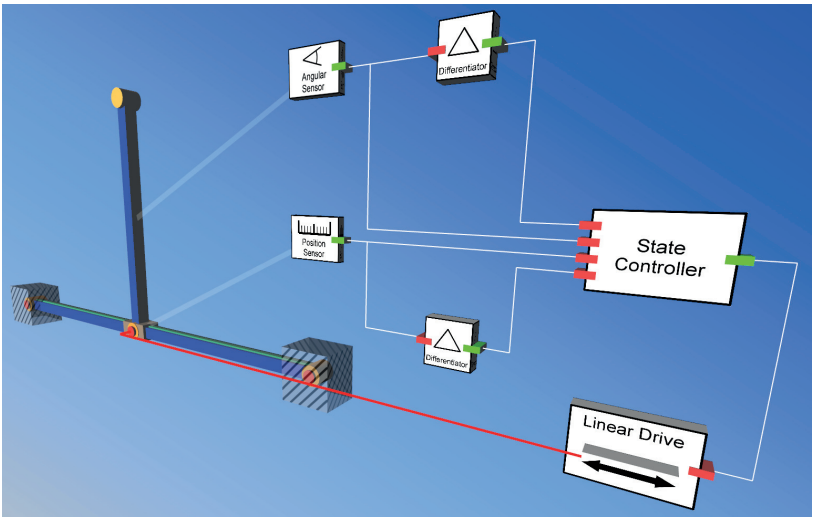


Abbildung 6.13: Inverses Pendel mit Zustandsregler

Die Regelung (im Abbildung rechts oben) beginnt mit der Messung des Pendelwinkels und der Wagenposition, aus denen jeweils eine differenzierte Größe berechnet wird. Alle vier Zustandsgrößen des Systems (Winkel α_{Pendel} und Winkelgeschwindigkeit ω_{Pendel} des Pendels, Position s_{Wagen} und Geschwindigkeit v_{Wagen} des Wagens) werden dem Zustandsregler übergeben, der daraus eine geeignete Antriebskraft $F_{Antrieb}$ bestimmt, die die Balance des Pendels sicherstellt:

$$F_{Antrieb} = k_s s_{Wagen} + k_v v_{Wagen} + k_\alpha \alpha_{Pendel} + k_\omega \omega_{Pendel} \qquad (6.1)$$

6.3.2 Modellvisualisierung über Domänengrenzen hinweg

Einen interessanten Aspekt an diesem Beispiel bietet die Visualisierung des Aufbaus, insbesondere die Kopplung der domänenspezifischen Darstellungsformen des mechanischen und des regelungstechnischen Teilmodells (3D-Modell und Blockdarstellung). Die Herausforderung besteht hierbei in der Darstellung des Zusammenhangs zwischen den Sensoren und dem Objekt, dessen Messwerte sie bestimmen. Während der Sensor in der Blockdarstellung seine Position naturgemäß nicht verändert, bewegt sich das Messobjekt Pendel. Die Überbrückung des hieraus resultierenden veränderlichen Abstands erfolgt im Beispiel durch eine halbtransparente

Linie, die den Zusammenhang zwischen beiden Modellbestandteilen dokumentiert und später durch ein konkretisiertes Messprinzip ersetzt werden kann. Der auf diese Weise visuell geschlossene Regelkreis verbessert die Verständlichkeit der Anordnung.

6.3.3 Erweiterung um einen Algorithmus zum Aufschwingen des Pendels

Eine mögliche Erweiterung des Beispiels besteht in der Formulierung eines Aufschwingalgorithmus, der das Pendel aus einer hängenden Ausgangsposition in den Balancezustand überführt [WDL95]. Hierzu wird der Zustandsregler durch einen programmierbaren Block ersetzt, dessen Funktion in der Skriptsprache *Lua* [IdC06] ausgedrückt wird. Solche Miniaturprogramme bestehen meist nur aus wenigen Zeilen Code, die aus den Eingangsvariablen des Blocks seine Ausgangsgrößen berechnen. Ein erster Entwurf des Aufschwingalgorithmus könnte als Pseudocode folgendermaßen aussehen:

```
1. beschleunigeLinks( 200ms )
2. beschleunigeRechts( 200ms )
3. wenn Betrag( Pendelwinkel ) < Mindestwinkel gehe zurück zu 1
4. balancierePendel()
```

Ein Simulationsversuch offenbart einen relativ schlecht koordinierten Aufschwingvorgang, da das Pendel meist im falschen Moment beschleunigt wird. Es ist also zweckmäßig, nach jedem Auslenken des Pendels auf dessen Nulldurchgang zu warten und anschließend es anschließend zu beschleunigen. Der Algorithmus erweitert sich zu:

```
1. beschleunigeLinks( 200ms )
2. warteAufNulldurchgang()
3. beschleunigeRechts( 200ms )
4. warteAufNulldurchgang()
5. wenn Betrag( Pendelwinkel ) < Mindestwinkel gehe zurück zu 1
6. balancierePendel()
```

Mit dieser Verbesserung gelingt das Aufschwingen definiert und in wenigen Zyklen. Der hier angedeutete Vorgang der Entwicklung des Algorithmus demonstriert den Nutzen der programmierbaren Blöcke. Sie ermöglichen, ihr Verhalten mit geringem Aufwand flexibel zu beschreiben und zu modifizieren. Das so ausgedrückte technische Prinzip des Reglers kann später problemlos in eine konkrete elektronische Komponente umgesetzt werden, da hier ohnehin oft Microcontroller und Speicherprogrammierbare Steuerungen zur Anwendung kommen.

Aus der Sicht einer aussagekräftigen Visualisierung bleibt die Funktionalität des Algorithmus allerdings hinter dem Funktionsblock verborgen. Als Alternative zur Beschreibung mittels Skriptcode bietet sich in diesem Beispiel eine Darstellung des Algorithmus als Zustandsmaschine (Endlicher Automat, finite state machine) an. Diese Möglichkeit wurde nicht implementiert, ist aufgrund ihrer Anschaulichkeit aber durchaus interessant (Abbildung 6.14). Die einfache Modifizierbarkeit der Variante mit Skriptcode erreicht sie hingegen nicht.

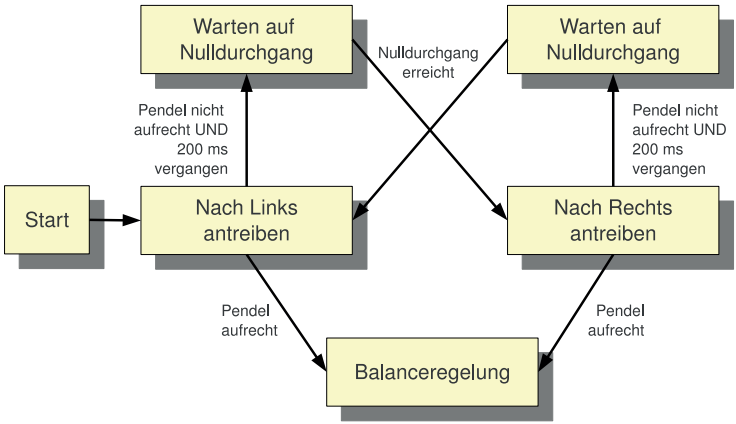


Abbildung 6.14: Zustandsmaschine des Aufschwingalgorithmus

6.4 Fehlerschätzung an der Nanopositionier- und Nanomessmaschine

Die Nanopositionier- und Nanomessmaschine (NPMM) ist eine hochpräzise Mess- und Positioniereinrichtung, die im Rahmen des Sonderforschungsbereichs 622 an der Technischen Universität Ilmenau entwickelt wurde [Hau02]. Die Maschine verfügt über einen Messbereich von 25mm x 25mm x 5mm. Ihr Funktionsprinzip ist durch eine feststehende Tastspitze gekennzeichnet, relativ zu der das Messobjekt bewegt wird. Die Istposition des Messobjektes wird mittels laserinterferometrischer Messungen bestimmt und von einer Regeleinrichtung mit der Sollposition verglichen. Mit diesen Informationen erfolgt die Ansteuerung der Antriebe, die den Positionierfehler ausgleichen bzw. verringern. Eine Reihe konstruktiver, verfahrenstechnischer und werkstofftechnischer Maßnahmen, wie die Umsetzung des Abbeschen Komparatorprinzips oder die Verwendung von Materialien mit

geringem Wärmeausdehnungskoeffizienten sorgen für eine extrem hohe Positioniergenauigkeit.
Die NPMM wird kontinuierlich weiterentwickelt und verbessert. In diesem Zusammenhang finden auch Betrachtungen zur Optimierung der Messkreise statt (z. B. [Fra10]). Die Genauigkeit der Messung wird unter anderem durch mechanische und thermische Belastungen beeinflusst. In einem Teilprojekt des SFB 622 wurde der Frage nachgegangen, inwieweit sich solche Einflüsse durch Ausnutzung bestimmter konstruktiver Eigenschaften minimieren lassen (z. B. Symmetrien, Invarianzen). Zu diesem Zweck sollten unterschiedliche Ausprägungen der Messanordnung, wie beispielsweise die in Abbildung 6.15 dargestellte Variante mit Tetraederspiegel, miteinander verglichen werden.

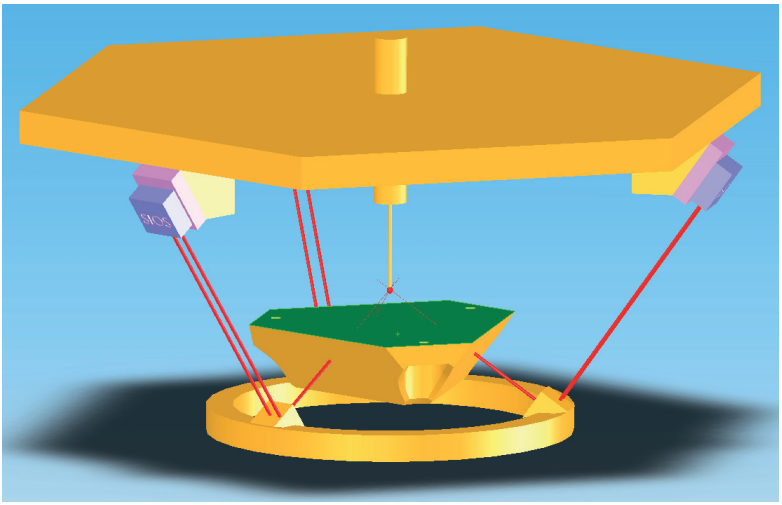


Abbildung 6.15: Messanordnung der Nanopositionier- und Nanomessmaschine (Quelle: SFB 622 der TU Ilmenau)

Die Suche nach einem geeigneten Simulationswerkzeug gestaltete sich schwierig, da sich die Problemstellung nicht nur über unterschiedliche Domänen erstreckt, sondern diese zusätzlich über einen Regelkreis miteinander in Beziehung stehen (interferometrische Messung, Auswertung im Regler, Ansteuerung der Antriebe und der Positioniereinrichtung). Dieses Beispiel verdeutlicht die in Kapitel 2 beschriebenen Nachteile verfügbarer Softwarelösungen. Der parallele Einsatz domänenspezifischer Werkzeuge würde die Modellierung der Teilsysteme zwar durchaus ermöglichen, die Betrachtung des Regelkreises als Ganzes aber erschweren. Auf der

anderen Seite bieten allgemeine Simulationssysteme, z. B. auf der Basis von *Modelica*, die Möglichkeit zur domänenübergreifenden Abbildung des Gesamtsystems, weisen aber Defizite bezüglich einer den Domänen angepassten Modelldarstellung auf.
Aufgrund der Tatsache, dass bei der Modellierung der kinematischen und optischen Komponenten der Mess- und Positioniereinrichtung vornehmlich komplexe räumliche und geometrische Informationen einzugeben waren, kam zunächst eine Lösung mit dem CAD-System *SolidWorks* zum Einsatz. Es handelte sich dabei um dreidimensionale Skizzen aus Ebenen, Liniensegmenten und geometrischen Constraints. Abbildung 6.16 zeigt eines dieser Modelle. Das Dreieck im unteren Teil des Bildes repräsentiert den Tetraederspiegel, die davon ausgehenden Linien die Strahlen der Laserinterferometer. Das Element mit der Bezeichnung „Ebene1“ verkörpert den Rahmen mit den Umlenkspiegeln.

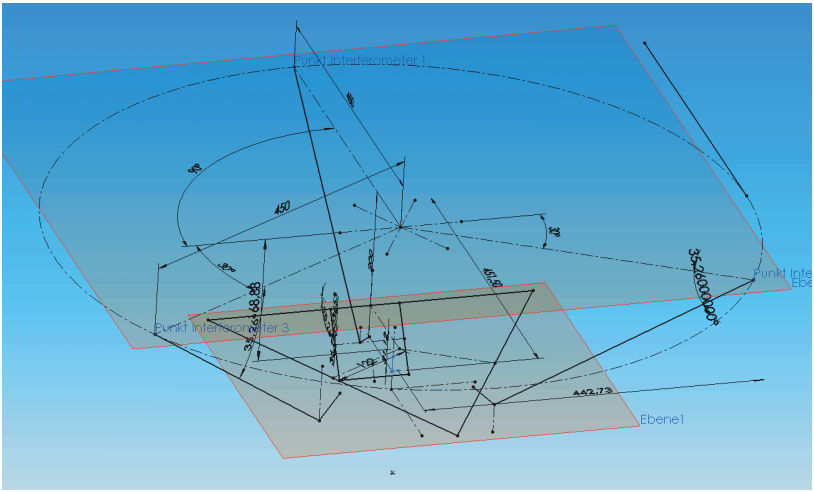


Abbildung 6.16: Geometrisches Ersatzmodell der Messanordnung im CAD-System *SolidWorks* (Quelle: SFB 622 der TU Ilmenau)

Eine der Untersuchungen beschäftigte sich mit den Auswirkungen, die eine Verkippung dieses Rahmens auf das Positionierergebnis hätte. Dazu sollte die Verkippung schrittweise erhöht werden, während die Abweichung des Positionierergebnisses von der Idealposition als Fehlermaß herangezogen wurde. In der Lösung auf Basis eines CAD-Modells ist der Algorithmus des Reglers nicht explizit hinterlegt. Sein Verhalten ergibt sich aus der Annahme, dass in der Simulation jeweils dieselbe

Position angefahren wird. In diesem Fall haben alle Interferometerstrahlen eine konstante Länge, was im CAD-Modell mittels Constraints formuliert ist. Das Ergebnis der Berechnung war eine Tabelle, die den Positionierfehler in Abhängigkeit vom Verkippungswinkel darstellt. Die beschriebene Lösung in *SolidWorks* weist einige offensichtliche Nachteile auf. Die Problemstellung mit ihren drei Domänen muss zuerst in ein geometrisches Ersatzmodell transformiert werden. Der Ingenieur ist also nicht in der Lage, das Szenario in Begriffen wie Tetraederspiegel, Koordinatentisch, Sensor oder Regler zu beschreiben. Stattdessen formuliert er das Modell in Form geometrischer Konzepte wie Ebenen, Linien und Abstände. Hinzu kommt, dass der implizit repräsentierte Regler wenig flexibel ist. Die über ihn getroffenen Annahmen stellen lediglich einen Spezialfall dar. Komplexere Regelvorgänge lassen sich auf diese Weise nicht abbilden.
Der in Abbildung 6.16 dargestellte Aufbau ist wenig anschaulich und erschließt sich dem Betrachter ohne zusätzliche Erklärungen kaum. Nicht nur seine Modellierung, sondern auch spätere Modifikationen können nur mit erheblichem Aufwand durchgeführt werden. Ein alternatives Modell der Mess- und Positioniereinrichtung wurde mit dem Entwurfswerkzeug für heterogene Systeme *Hesym* erstellt (Abbildung 6.17). Die Ergebnisse der Simulation stimmen mit denen der Berechnungen auf Basis von *SolidWorks* überein.

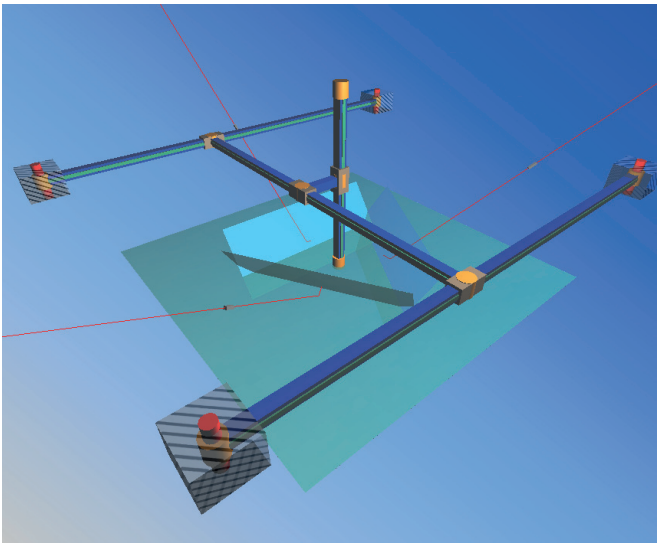


Abbildung 6.17: Modell im Entwurfswerkzeug für heterogene Systeme *Hesym*

Die Unterschiede zwischen beiden Ansätzen zeigen sich in der Handhabung und der Flexibilität der erstellten Modelle. Obwohl es sich um eine frühe Programmversion von *Hesym* handelte, in der noch nicht alle Interaktionswerkzeuge implementiert waren, gestaltete sich die Eingabe des Modells unkomplizierter als im CAD-System. Gleichzeitig war der Zeitaufwand geringer. Dies ist auf die Verwendung domänenangepasster Interaktionstechniken und Modellbestandteile (Symbole) zurückzuführen, die sich an der Begriffswelt des Ingenieurs orientieren. Auch die noch recht simple Modellvisualisierung führte bereits zu einer deutlich verbesserten Anschaulichkeit. Ein weiterer Vorteil der so gestalteten Nutzerschnittstelle und der symbolbasierten Modellbildungsstrategie liegt in der einfachen Modifizierbarkeit des modellierten Aufbaus. Die Erstellung anderer Varianten, beispielsweise unter Einsatz von Prismen anstelle der Spiegel, gelingt nun einfacher und schneller.
Der in der Abbildung nicht visualisierte Regler ist als Skriptcode realisiert. Im Gegensatz zu der in *SolidWorks* eingesetzten Behelfslösung sind somit nahezu beliebige Algorithmen umsetzbar. Die Simulation des Bewegungssystems erfolgte mittels der *Open Dynamics Engine*, während die optischen Komponenten von einem einfachen *Raytracer* berechnet wurden. Das zwischen den Berechnungsmodulen eingesetzte solverunabhängige Gesamtmodell ermöglichte die Synchronisation in jedem Berechnungsschritt und damit die Simulation des geschlossenen Regelkreises.

6.5 Anwendung im Rahmen der Digitalen Mechanismen- und Getriebebibliothek

Dieser Abschnitt beschreibt den Einsatz von *MASP* und *Hesym* in einem Produktionsprozess zur Erstellung interaktiver Animationen und anderer Inhalte auf der Basis von Simulationsmodellen. Die aufgeführten Beispiele demonstrieren die Eignung der beiden Werkzeuge für eine solche Anwendung anhand bestimmter Aspekte. Dazu zählen die schnelle und unkomplizierte Erstellung von Modellen mittels eines Baukastens aus Prinzipsymbolen, die automatische Generierung von Simulationsmodellen, die Möglichkeiten zum Einsatz unterschiedlicher Stile zur Modellvisualisierung sowie den Export visualisierter Berechnungsdaten in aufgabenangepasster Form (z. B. zur Überlagerung von Videodaten).

6.5.1 Beschreibung der Digitalen Mechanismen- und Getriebebibliothek DMG-Lib

Die DMG-Lib ist eine digitale Bibliothek zur Sammlung und Bewahrung des Wissens über Getriebetechnik. Sie digitalisiert und verknüpft sehr unterschiedliche

Quellformen (Bücher, Zeichnungen, Dias, auch körperliche Modelle) und stellt die Ergebnisse auf einer Web-Plattform der Öffentlichkeit zur Verfügung (Abbildung 6.18, [DMG13, BBD+12, BBD+09, BCMD06]).

Abbildung 6.18: Portal der Digitalen Mechanismen- und Getriebebibliothek (Quelle: www.dmg-lib.org)

Die Arbeiten im Rahmen der DMG-Lib gehen über die reine Digitalisierung hinaus. So werden unter anderem statische Darstellungen von Mechanismen (z. B. Abbildungen in Büchern oder Fotos) per Animation in Bewegung versetzt. Auf diese Weise wird nicht nur eine verbesserte Verständlichkeit der Abbildungen, sondern auch eine schnellere Informationsaufnahme seitens des Betrachters erreicht [CLR+13, BDR07].

6.5.2 Problembeschreibung und Anwendungsfälle

Das Projekt DMG-Lib verfügt über einen umfangreichen Bestand an Mechanismendarstellungen. Neben Foto- und Diasammlungen existieren Lehrbücher und Getriebeatlanten [BDR11]. Nicht nur die große Anzahl, sondern auch die häufig sehr komplexen Bewegungen machen den Versuch einer manuellen Animation, z. B. per Keyframe-Technik zu einer kaum zu bewältigenden Aufgabe. Es liegt daher

nahe, rechnergestützte Modelle der abgebildeten Mechanismen zu erstellen und diese mittels Bewegungssimulation zu animieren.
Zu den besonderen Problemen dieses Vorhabens gehört, dass die Struktur der Mechanismen in den Abbildungen nur implizit gespeichert ist. Die digitalen Bilddateien („Pixelbilder") bestehen aus einer Matrix von Bildpunkten mit Farbwerten, die keine direkt maschinell verarbeitbaren kinematischen Zusammenhänge beschreiben. Übliche Verfahren der Bewegungssimulation erfordern allerdings eine explizite Repräsentation der Mechanismenstruktur. Vor diesem Hintergrund ist es notwendig, sowohl einen Arbeitsprozess als auch ein Werkzeug zu erschaffen, mit denen aus den bildlichen Darstellungen in vertretbarer Zeit kinematische Strukturen extrahiert werden können.
Eine automatische Extraktion gestaltet sich aufgrund des sehr unterschiedlichen Bildmaterials (Qualität, Darstellungsstil) schwierig. Es wird daher eine durch Methoden der Bildverarbeitung unterstützte manuelle Verarbeitung bevorzugt. Ein entsprechendes Werkzeug muss so gestaltet sein, dass ein möglichst geringer Zeitbedarf für die Interaktion entsteht. Im Rahmen des Projektes DMG-Lib werden angepasste Versionen von *Hesym* und seines Vorgängers *MASP* eingesetzt.
Typische Anwendungen der Werkzeuge sind

- die Erzeugung von Animationen auf der Basis von Fotos und Zeichnungen,
- die Extraktion einer Strukturbeschreibung von Mechanismendarstellungen (XML) und
- die Berechnung von Zusatzinformationen zur Überlagerung mit Videomaterial.

6.5.3 Arbeitsablauf

Abbildung 6.19 stellt den Arbeitsablauf der Erzeugung von Animationen dar. Aus den Eingabedaten wird zunächst eine Strukturbeschreibung extrahiert, die als Basis für ein kinematisches Modell dient. Dieses wird als XML-Beschreibung in einer Datenbank abgelegt. Im folgenden Schritt wird mit dem Modell eine Bewegungssimulation durchgeführt, wobei die Darstellung in einem bestimmten Visualisierungsstil erfolgt (z. B. Prinzipsymbolik, Grobgestalt). Das Ergebnis der Simulation ist eine Animation, die optional mit Berechnungsdaten überlagert werden kann.

6.5.4 Eingabedaten

Der Arbeitsprozess beginnt ausgehend von den Eingabedaten, die in Form digitaler Pixelbilder (Standbilder, Videosequenzen) vorliegen. Typische Beispiele für Quel-

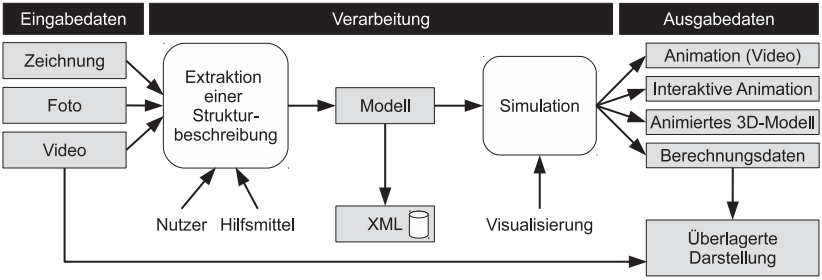


Abbildung 6.19: Arbeitsablauf zur Erzeugung von Animationen aus Bild- und Videomaterial

len sind Fotos oder Videoaufzeichnungen körperlicher Getriebemodelle, Skizzen, technische Zeichnungen sowie Abbildungen in digitalisierten Büchern (Abbildung 6.20).

Abbildung 6.20: Beispiele für Eingabedaten bei der Animationserzeugung (Quelle: www.dmg-lib.org)

6.5.5 Extraktion der Struktur der abgebildeten Mechanismen

Das Ziel des Extraktionsprozesses besteht in der Anfertigung einer Beschreibung der kinematischen Struktur aus der Abbildung eines Mechanismus. Er kann vollständig manuell oder mit Unterstützung durch Verfahren der Bildverarbeitung durchgeführt werden [BDHR05]. Insgesamt erweist sich das beschriebene Verfahren als relativ schnell. In den meisten Fällen liegt das Ergebnis innerhalb weniger Minuten vor.

Ablauf der manuellen Extraktion

Zur Vorbereitung der manuellen Extraktion wird die Abbildung des Mechanismus in *MASP* oder *Hesym* geladen und als „Hintergrundbild“ im Modellier- und Zeichenbereich angezeigt. Der Nutzer interpretiert den Inhalt des Bildes und platziert entsprechende Prinzipsymbole über der Abbildung (Abbildung 6.21). Für eine genauere Positionierung der Symbole kann die Darstellung des Bildes vergrößert werden. Besondere Konturen (z. B. Kurvenscheiben) werden als Polygonzüge approximiert. Das Ergebnis ist die kinematische Strukturbeschreibung des Mechanismus. Abschließend überprüft der Nutzer die Funktion des Modells mittels Bewegungssimulation und speichert es in eine Datei oder Datenbank.

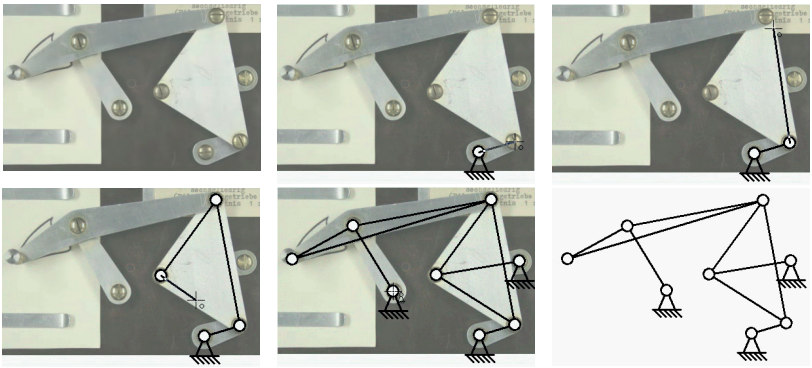


Abbildung 6.21: Manuelle Extraktion einer kinematischen Struktur durch Positionierung von Prinzipsymbolen auf einer Abbildung

Probleme des Extraktionsprozesses

Ein Problem der manuellen Extraktion besteht darin, dass die Positionierung der Prinzipsymbole durch optisches Vergleichen mit dem Inhalt des Pixelbildes geschieht und somit dem „Augenmaß“ des Nutzers unterliegt. Selbst bei gewissenhafter Arbeitsweise entstehen kleine Abweichungen zu den in der Abbildung dargestellten Maßen und Lagen der Bestandteile des Mechanismus. Verstärkt wird dieser Effekt durch bereits im Abbildung enthaltene Fehler, die von mangelhaften Fotos, Videos oder Scandaten (Unschärfe, perspektivische Verzerrung, Kompressionsartefakte) bis zu ungenau ausgeführten Skizzen und Zeichnungen reichen. In den meisten Fällen sind die Auswirkungen der so in das Modell einfließenden Abweichungen zu vernachlässigen, da sie selten zu grundlegend anderen Bewegungseigenschaften des Mechanismus führen und somit das Ziel einer verständ-

nisfördernden Animation weiterhin erfüllt werden kann. Eine Ausnahme bilden kinematische Strukturen, die ihre Funktion nur unter Annahme bestimmter Maße oder Maßverhältnisse erfüllen (z. B. Geradführungen). In diesen Fällen ist ein gewisses getriebetechnisches Verständnis des Nutzers erforderlich, um die Maße entsprechend zu korrigieren. Zusätzliche Ungenauigkeiten können durch angenäherte Repräsentation von Konturen entstehen (z. B. Kurvenscheibe als Polygon).

Unterstützung der Extraktion mit Methoden der Bildverarbeitung

Das Ziel des Einsatzes von Verfahren der Bildverarbeitung besteht im automatischen Auffinden bestimmter Regelgeometrien oder zusammenhängender Konturen im Quellmaterial. Diese sollen dem Nutzer als Formelemente oder Positionierungshilfen bei der Modellierung zur Verfügung gestellt werden.
Ein einfaches Verfahren ist die *Hough*-Transformation [DH72]. Es ermöglicht die Suche nach parametrisierbaren Geometrien wie Linien oder Kreisen. In Getriebedarstellungen finden sich häufig kreisförmige Linien oder Flächengrenzen, die Hinweise auf rotatorische Bauelemente wie beispielsweise Drehgelenke oder Räder liefern. Die identifizierten Kreise werden im Abbildung angezeigt (überlagert) und können bei der interaktiven Modellierung zur einfachen Festlegung von Punkten und Rotationsachsen herangezogen werden (Abbildung 6.22). Im Allgemeinen werden mehr Kreise erkannt, als für die Modellierung des Mechanismus notwendig ist. Die Aufgabe des Nutzers besteht in der Auswahl der relevanten Kreise. Der Aufwand für die genaue manuelle Positionierung entfällt.

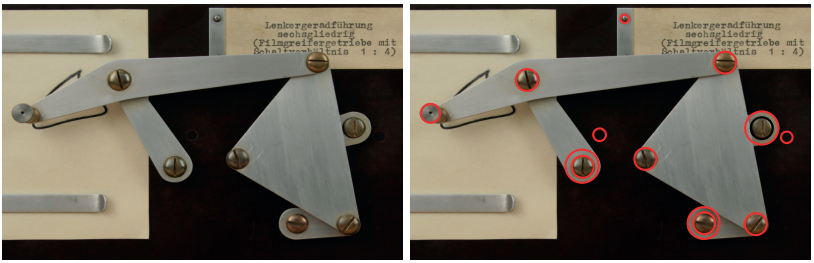


Abbildung 6.22: Automatisches Auffinden von Kreisen mittels Hough-Transformation am Beispiel des Bildes eines Filmgreifergetriebes (Quelle: Getriebesammlung der TU Dresden)

Ein weiteres Hilfsmittel ist das Auffinden von Konturen mittels Linienverfolgung und Segmentierung. Es ist von großem Nutzen, wenn der abgebildete Mechanismus individuelle Formelemente (z. B. Kurvenscheiben) enthält, die sonst manuell

modelliert werden müssten (Zeitaufwand). Das Verfahren beginnt mit einer Reihe von Vorverarbeitungsschritten, zu denen eine Rauschreduktion sowie eine Kantenerkennung unter Anwendung des Sobel-Operators [NA08] gehören. Das so behandelte Bild enthält nun Konturlinien an den Grenzen farbiger Flächen. Die Linien werden zunächst skelettiert und anschließend dem Linienverfolgungsalgorithmus zugeführt. Dieser findet Segmente zusammenhängender Bildpunkte, aus denen später die Gesamtkontur zusammengesetzt wird (Abbildung 6.23). Eine detaillierte Beschreibung des Verfahrens befindet sich in [Jäg08].

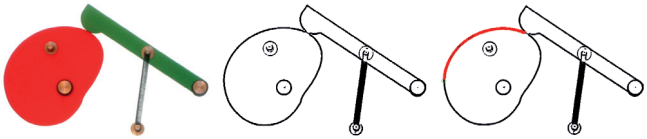


Abbildung 6.23: Finden von Konturen mittels Skelettierung und Segmentierung

6.5.6 Strukturbeschreibung

Das Ergebnis der Extraktion ist eine Beschreibung des technischen Prinzips des Mechanismus als Graph. Sie ist somit im Allgemeinen abstrakter als die Darstellung im Originalbild, weist aber gleiche kinematische Eigenschaften auf. Die Beschreibung wird als XML-basiertes Format (DMGM, *DMG-Lib Mechanism*, beschrieben in [Sto07]) gespeichert (Abbildung 6.24). Neben der Weiterverarbeitung zu Animationsdaten dienen diese XML-Dateien zum Aufbau eines Lösungsspeichers kinematischer Strukturen [RBD11, BDR08].

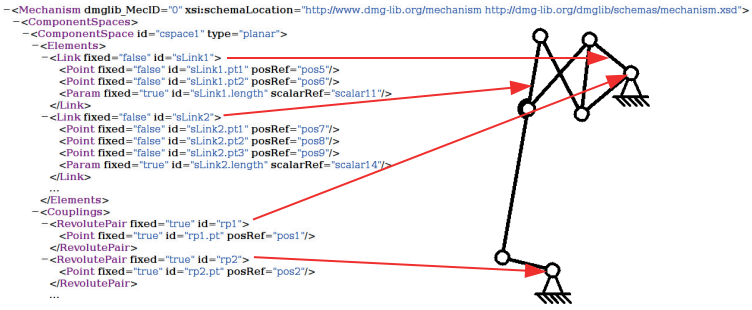


Abbildung 6.24: XML-Format zur Beschreibung kinematischer Strukturen

6.5.7 Visualisierungsstile

Die Simulation und Darstellung des Mechanismus ist bereits auf Basis des extrahierten technischen Prinzips möglich. Dennoch gibt es Fälle, in denen eine Anpassung der Visualisierung zweckmäßig ist, z. B. um den Darstellungsstil der Zeichnungen in einem bestimmten Buch nachzuahmen oder das Aussehen des Mechanismus an das einer Baureihe körperlicher Modelle anzulehnen (Abbildung 6.25). Eine andere Anwendung erweiterter Visualisierungen ist das Hinzufügen illustrierender Formelemente, die beispielsweise eine mögliche Anwendung oder besondere Eigenschaften des Mechanismus verdeutlichen (Abbildung 6.26).

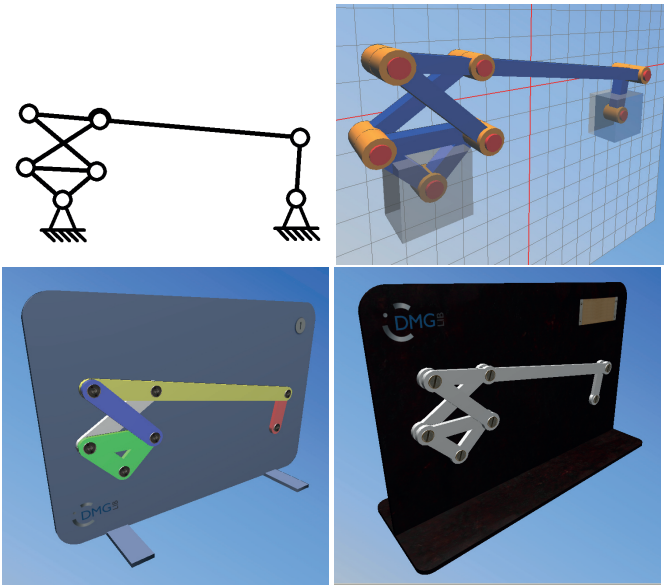


Abbildung 6.25: Visualisierungsstile am Beispiel des Kreuzlenkers: ebenes technisches Prinzip (links oben), dreidimensionales technisches Prinzip (rechts oben), körperliche Modelle (unten)

Für die Umsetzung der unterschiedlichen Darstellungsstile kommen die in Abschnitt 4.7.6 beschriebenen Visualisierungselemente zum Einsatz. Diese existieren einerseits in einer schnell und einfach zu verwendenden vordefinierten Form, die zusätzlich parametrisiert werden kann (Maße/Querschnitte, Farbe/Textur). Andererseits besteht die Möglichkeit, Dreiecksnetze frei zu modellieren (z. B. Polygoneingabe und Extrusion) oder aus anderen Modellierwerkzeugen zu importieren.

Die so entstehenden Visualisierungen werden mit dem Prinzipelement verknüpft und üblicherweise mit dessen Koordinatensystem bewegt.
Abbildung 6.26 zeigt die Erstellung eines Formelementes am Beispiel eines Führungsgetriebes. Ausgehend vom Originalbild (links oben) wird zunächst die kinematische Struktur extrahiert (rechts oben). Diese gibt die Bewegung des Mechanismus wieder, verzichtet aber auf die detaillierte Darstellung der anwendungsbezogenen Ausformung des Abtriebselementes. Daher wird der Umriss des Abtriebselementes als Polygon nachgezeichnet (rechts unten) und schließlich mittels einer Extrusionsoperation in einen dreidimensionalen Körper verwandelt (rechts unten).

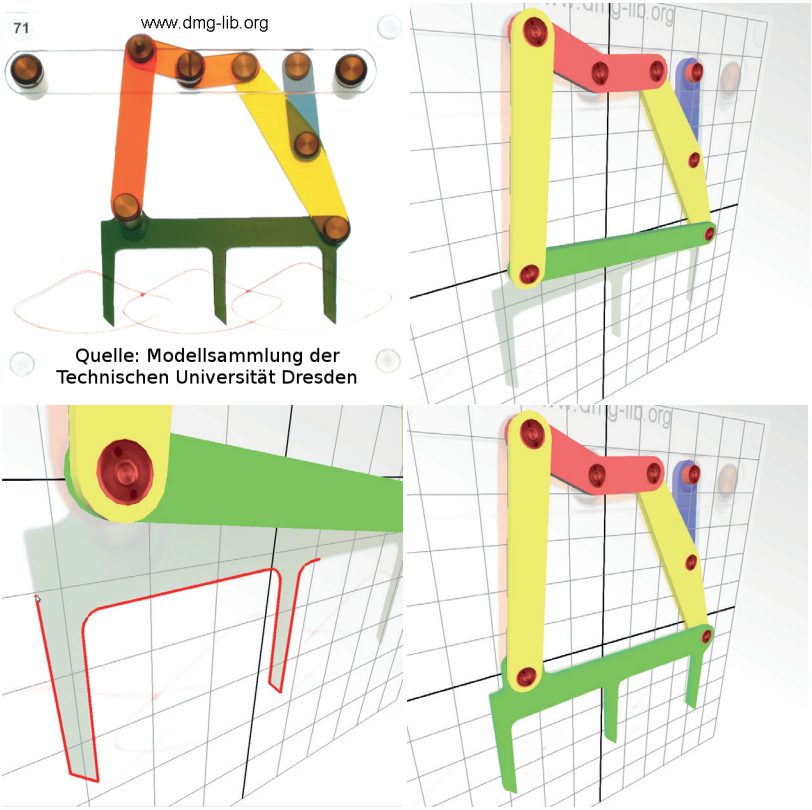


Abbildung 6.26: Erstellung illustrierender Formelemente. Schritt 1: Modellierung der kinematischen Struktur (oben), Schritt 2: Erzeugung der Formelemente aus Polygonen (unten).

6.5.8 Bewegungssimulation

Nach der Modellextraktion und einer eventuellen Anpassung der Visualisierung erfolgt der Simulationsvorgang. Das Ziel besteht in der Erzeugung einer Bewegung über einen bestimmten Antriebsbereich (rotatorisch/translatorisch) und Speicherung (Capture) einer definierten Anzahl von Modellzuständen. Hierfür ist die Abstimmung von Simulationsschrittweite und Antriebsgeschwindigkeit notwendig. Die Ausgabe erfolgt in Form von Videobildern oder Keyframes bzw. Overlays. Für die Berechnung des Bewegungsverhaltens sowie diverser kinematischer Kenngrößen wird auf einen geometrischen Constraint-Solver oder einen Mehrkörpersystem-Solver zurückgegriffen (siehe auch [DBR06]).
Für eine Überlagerung existierenden Videomaterials mit Berechnungsdaten (z. B. kinematische Größen) muss eine zeitliche Synchronisierung des Videoinhalts mit dem Simulationsverlauf durchgeführt werden. Dies kann insbesondere bei Schwankungen in der Geschwindigkeit der Antriebsbewegung zu einem erhöhten manuellen Aufwand führen.

6.5.9 Ausgabedaten und Beispiele

Das primäre Ausgabeformat des Arbeitsablaufs sind Video- oder Bildsequenzen. Diese können sowohl in üblichen Bild- und Videoformaten (z. B. PNG, JPG, AVI, MPEG, Abbildung 6.27) vorliegen, als auch in Form interaktiver Videosequenzen ausgegeben werden. Für letztere wurde eine spezielle Abspielsoftware entwickelt (*AIS-Player, Augmented Image Sequence Player*, Abbildung 6.28), die Mausgesten in Bewegungen auf der Zeitachse transformiert und dem Nutzer den Eindruck vermittelt, dass er den Antrieb des dargestellten Mechanismus mit dem Mauszeiger steuert. Diese Art der Präsentation findet vielfach Anwendung auf der Internetseite des Projektes DMG-Lib. Eine weitere Möglichkeit der Ausgabe sind 3D-Modelle im VRML-Format, die mit einem entsprechenden Browser-Plug-In betrachtet werden können.

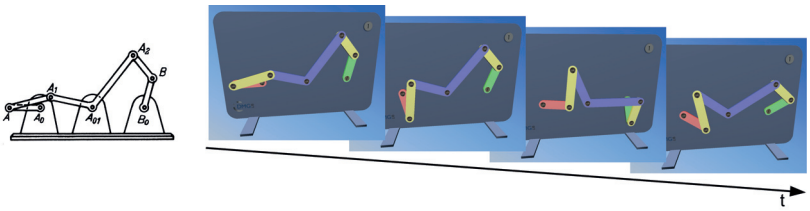


Abbildung 6.27: Aus einer Zeichnung erstellte Animation

Neben den so ausgegebenen Modellzuständen werden während der Simulation gegebenenfalls weitere Informationen berechnet, deren Darstellung als Überlagerung oder Diagramm erfolgen kann (Beispiele: Berechnete Punktbahn oder Einblendung des technischen Prinzips in den Abbildungen 6.28 und 6.29).
Der beschriebene Arbeitsablauf erweist sich dank spezialisierter Werkzeuge als recht schnell und unkompliziert durchführbar. Der manuelle Zeitaufwand pro Modell bewegt sich je nach Komplexität und Visualisierungsstil zwischen ca. 3 und 15 Minuten und erlaubt somit die serienmäßige Erzeugung großer Anzahlen von Animationen.

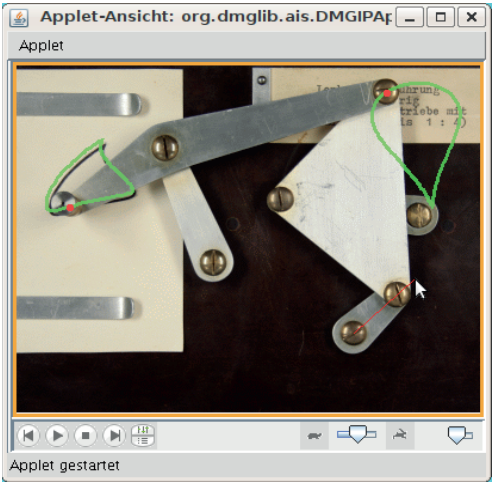


Abbildung 6.28: AIS-Player: Interaktives Realvideo mit berechneten Punktbahnen überlagert

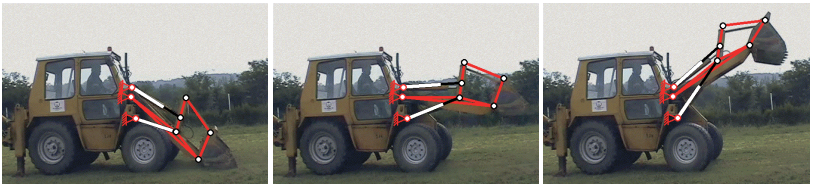


Abbildung 6.29: Überlagerung von Realvideo mit nachmodelliertem technischen Prinzip

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die vorliegende Arbeit entwickelt Lösungen bezüglich zweier Defizite bisher eingesetzter Softwarewerkzeuge zur Unterstützung der frühen Phasen des konstruktiven Entwicklungsprozesses.

7.1.1 Phasenübergreifender Entwurf

Beim ersten Defizit handelt es sich um die mangelnde Unterstützung eines phasenübergreifenden Entwurfs. Diese ist auf die lose oder nicht vorhandene Kopplung der Entwurfswerkzeuge zurückzuführen, die im konstruktiven Entwicklungsprozess eingesetzt werden und mit ihren Modellen ein Produkt auf unterschiedlichen Abstraktionsebenen abbilden. Ein Ziel dieser Arbeit bestand daher in der stärkeren Integration solcher Werkzeuge und ihrer Produktmodelle. Dazu wurde zunächst untersucht, welche Produktmerkmale sich in diesen Modellen phasenübergreifend identifizieren und miteinander in Beziehung setzen lassen. Darauf aufbauend entstand ein Konzept für die constraint-basierte, bidirektionale Kopplung von Produktmodellen der Funktions-, Prinzip- und Gestaltebene.
Vertiefende Betrachtungen wurden bezüglich des Übergangs vom technischen Prinzip zur Grobgestalt angestellt. Dabei wurden gleichzeitig Ansätze zur Rechnerunterstützung der Gestaltfindung entwickelt. Die entstandene softwaretechnische Umsetzung dieser Konzepte koppelt das Entwurfswerkzeug für Bewegungssysteme *MASP* mit einem kommerziellen 3D-CAD-System.

Das entstandene Softwarepaket erlaubt es dem Anwender, aus dem Modell des technischen Prinzips interaktiv eine Grobgestalt im CAD-System zu erarbeiten. Im Rahmen dieses Übergangs zwischen den Abstraktionsniveaus erzeugt das Werkzeug automatisiert Verknüpfungsinformationen. Ergeben sich Änderungen am technischen Prinzip in *MASP* oder am Volumenmodell im CAD-System, so kann mit Hilfe der Verknüpfungen eine Synchronisierung beider Modelle vorgenommen werden, so dass diese inhaltlich konsistent bleiben.
Auf der Grundlage einer solchen Modellkopplung wurde außerdem ein Verfahren erarbeitet, das die Manipulation von CAD-Modellen in VR-Umgebungen erlaubt. Die Umsetzung und Erprobung dieses Verfahrens erfolgte unter Nutzung von *MASP*, des CAD-Systems *SolidWorks* und der VR-Software *Virtual Design 2* in der VR-Einrichtung *FASP* der Technischen Universität Ilmenau.

7.1.2 Domänenübergreifender Entwurf

Das zweite Defizit betrifft das Fehlen nutzerfreundlicher Werkzeuge zum Entwurf technischer Systeme, die sich über mehrere Anwendungsdomänen erstrecken. Im Rahmen der Arbeit wurde daher ein Konzept für eine Softwareplattform erarbeitet und umgesetzt, die den Entwurf und die Simulation heterogener Systeme in frühen Phasen der Produktentwicklung domänenübergreifend ermöglicht. Gleichzeitig bietet das Werkzeug eine nutzerorientierte Mensch-Maschine-Schnittstelle, die Begriffe und Darstellungsformen unterschiedlicher Domänen berücksichtigt und miteinander kombiniert.
Die Entwicklung dieses Werkzeugs erforderte die Lösung einer Reihe von Teilproblemen. Um eine Erweiterbarkeit um nahezu beliebige Domänen und Simulationsverfahren sicherzustellen, musste ein domänenübergreifendes Produktmodell geschaffen werden, das technische Systeme gesamtheitlich und unabhängig von den Erfordernissen einzelner Berechnungsverfahren abbildet. Infolgedessen wurde außerdem die Entwicklung einer Infrastruktur notwendig, die eine aufgabenabhängige Transformation des domänenübergreifenden Gesamtmodells in berechnungsspezifische Modelle ermöglicht. Die zu diesem Zweck eingesetzten Transformationselemente führen überdies Berechnungsergebnisse in das domänenübergreifende Gesamtmodell zurück, so dass dieses als zentrale Kommunikationsinstanz für unterschiedliche Berechnungsverfahren fungiert.
Eine weitere Teilaufgabe stellte die Entwicklung eines ebenfalls erweiterbaren Visualisierungs- und Interaktionssystems für das Entwurfswerkzeug dar. Dieses wurde so gestaltet, dass es dem Ingenieur eine direkte Interaktion mit dem Produktmodell in seiner gewohnten Darstellungs- und Begriffswelt ermöglicht. Zur Implementierung des Werkzeugs war außerdem die Schaffung einer erweiterbaren, flexiblen

Softwarearchitektur notwendig. Insgesamt zeigte sich, dass neben der Betrachtung jedes einzelnen dieser Teilprobleme auch ihre Abstimmung aufeinander für die Umsetzung eines funktionsfähigen, gebrauchstauglichen Werkzeugs unerlässlich ist.
Die Erarbeitung des Konzeptes wurde von einer softwaretechnischen Umsetzung der Plattform begleitet. Dazu erfolgte beispielhaft die Integration von Modellelementen und Simulationsverfahren der Anwendungsdomänen Mechanik, Optik und Regelungstechnik. Anhand verschiedener Beispiele konnte die Gebrauchstauglichkeit eines solchen Werkzeugs belegt werden.

7.2 Ausblick

7.2.1 Integrierte Produktmodelle

Zukünftige Entwicklungen auf dem Gebiet der phasenübergreifenden Modellierung sollten sich nicht auf die bloße Kopplung existierender Softwarewerkzeuge und der darin enthaltenen Modelle beschränken, sondern von Anfang an auf eigens konzipierte phasenübergreifende Modelle aufbauen, die die Informationen aller Entwurfsschritte aufnehmen, über den gesamten Entwicklungsprozess erhalten und miteinander in Beziehung setzen. Diese Idee impliziert tiefgreifende Änderungen an den beteiligten Softwarewerkzeugen und den zugrundeliegenden Konzepten. Vor dem Hintergrund der gelungenen Kopplung der spezialisierten Modelle existierender Werkzeuge erscheint ihre Umsetzung möglich.

7.2.2 Erweiterte CAD-Schnittstelle für MASP

Ein Schwerpunkt der Arbeiten an *MASP* in näherer Zukunft betrifft die Evaluierung der CAD-Schnittstelle des Prototyps bezüglich weiterer CAD-Systeme, auch um Erfahrungen mit der Erstellung und Pflege von Adaptermodulen (siehe Abschnitt 3.6.2) zu sammeln. Weiterhin werden aktuelle Standardisierungsbestrebungen auf dem Gebiet der Schnittstellen und Austauschformate (z. B. AutomationML, [DLPH08]) verfolgt, um die Kopplung von *MASP* mit CAD-Systemen in Zukunft zu vereinfachen.

7.2.3 Integration zusätzlicher Analyse- und Syntheseverfahren in MASP

Ebenfalls von Bedeutung ist die Integration zusätzlicher Analyseverfahren in *MASP*, einschließlich der Entwicklung von Strategien zur Rückführung ihrer Ergebnisse in

den Entwicklungsprozess im Rahmen phasenübergreifender Entwurfsiterationen und Syntheseverfahren. Hierzu gibt es Überlegungen bezüglich einer Verbindung von *MASP* mit dem leistungsfähigen getriebetechnischen Analyse- und Synthesewerkzeug GENESYS [Bre95]. Zur besseren Unterstützung der automatisierten Gestalterzeugung sind die Erstellung von Lösungsspeichern und die Aufbereitung existierender Teilekataloge notwendig. Hierzu müssen in der Praxis einsetzbare Werkzeuge geschaffen werden.

7.2.4 Verallgemeinertes Verfahrens zur Manipulation von CAD-Modellen in VR-Umgebungen

Das im Rahmen dieser Arbeit entwickelte Verfahren zur Manipulation von CAD-Modellen in der Virtuellen Realität basiert auf der Reduktion der Parameter des CAD-Modells auf eine in VR-Umgebungen beherrschbare Anzahl. Die Auswahl der Parameter erfolgt aufgabenabhängig und wird im hier betrachteten Fall auf der Basis eines in *MASP* modellierten und mit dem CAD-Modell gekoppelten technischen Prinzips vorgenommen.
Das Verfahren kann verallgemeinert werden, indem andere Selektionsmethoden für die Parameter des CAD-Modells zum Einsatz kommen (siehe auch Abschnitt 3.7.7). Damit kann eine direkte Kopplung des CAD-Systems mit der VR-Software, ohne MASP als Vermittler, vorgenommen werden. Dennoch ist die Implementierung der Parameterauswahl in einem der zur Kommunikation zwischen dem CAD-System und der VR-Software verwendeten Plug-Ins notwendig.
Die Leistungsfähigkeit des Verfahrens bei der Manipulation umfangreicher CAD-Modelle wurde im Rahmen der Arbeit nicht untersucht. Voraussichtlich sind zusätzliche Maßnahmen erforderlich, die eine interaktive Verarbeitungsgeschwindigkeit sicherstellen. Erfolg verspricht unter anderem eine detaillierte Änderungsverfolgung, um nach Modifikationen des CAD-Modells den Aufwand für die Neuaufbereitung der VR-Darstellung auf geänderte Modellteile zu beschränken.

7.2.5 Einbindung und verbesserte Koordination von Berechnungsverfahren und -werkzeugen

Im Bereich der domänenübergreifenden Modellierung besteht weiterer Forschungsbedarf bezüglich der Koordination der integrierten Berechnungsmodule. Durch die Schaffung einer Koordinationsinstanz mit globaler Sicht auf die Interna der solverspezifischen Teilmodelle könnte beispielsweise die Behandlung zyklischer Abhängigkeiten über die Grenzen der Teilmodelle hinweg gelingen.

Eine andere zweckmäßige Erweiterung von *Hesym* läge in der Schaffung einer Möglichkeit zur Verarbeitung von Modellen, die in einer allgemeinen Modellierungssprache wie *Modelica* formuliert sind. Im Gegensatz zu existierenden Softwarelösungen auf der Basis solcher Sprachen, bei denen domänenspezifische Darstellungen nicht oder nur als reine Visualisierungen verfügbar sind, sollte die Interaktionsphilosophie von *Hesym* zur direkten Arbeit mit dem dargestellten Modell beibehalten werden.

7.2.6 Aufbau und Nutzung von Lösungsrepositorien

Als Entwurfswerkzeug für frühe Phasen dürfte *Hesym* von der Arbeit mit Lösungsrepositorien profitieren. Zu deren Integration müssen Schnittstellen für den direkten Zugriff, Austauschformate sowie alle notwendigen Recherchewerkzeuge geschaffen werden. Weiterhin ist beim Aufbau solcher Sammlungen sicherzustellen, dass die in ihnen gespeicherten Lösungen maschinenlesbar und somit ohne weitere Bearbeitung oder Interpretation in das Entwurfswerkzeug zu übernehmen sind. In diesem Zusammenhang wurden *MASP* und *Hesym* bereits als Werkzeuge zur Extraktion mechanischer Strukturen aus bildlichen Darstellungen eingesetzt (Abschnitt 6.5.5). Die aufbereiteten Lösungen sind Teil der Getriebedatenbank der Digitalen Mechanismen- und Getriebebibliothek DMG-Lib.

7.2.7 Integration des phasen- und domänenübergreifenden Entwurfs

Eine naheliegende Idee, die aus den zwei Schwerpunkten der Arbeit resultiert, ist die Kombination der phasenübergreifenden und der domänenübergreifenden Modellierung. Aus beiden Ansätzen könnte ein Werkzeug entstehen, das den Entwurf heterogener Systeme erlaubt und das gleichzeitig in eine dem konstruktiven Entwicklungsprozess folgende Werkzeugkette integriert ist.

Literaturverzeichnis

[Ala09] *Simulationswerkzeug alaska*, 2009. http://www.tu-chemnitz.de/ifm/ produkte-html/ alaska.html.

[Alg13] *Algodoo - a new way of learning physics*, 2013. http://www.algodoo.com.

[Amb97] S. Ambrosy: *Methoden und Werkzeuge für die integrierte Produktentwicklung*. Dissertation, Technische Universität München, 1997.

[Amf02] M. Amft: *Phasenübergreifende bidirektionale Integration von Gestaltung und Berechnung*. Dissertation, Technische Universität München, 2002.

[Ass08] Modelica Association: *Modelica Standard Library - Tutorial and Reference*, 2008. http://www.modelica.org.

[BBD99] Torsten Brix, Beat Brüderlin und Ulf Döring: *Conditional Constraints in Conceptual Design*. In: *Proceedings of the 32nd International Symposium on Automotive Technology and Automation (ISATA 1999)*. D. Roller, 1999.

[BBD+09] Rike Brecht, Torsten Brix, Ulf Döring, Veit Henkel, Heidi Krömker und Michael Reeßing: *Digital Mechanism and Gear Library - Multimedia Collection of Text, Pictures and Physical Models*. In: Maristella Agosti (Herausgeber): *Research and Advanced Technology for Digital Libraries, Conference proceedings of the ECDL 2009*, Band 5714 der Reihe *Lecture Notes in Computer Science*, Seiten 489–490. Springer, 2009.

[BBD+12] Rike Brecht, Torsten Brix, Ulf Döring, Sascha Falke und Michael Reeßing: *Providing Suitable Information Accesses for New Users of Digital Libraries–Case Study DMG-Lib*. In: *Proceedings of the International Workshop on Supporting Users' Exploration of Digital Libraries (SUEDL)*, Seite 23, Cyprus, 2012.

[BBDH01] Torsten Brix, Beat D. Brüderlin, Ulf Döring und Günter Höhne: *Feature- and Constraint-Based Design of Solution Principles*. In: *Proceedings of the International Conference on Engineering Design, ICED 01*, Seiten 613–620, Glasgow, August 2001.

[BBDH02] Torsten Brix, Beat Brüderlin, Ulf Döring und Günter Höhne: *Representation and Analysis of Solution Principles*. In: *CAD 2002: Corporate Engineering Research*, Seiten 45–56. GI German Informatics Society, 2002.

[BBH+02] Torsten Brix, Beat Brüderlin, Günter Höhne, Michael Reeßing und Gerhard Wolf: *Feature- und constraint-basierte Modellierung auf der Funktions-, Prinzip- und Gestaltebene*. In: *47. Internationales Wissenschaftliches Kolloquium*, Ilmenau, 2002. Technische Universität Ilmenau.

[BCMD06] Torsten Brix, B. Corves, K.-H. Modler und Ulf Döring: *Digitale Mechanismen- und Getriebebibliothek*. In: *VDI-Getriebetagung "Bewegungstechnik"*, Fulda, September 2006.

[BDHR05] Torsten Brix, Ulf Döring, Günter Höhne und Michael Reeßing: *Extraktion technischer Prinzipe und Bewegungssimulation für eine verbesserte technische Dokumentation*. In: *50. Internationales Wissenschaftliches Kolloquium*, Band 2005, Seiten 581–582, Ilmenau, September 2005. Verl. ISLE.

[BDR03] Torsten Brix, Ulf Döring und Michael Reeßing: *Multi-Stage Modeling in the Early Phases of Design*. In: *Proceedings of the 14th International Conference on Engineering Design, ICED 03*, Seiten 279–280, Stockholm, August 2003.

[BDR05] Torsten Brix, Ulf Döring und Michael Reeßing: *Constraint-based Computational Kinematics*. In: *International Workshop on Computational Kinematics*, Cassino, Italien, Mai 2005.

[BDR07] Torsten Brix, Ulf Döring und Michael Reeßing: *The Digital Mechanism and Gear Library as a Tool to Support the Engineering Designer's Work*. In: *Proceedings of the 16th International Conference on Engineering Design, ICED 07*, Paris, France, August 2007.

[BDR08] Torsten Brix, Ulf Döring und Michael Reeßing: *A Web-based Solution Repository in Mechanism Theory to Support the Design Process*. In: *Proceedings of the 10th International Design Conference - DESIGN 2008*, Seiten 583–592, Cavtat, Dubrovnik, Croatia, Mai 2008.

[BDR11] Torsten Brix, Ulf Döring und Michael Reeßing: *Creating Present-Day Solutions from Historical Knowledge*. In: *13th World Congress in Mechanism and Machine Science*, Guanajuato, México, Juni 2011.

[Ber04] Benjamin Berger: *Realisierung einer prototypischen Hardwarelösung für ein inverses Pendel*. Diplomarbeit, Technische Universität Chemnitz, Oktober 2004.

[BKR+04] Tommy Baumann, Ronny Krüger, Michael Reeßing, Beat Brüderlin und Horst Salzwedel: *Computergestützter Konzeptentwurf dynamischer und mechatronischer Systeme mit Regelkreisen*. In: *Synergies between information processing and automation*, Band 2004, Seiten 168–173, Aachen, 2004. Shaker.

[BME+07] Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Jim Conallen, Kelli A. Houston und Bobbi J. Young: *Object-Oriented Analysis and Design with Applications*. Pearson Education, 2007.

[boo13] *The boost Documentation*, 2013. http://www.boost.org/ doc.

[Bow05] Doug A. Bowman: *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2005.

[Bow09] Doug A. Bowman: *3D UIs 101*. In: *ACM Conference on Human Factors in Computing Systems Course Notes*, Boston, 2009. ACM SIG CHI.

[BR98] Beat Brüderlin und D. Roller: *Geometric Constraint Solving and Application*. Springer-Verlag, Berlin, 1998.

[Bre95] C. Breitfeld: *GENESYS - Architektur und Kernrealisierung eines Softwaresystems zur Entwicklung ungleichmäßig übersetzender Getriebe*. Fortschritt-Berichte VDI. VDI-Verlag, 1995.

[Bri01] Torsten Brix: *Feature- und constraint-basierter Entwurf technischer Prinzipe*. Dissertation, Technische Universität Ilmenau, 2001.

[Brü87] Beat Dominik Brüderlin: *Rule-Based Geometric Modelling*. Dissertation, Eidgenössische Technische Hochschule Zürich, 1987.

[Brü93] Beat Brüderlin: *Using Geometric Rewrite Rules for Solving Geometric Problems Symbolically*. Theoretical Computer Science, 116(2):291–303, 1993.

[Brü10] Beat Brüderlin: *Grundlagen der grafischen Interaktion (Vorlesungsskript)*. Fachgebiet Grafische Datenverarbeitung, TU Ilmenau, 2010.

[BS12] Martin Busch und Bernhard Schweizer: *Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit*

coupling approach. Archive of Applied Mechanics, 82(6):723–741, 2012.

[Bus12] Martin Busch: *Zur effizienten Kopplung von Simulationsprogrammen*. kassel university press GmbH, 2012.

[BZB09] Hans-Joachim Bungartz, Stefan Zimmer und Martin Buchholz: *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Berlin: Springer Verlag, 2009.

[CKS03] Volker Carstens, Ralf Kemme und Stefan Schmitt: *Coupled simulation of flow-structure interaction in turbomachinery*. Aerospace Science and Technology, 7(4):298–306, 2003.

[CLR+13] Valentin Ciupe, Erwin-Christian Lovasz, Michael Reessing, Veit Henkel, Corina Gruescu und E.S. Zabava: *Interactive Animation Production by means of Advanced Image Processing*. In: *The 11th IFToMM International Symposium on Science of Mechanisms and Machines - SYROM 2013*, 2013.

[DBR06] Ulf Döring, Torsten Brix und Michael Reeßing: *Application of Computational Kinematics in the Digital Mechanism and Gear Library DMG-Lib*. Mechanism and Machine Theory, 41(8):1003–1015, August 2006.

[DH72] Richard O. Duda und Peter E. Hart: *Use of the Hough transformation to detect lines and curves in pictures*. Commun. ACM, 15(1):11–15, Januar 1972.

[DIN06] *DIN EN 60027-6: Formelzeichen für die Elektronik - Teil 6: Steuerungs- und Regelungstechnik*. Berlin: Beuth-Verlag, 2006.

[DLPH08] Rainer Drath, Arndt Lüder, Jörn Peschke und Lorenz Hundt: *AutomationML - the glue for seamless automation engineering*. In: *ETFA*, Seiten 616–623. IEEE, 2008.

[DMG13] *Web-Portal der Digitalen Mechansimen- und Getriebebibliothek*, 2013. http://www.dmg-lib.org.

[Dör11] Ulf Döring: *Ficucs : ein Constraint-Solver für geometrische Constraints in 2D und 3D*. Dissertation, Technische Universität Ilmenau, 2011.

[Düs01] S. Düsselmann: *Entwicklungsleitsystem Eleit*, 2001. http://tu-berlin.de/ www-kt.

[Dyl02] A. Dyla: *Modell einer durchgängig rechnerbasierten Produktentwicklung*. Dissertation, Technische Universität München, 2002.

[EAA95] K. Ehrlenspiel, S. Ambrosy und G. Aßmann: *Integrierter Konstruktionsarbeitsplatz*. ZWF: Zeitschrift für wirtschaftlichen Fabrikbetrieb, 90(9):410–412, 1995.

[Fel89] Jörg Feldhusen: *Systemkonzept für die durchgängige und flexible Rechnerunterstützung des Konstruktionsprozesses*. Dissertation, Techn. Univ., Diss.–Berlin, 1989, Berlin, 1989.

[FKB03] Markus Färber, Ronny Krüger und Beat Brüderlin: *High-Level-Interaktion in Virtual-Reality-Umgebungen*. In: *Proceedings of 2. Paderborner Workshop: Augmented und Virtual Reality in der Produktentstehung*, Seiten 56–71, 2003.

[FP92] H. J. Franke und M. Peters: *Konstruktionsumgebung MOSAIK – eine grafisch-interaktive Benutzeroberfläche zur Integration von Konstruktionswerkzeugen*, Band 993.3 der Reihe *VDI-Berichte*, Seiten 175–191. Düsseldorf: VDI-Verlag, 1992.

[Fra10] T. Frank: *Konzeption und konstruktive Gestaltung der Messkreise von Nanomessmaschinen*. Dissertation, Technische Universität Ilmenau, 2010.

[Fri06] Peter Fritzson: *Principles of object-oriented modeling and simulation with Modelica 2.1*. IEEE Press, Piscataway, NJ, 2006.

[Gal07] W. O. Galitz: *The Essential Guide to User Interface Design : An Introduction to GUI Design Principles and Techniques*. John Wiley & Sons, 2007.

[GHJV11] E. Gamma, R. Helm, R. Johnson und J. Vlissides: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Programmer's Choice. Addison Wesley Verlag, 2011.

[Hai61] K. Hain: *Angewandte Getriebelehre*. Bücher der Technik. VDI-Verlag, 1961.

[Hau02] Tino Hausotte: *Nanopositionier- und Nanomessmaschine*. Dissertation, Technische Universität Ilmenau, 2002.

[HB97] Ching-yao Hsu und Beat Brüderlin: *A Hybrid Constraint Solver using Exact and Iterative Geometric Constructions*. In: Dieter Roller und Pere Brunet (Herausgeber): *CAD Systems Development. Tools and Methods*. Springer, Berlin, 1997.

[HH93] D. Hix und H. Hartson: *Developing User Interfaces: Ensuring Usability through Product & Process*. John Wiley and Sons, 1993.

[HRWT10] Stephan Husung, Michael Reeßing, Christian Weber und Anita Tröbs: *Verhaltensmodellierung technischer Systeme für audio-visuelle VR-Anwendungen*. In: Michael Grafe Jürgen Gausemeier (Herausgeber): *Proceedings of 9. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung Paderborn 2010*, Seiten 143–156. HNI-Verlagsschriftenreihe, 2010.

[HSR+11] Stephan Husung, Christoph Sladeczek, Michael Rath, Sandra Brix, Torsten Brix und Christian Weber: *Audiovisuelle VR-Simulation am Beispiel einer Pick-and-Place Maschine*. In: *Digitales Engineering und virtuelle Techniken zum Planen, Testen und Betreiben technischer Systeme / IFF-Wissenschaftstage*, Seiten 39–46, Magdeburg, Juni 2011. Stuttgart: Fraunhofer-Verlag.

[HSS83] Günter Höhne, M. Schilling und H. Sperlich: *Neue Erkenntnisse der Konstruktionswissenschaft*. In: *28. Internationales wissenschaftliches Kolloquium*, 1983.

[Hsu96] Ching-Yao Hsu: *Graph-based Approach for Solving Geometric Constraint Systems*. Dissertation, Department of Computer Science, The University of Utah, Salt Lake City (UT), 1996.

[IdC06] Roberto Ierusalimschy, Luiz Henrique de Figueiredo und Waldemar Celes: *Lua 5.1 Reference Manual*. Lua.org, 2006.

[ISO95] *DIN En ISO 9241 Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten, Deutsche Fassung*, 1995.

[iVi04] *Leitprojekt integrierte virtuelle Produktentstehung*, 2004. http://www.ivip.de.

[Jäg08] S. Jäger: *Extraktion von Kurvenscheibengeometrien*. Diplomarbeit, Technische Universität Ilmenau, 2008.

[JF88] Ralph E. Johnson und Brian Foote: *Designing Reuseable Classes*. Journal of Object-Oriented Programming, 1988.

[Joh01] T. Johannsen: *PHAKOM - Phasenübergreifende Unterstützung komplexer Gestaltungs- und Berechnungsprozesse*. http://www.ikmfbs.ing.tu-bs.de/forschungprojekte/projekte/phakom/phakom.html, 2001.

[Kas92] N. Kastrup: *Synthese von Prinziplösungen mit Hilfe eines Katalogsystems - Ein Beitrag zur Konstruktionsmethodik*. Dissertation, RWTH Aachen, 1992.

[KF06] Yvonne Kern und Aimo Fuchs: *Usability Testing der Software Interworks3D/MASP - ein Framework zur Modellierung und Visualisierung dreidimensionaler Objekte*, 2006. Medienprojekt, TU Ilmenau.

[KK98] R. Koller und N. Kastrup: *Prinziplösungen zur Konstruktion technischer Produkte*. Berlin: Springer, 1998.

[KP06] J. G. Korvink und Oliver Paul: *MEMS: A Practical Guide to Design, Analysis, and Applications*. Springer-Verlag, Berlin, 2006.

[Kra02] Werner Krause: *Grundlagen der Konstruktion: Elektronik - Elektrotechnik - Feinwerktechnik*. Fachbuchverlag Leipzig, 2002.

[Kra09] Hardy Krappe: *Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen* . Dissertation, Universität Karlsruhe, 2009.

[Kru03] Philippe Kruchten: *The Rational Unified Process. An Introduction.* Addison-Wesley Longman, Amsterdam, 2003.

[Küm99] M. Kümmel: *Integration von Methoden und Werkzeugen zur Entwicklung von mechatronischen Systemen*. Dissertation, Universität Paderborn, 1999.

[Kun02] H. Kunze: *Ein Beitrag zur Gewinnung von Funktionen für mechanische Produkte aus 3D-CAD-Gestaltmodelldaten*. Dissertation, Universität Karlsruhe, 2002.

[Lac07] Claude Lacoursière: *Ghosts and Machines : Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts*. Dissertation, Umea University, Computing Science, 2007.

[Lic65] Willibald Lichtenheldt: *Konstruktionslehre der Getriebe: Mit 350 Abbildungen und Konstruktionstafeln*. Akademie-Verlag Berlin, 1965.

[Lun08] Jan Lunze: *Regelungstechnik. 1, Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Berlin, Heidelberg: Springer-Verlag, 2008.

[MAS13] *Entwurfswerkzeug MASP*, 2013. http://www.tu-ilmenau.de/ gdv/ forschung/ projekte/ modeling-of-solution-principles.

[Mat13] *Modeling a Piston*, 2013. http://www.mathworks.de/ videos/ modeling-a-piston-68842.html.

[MLS97] H. Meerkamm, C. Löffel und W. Schweiger: *Integration von Berechnungswerkzeugen in den Konstruktionsprozeß – ein ganzheitlicher Ansatz auf Basis des Konstruktionssystems mfk*. Konstruktion, 49(1):26–32, 1997.

[MNK12] R.C. Martin, J.W. Newkirk und R.S. Koss: *Agile Software Development: Principles, Patterns, and Practices*. Alan Apt series. Pearson Education, Limited, 2012.

[Möb06] Frank Möbus: *Konzeption und prototypische Implementierung einer allgemeinen CAD-Schnittstelle für die Software MASP*, 2006. Studienarbeit, Technische Universität Ilmenau.

[Mod13] *Modelica Standard Library - Multibody Examples*, 2013. http://www.maplesoft.com/documentation_center/ online_manuals/ modelica/ Modelica_Mechanics_MultiBody_Examples_Loops.html.

[NA08] M. Nixon und A.S. Aguado: *Feature Extraction & Image Processing*. Feature Extraction and Image Processing Series. Elsevier Science, 2008.

[NH04] James Nutaro und Phil Hammonds: *Combining the model/view/control design pattern with the DEVS formalism to achieve rigor and reusability in distributed simulation*. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, 1(1):19–28, 2004.

[Nii86a] H. Penny Nii: *Blackboard Application Systems and a Knowledge Engineering Perspective*. The AI Magazine, 7(3):82–107, 1986.

[Nii86b] H. Penny Nii: *The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*. The AI Magazine, 7(2):38–53, 1986.

[NP73] Laurence W. Nagel und D.O. Pederson: *SPICE (Simulation Program with Integrated Circuit Emphasis)*. Technischer Bericht UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[Par12] *CADENAS PARTserver - Elektronischer Teilekatalog*, 2012. http://www.partserver.com.

[PB03] G. Pahl und W. Beitz: *Konstruktionslehre*. Berlin: Springer Verlag, 2003.

[Phi04] Hans-Werner Phillippsen: *Einstieg in die Regelungstechnik: Vorgehensmodell für den praktischen Reglerentwurf*. Hanser Fachbuchverlag, 2004.

[Pro07] ProSTEP: *KBE-Methodik: Wissensbasierte 3D-CAD Modellierung*, 2007. http://www.prostep.de/ de/ prostep/ medien/ cad-cam_kbe.htm.

[RB09] Michael Reeßing und Torsten Brix: *Domain-Spanning Design Tools for Heterogeneous Systems*. In: *Proceedings of the 17th International Conference on Engineering Design (ICED'09)*, Band 6, Seiten 393–404, Stanford, 2009.

[RBD07] Michael Reeßing, Torsten Brix und Ulf Döring: *Modeling of Heterogeneous Systems in Early Design Phases*. In: Frank-Lothar Krause (Herausgeber): *The Future of Product Development, Proceedings of the 17th CIRP Design Conference*, Seiten 247–258, Berlin, 2007. Springer.

[RBD11] Michael Reeßing, Torsten Brix und Ulf Döring: *Providing Design Solution Repositories in the Field of Mechanism Theory*. Proceedings of the 18th International Conference on Engineering Design (ICED11), Vol. 6, Seiten 272–281, 2011.

[RDB07] Michael Reeßing, Ulf Döring und Torsten Brix: *Conceptual Design of Heterogeneous systems*. In: *Proceedings of COBEM 2007, 19th International Congress of Mechanical Engineering*, Brasilia, November 2007.

[Ree79] Trygve Reenskaug: *Models - Views - Controllers*. Technical note, Xerox PARC. A scanned version on http://heim.ifi.uio.no/ trygver/mvc/index.html, Mai 1979.

[Rot00] K. Roth: *Konstruieren mit Konstruktionskatalogen*. 3. Aufl., Berlin, Heidelberg, New York: Springer-Verlag, 2000.

[SAKJ98] B. Swienczek, F. Arnold, T. Kilb und A. T. Janocha: *Maßgeschneiderte CAx-Systeme auf Basis von Komponenten*. In: *Tagungsband zum 2. Workshop Konstruktionstechnik: Innovation - Konstruktion - Berechnung*, Kühlungsborn, September 1998.

[Sch00] A. Schön: *Konzept und Architektur eines Assistenzsystems für mechatronische Produktentwicklung*. Dissertation, Universität Erlangen, 2000.

[Sho92] Ken Shoemake: *ARCBALL: a user interface for specifying three-dimensional orientation using a mouse*. In: *Proceedings of the conference on Graphics interface '92*, Seiten 151–156, San Francisco, 1992. Morgan Kaufmann Publishers Inc.

[Shr09] Dave Shreiner: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1.* Addison-Wesley Longman, Amsterdam, 2009.

[SKW92] H. Seifert, D. Kruse und M. Wehmöller: *Netzwerkbasiertes Kommunikationskonzept zur integrierten Produktentwicklung*. VDI-Z, 134(10):89–96, Oktober 1992.

[Smi06] Russell Smith: *Open Dynamics Engine v0.5 User Guide*. http://ode.org/ode-latest-userguide.html, 2006. http://ode.org/ode-latest-userguide.html.

[Spa09] Christiane Spath: *Simulationen - Begriffsgeschichte, Abgrenzung und Darstellung in der wissenschafts- und technikhistorischen Forschungsliteratur*. Magisterarbeit, Universität Stuttgart, 2009.

[Spe03] André Speiser: *Entwicklung von Konzepten für den rechnergestützten Übergang vom technischen Prinzip zum Grobentwurf*, 2003. Projektarbeit, Technische Universität Ilmenau.

[SPSS09] R. Stelzer, D. Petermann, W. Steger und B. Saske: *Kollaborationsumgebung in einer heterogenen CAD-VR Systemlandschaft*. In: J. Gausemeier und M. Grafe (Herausgeber): *Tagungsband zum 8. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung* , Nummer Band 252 in *HNI-Verlagsschriftenreihe*, 2009.

[STL11] *Standard Template Library Programmer's Guide*, 2011. http://www.sgi.com/ tech/ stl.

[Sto07] Michael Stolp: *Modellierung und Export von Mechanismen in der DMG-Lib*. Diplomarbeit, Technische Universität Ilmenau, 2007.

[Stü09] Wolfgang Stürzlinger: *Guidelines for Developing 3D UIs*. In: *ACM Conference on Human Factors in Computing Systems Course Notes*. ACM SIG CHI, 2009.

[SWR+02] Ralph Schönfelder, Gerhard Wolf, Michael Reeßing, Ronny Krüger und Beat Brüderlin: *A pragmatic approach to a VR/AR component integration framework for rapid system setup*. In: *1. Paderborner Workshop: Augmented und Virtual Reality in der Produktentstehung*. Heinz Nixdorf Institut, Juni 2002.

[Tra12] *TraceParts - Teilekatalog*, 2012. http://www.traceparts.com.

[VAC+08] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, T. Kehrer, U. Mehlig und U. Zdun: *Software-Architektur*. Spektrum Akademischer Verlag, 2008.

[van09] Jeroen van der Zijp: *Fox Toolkit Documentation*, 2009. http://www.fox-toolkit.org/ doc.html.

[VDI87] Hg. VDI-Gesellschaft Entwicklung-Konstruktion-Vertrieb (Herausgeber): *VDI Richtlinie 2143, Bewegungsgesetze für Kurvengetriebe, Blatt 1 und 2.* Berlin: Beuth-Verlag, 1987.

[VDI93] Hg. VDI-Gesellschaft Entwicklung-Konstruktion-Vertrieb (Herausgeber): *VDI-Richtlinie 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. VDI-Handbuch Konstruktion.* Berlin: Beuth-Verlag, 1993.

[VDI94] Hg. VDI-Gesellschaft Entwicklung-Konstruktion-Vertrieb (Herausgeber): *Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik. VDI/VDE-Handbuch Mikro- und Feinwerktechnik.* Berlin: Beuth-Verlag, 1994.

[VDI04] Hg. VDI-Gesellschaft Entwicklung-Konstruktion-Vertrieb (Herausgeber): *VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme*. Berlin: Beuth-Verlag, 2004.

[VDI11] Hg. VDI-Gesellschaft Entwicklung-Konstruktion-Vertrieb (Herausgeber): *VDI Richtlinie 2142, Auslegung ebener Kurvengetriebe, Blatt 1 und 2.* Berlin: Beuth-Verlag, 2011.

[Wat06] Robert Watty: *Methodik zur Produktentwicklung in der Mikrosystemtechnik*. Dissertation, Universität Stuttgart, 2006.

[WDL95] Qifeng Wei, W. P. Dayawansa und W. S. Levine: *Nonlinear controller for an inverted pendulum having restricted travel.* Automatica, 31(6):841–850, Juni 1995.

[Web05] Christian Weber: *CPM/PDD - An Extended Theoretical Approach to Modelling Products and Product Development Processes*. In: *Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes*, Seiten 159–179, 2005.

[Web12] Christian Weber: *Produkte und Produktentwicklungsprozesse abbilden mit Hilfe von Merkmalen und Eigenschaften - eine kritische Zwischenbilanz*. In: *23. Symposium Design for X*, Seiten 25–62, Bamberg, Oktober 2012. Hamburg : TuTech Verl.

[Wei08] Martin Weidner: *Prototypische Erweiterung einer allgemeinen CAD-Schnittstelle auf Basis ausgewählter Modellelemente der Software MASP*, 2008. Studienarbeit, Technische Universität Ilmenau.

[Xer13] *Xerces-C++ XML Parser*, 2013. http://xerces.apache.org/ xerces-c/index.html.

[Zwi05] Olaf Zwintzscher: *Software-Komponenten im Überblick: Einführung, Klassifizierung & Vergleich von JavaBeans, EJB, COM+, Net, CORBA, UML 2.* Herdecke; Bochum: W3L-Verl., 2005.

Abbildungsverzeichnis

1.1 Grafische Darstellung der beiden Schwerpunktdimensionen der Arbeit 5

2.1 Konstruktiver Entwicklungsprozess nach VDI 2221 8
2.2 V-Modell der Mechatronik nach VDI 2206 mit domänenspezifischen Entwurfsphasen 9
2.3 Beispiele für Werkzeuge zur Modellierung und Analyse mechanischer Problemstellungen (Kinematik: *SAM*, *Cinderella*, MKS: *SIMPACK*, FEM: *CalculiX*) 11
2.4 Werkzeuge für die Modellierung analoger und digitaler Schaltungen (*Qucs*, *Signs*) 12
2.5 Beispiele für Werkzeuge zur Modellierung hydraulischer und pneumatischer Systeme (*ePlan*, *FluidSim*) 13
2.6 Beispiele für Werkzeuge zur Modellierung optischer Systeme (*ZEMAX*, *CODE V*, *ASAP*) 14
2.7 Werkzeug für den Reglerentwurf (*STEP 7*) 15
2.8 Entwurfsumgebungen auf der Basis von *Modelica* 16
2.9 Bildung des gedanklichen Modells einer Schubkurbel und Umsetzung als Blockschaltbild in der Software Simulink 20
2.10 Umständliche und wenig anschauliche Festlegung von Lageparametern in Simulink 20

3.1 Mögliche Kette zweckmäßig miteinander koppelbarer Modelle der verschiedenen Abstraktionsebenen 26
3.2 Zusammenhänge zwischen Modellen unterschiedlicher Abstraktionsebenen am Beispiel eines zweiseitigen Hebels 28

3.3 Sequentielle Arbeit mit Modellen unterschiedlicher Abstraktionsniveaus 29
3.4 Teilw. parallele Arbeit mit bidirektional verknüpften Modellen . 29
3.5 Parallele Arbeit mit bidirektional verknüpften Modellen 29
3.6 Transformation zwischen nutzerorientierter und constraint-basierter Modellrepräsentation in *MASP* 30
3.7 Benutzeroberfläche von *MASP* 31
3.8 Modelliervorgang in *MASP* am Beispiel einer Viergelenkkette . 32
3.9 Aufbau von Prinzipsymbolen aus geometrischen Grundelementen und Constraints 34
3.10 Konzept für ein phasenübergreifendes Entwurfswerkzeug . . . 36
3.11 Gestaltforderungen von Prinzipelementen und -kopplungen am Beispiel zweier über ein Drehgelenk verbundener Getriebeglieder 37
3.12 Zuordnungsmodell zwischen Prinzip- und Gestaltmerkmalen . 40
3.13 Aufbau eines Zuordnungs- und Gestaltentscheidungselements zur Kopplung mit einem CAD-System 41
3.14 Integration von *MASP* in *COSMOS3000* 43
3.15 Abstrahierte Schnittstelle zur Kopplung von *MASP* mit CAD-Systemen 45
3.16 Übertragung der funktionsrelevanten geometrischen Maße als Hilfselemente 46
3.17 Definition der Bauteilskizze in Abhängigkeit des Skelettes und Austragung zum Volumenkörper 47
3.18 Vorlagenmethode: Bibliothek von Lösungsvarianten für eine Prinzipstruktur 48
3.19 Beispiele für unterschiedliche Strukturen von Getriebegliedern . 49
3.20 Erzeugung unabhängiger Teilkörper und anschließende Verschmelzung mittels boolescher Operationen 50
3.21 Zusammensetzung einer Skizze aus Skizzenbausteinen mit feststehenden Schnittstellen 51
3.22 Automatische Erzeugung von Basisskizzen im CAD-Modell auf der Basis des technischen Prinzips und Beginn der Gestaltmodellierung (manuell) 53
3.23 Detaillierung der Grobgestalt mit den Interaktionsmethoden des CAD-Systems 54
3.24 Übertragung von Änderungen aus dem *MASP*-Modell in das CAD-Modell 54
3.25 Übertragung von Änderungen aus dem CAD-Modell in das Modell von *MASP* 55

3.26 Widersprüchliche Gestaltforderungen 56
3.27 Kopplung von *MASP* mit der VR-Software *VD2* unter Nutzung eines Netzwerkprotokolls 60
3.28 *VD2* als Benutzeroberfläche für *MASP* 61
3.29 Übertragung exportierter CAD-Modelle aus *SolidWorks* in die VR-Software *VD2* 62
3.30 Gesamtaufbau der Kopplung von *MASP*, *VD2* und *SolidWorks* 62
3.31 Beispiel Filmgreifergetriebe in VD2, SolidWorks und MASP . . 64
3.32 Filmgreifergetriebe in *VD2* mit (links) und ohne (rechts) Überlagerung der importierten CAD-Modelle mit Prinzipsymbolen . . 64
3.33 Geänderter Längenparameter des Getriebegliedes 65
3.34 Modellierung eines Mechanismus mit der prototypischen Implementierung der *MASP*-CAD-VR-Kopplung in der *FASP* . . . 66

4.1 Aufbau eines Entwurfswerkzeugs als integrierte, solverspezifische Lösung 71
4.2 Beispiel für Verfahren der geometrischen Optik, abgebildet mit geometrischen Primitiven und Constraints 71
4.3 Modell einer Viergelenkkette in Modelica als Blockdiagramm und als 3D-Visualisierung 73
4.4 Entwurfswerkzeug *MASP* mit Benutzeroberfläche und visualisierter Constraint-Darstellung des Modells (rechts) 76
4.5 Entwurfswerkzeug als Plattform für Berechnungsmodule 77
4.6 Domänenübergreifendes Gesamtmodell als Graph und Zuordnung von Teilmodellen zu Berechnungsmodulen 82
4.7 Modellelemente und Transformationselemente. Grundaufbau (links) und Anwendungsbeispiel (rechts). 84
4.8 Reduktion der Anzahl an Konvertierungen durch die Synchronisierung spezifischer Modellsichten über ein Gesamtmodell als gemeinsame Datenbasis 85
4.9 Beispiel für ein Modellelement mit Transformationselementen . 86
4.10 Mediengrenze als optische Wirkfläche 92
4.11 Domänenübergreifendes Gesamtmodell und solverspezifische Modelle 99
4.12 Transformationselement als Schnittstelle zwischen Modellelement und Berechnungsbibliothek 101
4.13 Aufbau des Transformationselementes 101
4.14 Adapter zur Vereinheitlichung der Schnittstellen von Berechnungsverfahren 103

4.15 Modellierung von Getriebegliedern aus Volumenkörpern in einem CAD-System 109
4.16 Modellierung von Getriebegliedern mit vorgefertigten Bauelementen in *Hesym* 110
4.17 Interaktives „Ausprobieren“ 111
4.18 Ausgaben des „Instruktors“ bei der Modellierung eines Rädergetriebes 112
4.19 Grundaufbau und Umfeld eines Interaktors 115
4.20 Beispiel für die Wiederverwendung von Interaktoren 116
4.21 Unterschiedliche Visualisierungsstile von Modellelementen . . . 117
4.22 Visualisierungselement als Brücke zwischen Modellelement und Renderer 118
4.23 Aufbau eines Visualisierungselementes bei Verwendung eines Szenengraphen 118
4.24 Körperliches Modell einer räumlichen Schubkurbel, Quelle: Getriebesammlung der Technischen Universität Dresden, Modell N7 120
4.25 Positionierung des Drehgelenks auf einer Referenzebene 121
4.26 Umwandlung des Drehgelenks in ein gestellfestes Lager 121
4.27 Positionierung des Kugelgelenks relativ zum Drehgelenk mittels Führungshilfslinien 122
4.28 Ableitung einer Hilfebene aus dem Koordinatensystem des Drehgelenks 122
4.29 Eingabe des Schubgelenkes 122
4.30 Dokumentation des Interaktionszustandes für den Interaktor zur Erzeugung von Schubgelenken 123
4.31 Ableitung einer Hilfsebene aus dem Schubgelenk 123
4.32 Positionierung des zweiten Kugelgelenkes 124
4.33 Eingabe der Getriebeglieder 124
4.34 Veränderter Modellzustand durch interaktive Ssimulation . . . 125
4.35 Grundaufbau des Entwurfswerkzeugs 128
4.36 Bildung von Modellelementen nach dem Vererbungsprinzip . . 129
4.37 Assoziation zwischen Modell- und Erweiterungselementen . . . 130
4.38 Schalter-/Kanal-Kommunikation zwischen Eingabegeräten und Interaktoren 132
4.39 Plug-In-Schnittstelle von *Hesym* 137

5.1 Entwurfswerkzeug für heterogene Systeme *Hesym* 141

5.2 Modulstruktur des Entwurfswerkzeugs mit Basis- und Erweiterungsmodulen 142
5.3 Beispiele für mechanische Modellelemente 145
5.4 Beispiele für optische Komponenten 146
5.5 Beispiele für automatisierungstechnische Komponenten 146
5.6 Beispiele für Interaktoren 148

6.1 Strahlenversatz an einer planparallelen Platte 150
6.2 Beschreibung optischer Zusammenhänge mit Primitiven des geometrischen Constraint-Solvers 151
6.3 Berechnung der Stellung der Messtrommel 151
6.4 Interaktive Vorgabe des Strahlenganges 152
6.5 Optische Achse (links) und positionierte Lichtquelle (rechts) . . 153
6.6 Positionierung der Linse und Einstellung der Blende 154
6.7 Anpassung des Strahlengangs durch Verschiebung des Brennpunktes der Linse in das Zentrum der Lichtquelle 154
6.8 Einfügen des Messspiegels 155
6.9 Auskopplung mittels eines halbtransparenten Spiegel 155
6.10 Positionierung des CCD-Elementes 156
6.11 Modellierung der Verstellmechanik 156
6.12 Feinfühlige Verstellung mittels Koppelgetriebe 157
6.13 Inverses Pendel mit Zustandsregler 158
6.14 Zustandsmaschine des Aufschwingalgorithmus 160
6.15 Messanordnung der Nanopositionier- und Nanomessmaschine (Quelle: SFB 622 der TU Ilmenau) 161
6.16 Geometrisches Ersatzmodell der Messanordnung im CAD-System *SolidWorks* (Quelle: SFB 622 der TU Ilmenau) 162
6.17 Modell im Entwurfswerkzeug für heterogene Systeme *Hesym* . . 163
6.18 Portal der Digitalen Mechanismen- und Getriebebibliothek (Quelle: www.dmg-lib.org) 165
6.19 Arbeitsablauf zur Erzeugung von Animationen aus Bild- und Videomaterial 167
6.20 Beispiele für Eingabedaten bei der Animationserzeugung (Quelle: www.dmg-lib.org) 167
6.21 Manuelle Extraktion einer kinematischen Struktur durch Positionierung von Prinzipsymbolen auf einer Abbildung 168
6.22 Auffinden von Kreisen mittels Hough-Transformation 169
6.23 Finden von Konturen mittels Skelettierung und Segmentierung 170
6.24 XML-Format zur Beschreibung kinematischer Strukturen . . . 170

6.25 Visualisierungsstile am Beispiel des Kreuzlenkers 171
6.26 Erstellung illustrierender Formelemente 172
6.27 Aus einer Zeichnung erstellte Animation 173
6.28 AIS-Player: Interaktives Realvideo mit berechneten Punktbahnen überlagert 174
6.29 Überlagerung von Realvideo mit nachmodelliertem technischen Prinzip 174

Liste der bisher erschienenen Bände, Stand 25.10.2017

Bericht aus dem Institut für Maschinenelemente und Konstruktion (IMK), 1990 – 2010

Band 1 **Institut für Maschinenelemente und Konstruktion der TU Ilmenau (Hrsg.):**
Forschung und Lehre im Institut für Maschinenelemente und Konstruktion (Institutsbericht)
Ilmenau : ISLE, 1999. - ISBN 3-932633-37-7

Band 2 **Spiller, Frank:**
Möglichkeiten der rechentechnischen Umsetzung von Erkenntnissen aus der Konstruktionssystematik unter Nutzung der Featuretechnologie
(Dissertation TU Ilmenau 1998)
Ilmenau : ISLE, 1998. - ISBN 3-932633-20-2

Band 3 **Leibl, Peter:**
Entwicklung eines featureorientierten Hilfsmittels für die Konstruktion kostengünstiger Produkte
(Dissertation TU Ilmenau 1998)
Ilmenau : ISLE, 1998. - ISBN 3-00-003695-4

Band 4 **Lutz, Steffen:**
Kennlinie und Eigenfrequenzen von Schraubenfedern
(Dissertation TU Ilmenau 2000)
Ilmenau : ISLE, 2000. - ISBN 3-932633-47-4

Band 5 **Kletzin, Ulf:**
Finite-Elemente-basiertes Entwurfssystem für Federn und Federanforderungen
(Dissertation TU Ilmenau 2000)
Ilmenau : ISLE, 2000. - ISBN 3-932633-48-2

Band 6 **Volz, Andreas K.:**
Systemorientierter Karosserie-Konzeptentwurf am Beispiel der Crashsimulation
(Dissertation TU Ilmenau 1998)
Ilmenau : ISLE, 2000. - ISBN 3-932633-52-0

Band 7 **Brix, Torsten:**
Feature- und constraint-basierter Entwurf technischer Prinzipe
(Dissertation TU Ilmenau 2001)
Ilmenau : ISLE, 2001. - ISBN 3-932633-67-9

Band 8 **Rektor der TU Ilmenau und Institut für Maschinenelemente und Konstruktion der TU Ilmenau (Hrsg.) in Zusammenarbeit mit Carl Zeiss Jena GmbH**
Vom Arbeitsblatt zum virtuellen Prototyp – 50 Jahre Konstruktionssystematik
(Institutsbericht)
Ilmenau : ISLE, 2002. - ISBN 3-932633-68-7

Band 9 **Liebermann, Kersten:**
Rechnergestütztes Entwurfs- und Optimierungssystem für Schraubendruckfedern
(Dissertation TU Ilmenau 2003)
Ilmenau : ISLE, 2003. - ISBN 3-932633-74-1

Band 10 **Meissner, Manfred; Denecke, Klaus:**
Die Geschichte der Maschinenelemente als Fachgebiet und Institut an der Technischen Universität Ilmenau von 1953 bis 2003
(Institutsbericht)
Ilmenau : ISLE, 2003. - ISBN 3-932633-82-2

Band 11 **Geinitz, Veronika:**
Genauigkeits- und auslastungsoptimierte Schraubendruckfedern
(Dissertation TU Ilmenau 2006)
Ilmenau : ISLE, 2006. - ISBN 3-938843-11-X

Band 12 **Institut für Maschinenelemente und Konstruktion (Hrsg.):**
Festschrift zum Ehrenkolloquium anlässlich der Emeritierungen von Univ.-Prof. Dr.-Ing. habil. Dr. h.c. Günter Höhne und Univ.-Prof. Dr.-Ing. habil. Hans-Jürgen Schorcht
(Institutsbericht)
Ilmenau : ISLE, 2005. -ISBN 3-932633-97-0

Band 13 **Wittkopp, Tobias:**
Mehrkörpersimulation von Schraubendruckfedern
(Dissertation TU Ilmenau 2005)
Ilmenau : ISLE, 2005. - ISBN 3-938843-07-1

Band 14 **Frank, Stefan:**
Justierdrehen – eine Technologie für Hochleistungsoptik
(Dissertation TU Ilmenau 2007)
Ilmenau : ISLE, 2008. - ISBN 978-3-938843-35-4

Band 15 **Schilling, Thomas:**
Augmented Reality in der Produktentstehung
(Dissertation TU Ilmenau 2008)
Ilmenau : ISLE, 2008. - ISBN 978-3-938843-42-0

Band 16 **Lotz, Markus:**
Konstruktion von Messspiegeln hochgenauer Mess- und Positioniermaschinen
(Dissertation TU Ilmenau 2009)
Ilmenau : ISLE, 2009. - ISBN 978-3-938843-46-8

[Band 17] **Hackel, Tobias:**
Grundlegende Untersuchungen zu vertikalen Positioniersystemen für Nanopräzisionsmaschinen
(Dissertation TU Ilmenau 2010)
Münster, Westf : Monsenstein und Vannerdat, 2010. - ISBN 978-3-86991-111-3

[Band 18] **Frank, Thomas:**
Konzeption und konstruktive Gestaltung der Messkreise von Nanomessmaschinen
(Dissertation TU Ilmenau 2010)
Münster, Westf : Monsenstein und Vannerdat, 2010. - ISBN 978-3-86991-194-6

Berichte aus dem Institut für Maschinen- und Gerätekonstruktion (IMGK), 2010 - …

Band 19 **Sondermann, Mario:**
Mechanische Verbindungen zum Aufbau optischer Hochleistungssysteme
(Dissertation TU Ilmenau 2010)
Ilmenau : Univ.-Verl. Ilmenau, 2011. - ISBN 978-3-939473-94-7

Band 20 **Husung, Stephan:**
Simulation akustischer Produkteigenschaften unter Nutzung von Virtual Reality während der Produktentwicklung
(Dissertation TU Ilmenau 2011)
Ilmenau : Univ.-Verl. Ilmenau, 2012. - ISBN 978-3-86360-026-6

Band 21 **Dobermann, Dirk:**
Stabilisierung der Bildlage abbildender optischer Systeme
(Dissertation TU Ilmenau 2012)
Ilmenau : Univ.-Verl. Ilmenau, 2013. - ISBN 978-3-86360-056-3

Band 22 **Taubmann, Peter:**
Analyse der Ventilfederbewegung als Beitrag zur Beeinflussung der Verschleißursachen an den Auflageflächen
(Dissertation TU Ilmenau 2013)
Ilmenau : Univ.-Verl. Ilmenau, 2013. - ISBN 978-3-86360-059-4

Band 23 **Erbe, Torsten:**
Beitrag zur systematischen Aktor- und Aktorprinzipauswahl im Entwicklungsprozess
(Dissertation TU Ilmenau 2013)
Ilmenau : Univ.-Verl. Ilmenau, 2013. - ISBN 978-3-86360-060-0

Band 24 **Ginani, Luciano Selva:**
Optical Scanning Sensor System with Submicron Resolution
(Dissertation TU Ilmenau 2013)
Ilmenau : Univ.-Verl. Ilmenau, 2013. - ISBN 978-3-86360-068-6

Band 25 **Heidler, Nils:**
Untersuchungen zylindrischer Gasführungselemente für Hochvakuumanwendungen
(Dissertation TU Ilmenau 2015)
Ilmenau : Univ.-Verl. Ilmenau, 2016. - ISBN 978-3-86360-130-0

Band 26 **Reich, René:**
Möglichkeiten und Grenzen bei der Auslegung von Schraubendruckfedern auf Basis von Umlaufbiegeprüfungen
(Dissertation TU Ilmenau 2016)
Ilmenau : Universitätsverlag Ilmenau, 2016. - ISBN 978-3-86360-139-3

Band 27 **Resch, Jens:**
Kontextorientierte Entwicklung und Absicherung von festen Verbindungen im Produktentstehungsprozess der Automobilindustrie
(Dissertation TU Ilmenau 2016)
Ilmenau : Universitätsverlag Ilmenau, 2016. - ISBN 978-3-86360-143-0

Band 28 **Scheibe, Hannes:**
Aktiv-adaptive Polierwerkzeuge zur Herstellung rotationssymmetrischer Asphären
(Dissertation TU Ilmenau 2016)
Ilmenau : Universitätsverlag Ilmenau, 2016. - ISBN 978-3-86360-147-8

Band 29 **Reeßing, Michael:**
Softwarewerkzeuge für den phasen- und domänenübergreifenden Entwurf
(Dissertation TU Ilmenau 2016)
Ilmenau : Universitätsverlag Ilmenau, 2017. - ISBN 978-3-86360-169-0