Ein Beitrag zur rechnerunterstützten Merkmalsgewinnung und -verknüpfung aus Produktdatenmodellen zur Erkennung und Auswertung von Abhängigkeiten für die integrierte Produktentwicklung

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

an der Fakultät für Maschinenbau der
Universität Karlsruhe (TH)
genehmigte

DISSERTATION

von

Dipl.-Ing. Oliver Klaar

Aus:	Karlsruhe
Tag der mündlichen Prüfung:	23. Juni 2005
Hauptreferent:	Prof. em. Dr.-Ing. Prof. E.h. Dr. h.c. Grabowski
Koreferent:	Prof. Dr.-Ing. Christian Weber

Vorwort des Herausgebers

Der Maschinenbausektor ist in Deutschland nach wie vor einer der stärksten und gleichzeitig innovativsten Wirtschaftszweige; die Technologie steht international an der Spitze. Dabei entstehen Innovationen häufig durch die Integration von Technologien aus anderen Ingenieursdisziplinen, wie beispielsweise Elektronik oder Informatik. Bedingt durch die fortschreitende Integration von Technologien aus unterschiedlichen Disziplinen entstehen immer komplexere Systeme, die durch mannigfaltige, domänenübergreifende Abhängigkeiten zwischen den Produktkomponenten und ihren Eigenschaften gekennzeichnet sind.

Bei der Entwicklung solcher Produkte legen Anforderungen die Bedingungen fest, denen die Eigenschaften eines zu entwickelnden Produkts und seiner Komponenten zu genügen haben. Dadurch sind Anforderungen und Eigenschaften eng miteinander verknüpft. Anforderungen müssen daher im Mittelpunkt jeder Produktentwicklung stehen. Parallel zu der steigenden Komplexität von Produkten nimmt die für die Produktentwicklung zur Verfügung stehende Zeit durch die zunehmende internationale Konkurrenz in diesem Sektor stetig ab. Dadurch entsteht ein Spannungsfeld, das durch herkömmliche Methoden schwer aufzulösen ist. Die Folge daraus sind Qualitätsprobleme, deren Auswirkungen in vielen Bereichen bereits die Rentabilität von Unternehmen gefährden.

In den einzelnen Ingenieursdisziplinen werden spezialisierte Softwarewerkzeuge für die Produktmodellierung eingesetzt, in denen die geplanten Eigenschaften der Produktkomponenten in geeigneten Produktmodellen definiert und abgebildet werden. Der notwendige Datenaustausch zwischen den Softwarewerkzeugen erfolgt vorwiegend über standardisierte Datenschnittstellen. Er ist häufig nicht zufriedenstellend.

Aufbauend auf bekannten Ansätzen aus der Forschung wird in der vorliegenden Arbeit ein Konzept vorgestellt, mit dessen Hilfe Abhängigkeiten zwischen Produkteigenschaften, die aus unterschiedlichen Ingenieursdisziplinen und Produktmodellen stammen, automatisch ermittelt und erkannt werden können. Dieser Weg der Interpretation von Daten aus unterschiedlichen systemeigenen Datenmodellen ist eine vielversprechende Forschungsrichtung. Sie schafft eine Unabhängigkeit der Integration verschiedener Softwarewerkzeuge von der Geschwindigkeit des Standardisierungsprozesses von Datenmodellen. Für die Industrie ergibt sich dadurch eine Erleichterung bei der Auswahl von CAX-Systemen und deren Kopplung entlang des Entwicklungs- und Herstellungsprozesses.

Hans Grabowski

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung und Motivation 1
2 Einführung, Stand der Technik 5
2.1 Begriffsdefinitionen und -Erläuterungen 5
2.1.1 Die Grundlagen der Konstruktionsmethodik 5
2.1.2 Merkmale, Eigenschaften, Bedingungen und Anforderungen 8
2.1.3 Anforderungsgetriebene, integrierte Produktentwicklung 11
2.1.3.1 Ansatz zur Anforderungsmodellierung des Instituts für Rechneranwendung in Planung und Konstruktion (RPK) 12
2.2 Ansätze zur Produktdatenintegration 14
2.2.1 Kommerzielle Lösungen 15
2.2.1.1 Produktdatenmanagementsysteme, Product-Lifecycle-Management-Systeme 16
2.2.1.2 UGS Teamcenter 18
2.2.1.3 Das Produktmodell von STEP 19
2.2.2 Ansätze aus der Forschung 19
2.2.2.1 Das integrierte Produkt- und Produktionsmodell (PPM) 19
2.2.2.2 Der Metamodell-Ansatz aus DRAGON 20
2.2.2.3 Der Ansatz aus FIPER 24
2.2.2.4 Pan Galactic Engineering Framework (PGEF) 25
2.3 Systeme zur Abbildung von Abhängigkeiten zwischen Produkteigenschaften 26
2.3.1 Computer Aided Design (CAD) Werkzeuge 26
2.3.1.1 Geometrisch-Topologische Strukturmodelle (GT- bzw. B-Rep Modelle) 27
2.3.2 System Engineering (SE) Werkzeuge 31
2.3.2.1 System Modeling Language (SysML) 32
2.3.2.2 Das System RODON® der Firma R.O.S.E Informatik 35
2.3.3 Ansätze aus der Forschung 37
2.3.3.1 Ermittlung von Produktfunktionen aus 3D-CAD-Gestaltmodellen 37
2.3.3.2 Der Ansatz aus iViP 38
2.3.3.3 Property Driven Development/Design (PDD) 40
2.3.3.4 Der Ansatz aus dem Design Repository Project (DRP) 41
2.3.3.5 Konfigurations- und Verträglichkeitsmatrix (K-V-Matrix) nach Bongulielmi 42
2.4 Fazit 45
3 Konzept 47
3.1 Anforderungen an ein System zur anforderungsgetriebenen, integrierten Produkt-entwicklung 47
3.1.1 Verknüpfung von korrespondierenden Produktmerkmalen 47
3.1.2 Abbildung und Erkennung von Abhängigkeiten zwischen Produktmerkmalen 48
3.1.3 Anforderungen an die Flexibilität der Systemanbindung 48

3.2 Merkmalsgewinnung aus externen Produktmodellierungssystemen 49
3.2.1 Produktmerkmalsbibliotheken 51
3.2.2 Sichten auf die Gesamtproduktstruktur 51
3.2.3 Beziehungen zwischen Produktstrukturkomponenten 51
3.2.4 Vorgehensweise zur Identifikation und Organisation korrespondierender Produktteilsysteme aus unterschiedlichen Produktmodellierungswerkzeugen 52
3.2.4.1 Externes System steuert SAEP 53
3.2.4.2 SAEP steuert externes System 53
3.2.4.3 Korrespondierende Produktstrukturen in SAEP und externem System 54
3.2.5 Vorgehensweise zur Verknüpfung von Produktkomponenten unterschiedlicher Systeme 55
3.2.5.1 Automatische Analyse der Modellinhalte von Produktmodellierungswerkzeugen und SAEP 56
3.2.5.2 Schablonen zur Integration von Modellinhalten von Produktmodellierungswerkzeugen und SAEP 56
3.2.5.3 Manuelle Integration von Modellinhalten von Produktmodellierungswerkzeugen und SAEP 58
3.2.6 Verknüpfung von Merkmalen 59
3.2.6.1 Darstellung und Analyse von Maßeinheiten 60
3.3 Erkennung von Abhängigkeiten zwischen Merkmalen 61
3.3.1 Abbildung von Abhängigkeiten 63
3.3.2 Arten von Abhängigkeiten 64
3.3.2.1 Abstrakte Abhängigkeiten zwischen Produktmerkmalen 64
3.3.2.2 Mathematisch beschreibbare Abhängigkeiten zwischen Produktmerkmalen (Mathematische Abhängigkeiten) 65
3.3.3 Vorgehensweise zur Ermittlung von Abhängigkeiten zwischen Produktmerkmalen 66
3.3.3.1 Manueller Ansatz 66
3.3.4 Abbildung von Abhängigkeiten in Schablonen 67
3.3.5 Automatische Erkennung von Mathematischen Abhängigkeiten 68
3.3.5.1 Wirkräume und Kontaktsysteme 70
3.3.5.2 Abbildung von Wirkräumen in Schablonen 74
3.3.5.3 Kontaktsystembibliotheken 74
3.4 Fazit 75
4 Implementierung 77
4.1 Datenmodell 78
4.1.1 Abbildung von Produktmerkmalen 79
4.1.2 Abbildung von Produktmerkmalsbibliotheken 82
4.1.3 Abbildung und Verknüpfung von Produktstrukturelementen 83
4.1.4 Verknüpfung von Produktmerkmalen und Produktstrukturelementen 88
4.1.5 Abbildung von Abhängigkeiten zwischen Merkmalen 89
4.1.5.1 Nicht-mathematisch beschreibbare Abhängigkeiten 89
4.1.5.2 Mathematisch beschreibbare Abhängigkeiten zwischen Merkmalen 91
4.1.6 Abbildung von Wirkräumen und Kontaktsystemen 92

4.2 Basisklassen für die Darstellung nichtstrenger Hierarchien ... 94
4.3 Persistenzschicht ... 96
4.3.1 Persistierung der in SAEP erzeugten Informationen ... 96
4.3.2 Zugriff auf externe Autorensysteme ... 100
4.3.2.1 Die CADServices-Schnittstelle ... 101
4.4 Integrationsschicht ... 103
4.4.1 Initialisierung der Integrationsschicht ... 104
4.4.2 Laden und Speichern von Modellen ... 105
4.4.3 Verknüpfung korrespondierender Merkmale ... 106
4.4.3.1 Manueller Ansatz zur Verknüpfung von Modellentitäten ... 106
4.4.3.2 Schablonenbasierter Ansatz zur Verknüpfung von Merkmalen ... 107
4.4.3.3 Automatisierter Ansatz zur Verknüpfung von Merkmalen ... 107
4.5 Anwendungslogik ... 107
4.5.1 Auswertung von mathematischen Gleichungssystemen ... 108
4.5.2 Wirkraummanager (EffectSpaceManager) ... 112
4.5.2.1 Initialisierung des Wirkraummodells ... 113
4.5.2.2 Auswertung von Kontaktsystemen ... 117
4.5.2.3 Verwaltung von Kontaktsystemen ... 117
4.5.3 Beziehungsnetzwerkmanager (Relation Network Manager) ... 118
4.5.4 Der Attribute Library Manager ... 121
4.6 Fazit ... 122
5 Verifikation ... 125
5.1 Definition von Produktmerkmalen ... 126
5.1.1 Verwaltung von Produktanforderungen in Merkmalsbibliotheken ... 126
5.1.2 Ausprägung von Merkmalen ... 129
5.1.3 Modellierung von Produktstrukturen ... 130
5.2 Verknüpfung von Produktstrukturelementen und deren Merkmalen ... 133
5.2.1 Verknüpfung von korrespondierenden Produktkomponenten ... 134
5.2.2 Verknüpfung von korrespondierenden Produktmerkmalen ... 138
5.3 Modellierung von Abhängigkeiten ... 141
5.3.1 Manuelle Modellierung von Abstrakten Beziehungen ... 141
5.3.2 Manuelle Modellierung und Auswertung von mathematisch beschreibbaren Abhängigkeiten zwischen Produktmerkmalen ... 143
5.3.3 Verwaltung, Modellierung und Auswertung von Kontaktsystemen ... 147
5.3.3.1 Verwaltung von Kontaktsystemen ... 147
5.3.3.2 Modellierung von Kontaktsystemen ... 148
5.3.3.3 Auswertung von Kontaktsystemen ... 157
5.4 Bewertung und Ausblick ... 159
6 Zusammenfassung ... 163
Literatur ... 169
Abkürzungsverzeichnis ... 175

1 Einleitung und Motivation

Im Zuge des Trends zu immer kürzer werdenden Produktdurchlaufzeiten, der seit vielen Jahren verstärkt beobachtet werden kann, sind die Anforderungen an die Effizienz von Produktentwicklungsprozessen drastisch gestiegen. Diese Problematik wird durch die schnell wachsende Komplexität der Produkte, die zunehmend auch Erzeugnisse außerhalb des eigentlichen Hochtechnologiesektors betrifft[1], überlagert und verstärkt. Besonders bemerkbar macht sich diese Entwicklung in der Automobilbranche. Moderne Kraftfahrzeuge vereinen unter anderem Komponenten aus den Bereichen Mechanik, Elektrik, Elektronik und Informatik, die eng miteinander verzahnt sind. In modernen Mittelklasse-PKW werden beispielsweise durchschnittlich 20-60 Controller[2] eingesetzt [Bosc-2003], die mit anderen Controllern kommunizieren und über elektrische Aktoren mechanische oder hydraulische Komponenten beeinflussen können.

Die Integration von Technologien aus unterschiedlichen Ingenieursdisziplinen, wie Mechanik, Elektronik und Informatik bedingt eine hohe Komplexität moderner, technischer Erzeugnisse. Die dadurch entstehenden Abhängigkeiten zwischen den verschiedenen Domänen zuzuordnenden Produktmerkmalen sind nur schwer beherrschbar. Änderungen an einzelnen produktbeschreibenden Parametern haben oft eine Vielzahl von Auswirkungen auf andere Produkteigenschaften, die oft nicht erkannt werden und erst bei der Produktnutzung zutage treten. Dies ist immer verbunden mit hohen Kosten für den Hersteller und, im schlimmsten Fall, mit Schäden am Image des betroffenen Unternehmens. Beides manifestiert sich bei Rückrufaktionen. Ein namhafter, deutscher Automobilhersteller hat beispielsweise im Jahr 2005 im Rahmen einer sogenannten „Qualitätsoffensive“, seinen Kunden angeboten, die erst im Produktgebrauch erkannten Mängel zu beheben. Betroffen davon sind 1,3 Mio Fahrzeuge verschiedener Baureihen. Die Kosten für diese Aktion wurden von Unternehmensseite bisher nicht öffentlich genannt, werden von Analysten aber auf einen dreistelligen Millionenbetrag geschätzt [WiWo-2005]. Der Imageschaden durch die häufig aufgetretenen Mängel ist ebenfalls beträchtlich. Die konstruktionsbedingten Fehler betrafen Komponenten, die Technologien aus unterschiedlichen Domänen enthalten. So war unter anderem das elektrohydraulische Bremssystem, der Spannungsregler der Lichtmaschine sowie die Software des Batteriesteuergeräts mehrerer Baureihen betroffen [WiWo-2005].

In der Maschinenbaubranche haben sich 3D CAD-Systeme als zentrales Werkzeug für die Produktentwicklung durchgesetzt. Mit ihrer Hilfe wird das fertige Produkt vorausgedacht und als virtuelles, rechnerinternes Modell abgebildet. CAD-Modelle beschreiben vornehmlich die Gestalt des fertigen Produkts. Es können aber auch Materialeigenschaften, wie die Dichte des verwendeten Werkstoffs definiert und für Masseberechnungen herangezogen werden. Dieses Produktmodell ist die zentrale Informationsquelle für alle nachgelagerten Planungs- und Fertigungsprozesse.

1 Beispiele dafür sind Waschmaschinen mit Infrarotschnittstellen für Fehlerdiagnose und Waschprogramm-Updates oder Haushalts-Kaffeemaschinen mit Internetanschluss für Firmware-Updates.

2 Als *Controller* werden elektronische Einheiten bezeichnet, die Prozesse steuern oder regeln [DiOh-1994].

Darüber hinaus haben sich auch in anderen Ingenieurswissenschaften rechnerunterstützte Werkzeuge etabliert. Für den Entwurf von elektrischen und elektronischen Schaltungen sind E-CAD-Systeme verfügbar, in denen Schaltungen entworfen und getestet werden können. Für den Bereich Funktionsmodellierung und Simulation komplexer Systeme werden System Engineering Werkzeuge eingesetzt. Im Bereich der Softwareentwicklung sind Programmierumgebungen im Einsatz, mit denen die Steuer- und Regelungssoftware für die informationstechnischen Komponenten, wie z.B. Controller oder Infotainmentsysteme eines Automobils entworfen wird.

Die Definition und Dokumentation von Produktanforderungen, die die Entwicklungsaufgabe selbst, sowie die zu beachtenden Randbedingungen beschreiben, erfolgt in den meisten Fällen mit Hilfe von Pflichten- und Lastenheften sowie Anforderungslisten, die mit Hilfe herkömmlicher Textverarbeitungs- und Tabellenkalkulationsprogramme erstellt werden.

Der Datenaustausch zwischen den spezialisierten Produktmodellen erfolgt meist über neutrale Dateiformate wie z.B. STEP[3], wobei die Schnittmenge der gemeinsam genutzten Informationen aufgrund der unterschiedlichen Modellierungsaspekte der beteiligten Werkzeuge als eher gering eingestuft werden kann. Für die Verwaltung der Informationsmengen haben sich in der Industrie Produktdatenmanagementsysteme (PDM-Systeme) und Product Lifecycle Management-Systeme (PLM-System) etabliert. Mit ihrer Hilfe werden die Dateien, in denen die Informationen der Modellierungswerkzeuge dauerhaft gespeichert werden, verwaltet.

Die bisher übliche Vorgehensweise zur Produktmodellierung mit Hilfe spezialisierter aber isolierter Werkzeuge und die Evaluierung von Produkteigenschaften ohne Rechnerunterstützung, beispielsweise anhand von Lastenheften, Pflichtenheften und Anforderungslisten, stößt bei komplexen, technischen Produkten zunehmend an ihre Grenzen. Notwendig ist eine rechnerinterne Verknüpfung von Produkteigenschaften und Anforderungen, um Konstruktionslösungen ohne Zeitverluste evaluieren zu können, sowie die rechnerunterstützte Beherrschung von Abhängigkeiten zwischen den Produkteigenschaften aus verschiedenen Ingenieurdisziplinen und Produktmodellierungswerkzeugen.

Motivation und Zielsetzung

Bisher verfügbare kommerzielle Systeme und Lösungsansätze aus der Forschung werden den oben genannten Herausforderungen nicht gerecht. Dies führte zu der Entwicklung eines Systems zur **rechnerunterstützten Merkmalsgewinnung und -verknüpfung aus Produktdatenmodellen zur Erkennung und Auswertung von Abhängigkeiten für die integrierte Produktentwicklung**.

Dieses System soll es dem Anwender ermöglichen, die in seinem gewohnten Produktmodellierungswerkzeug festgelegten Produkteigenschaften anhand von Anforderungen automatisch zu evaluieren, und das parallel zum Entwicklungsprozess. Darüber hinaus

3 STEP: Standard for the Exchange of Product Model Data

werden Methoden entwickelt und implementiert, die die korrekte Zuordnung von Anforderungen und Produkteigenschaften automatisieren und so den Aufwand für die Evaluierung von Produkteigenschaften minimieren.

Um die oben beschriebene Komplexität moderner, technischer Produkte zu beherrschen, muss das System eine ganzheitliche Sicht auf das Produkt ermöglichen. Die Auswirkungen von Änderungen von Produkteigenschaften auf andere Bereiche des Produkts müssen erkannt, verfolgt und, wenn möglich, quantifiziert werden. Daher werden Methoden entwickelt, die Abhängigkeiten zwischen Produktmerkmalen in Produktdatenmodellen aus unterschiedlichen Domänen abbilden, verfolgen und auswerten. Darüber hinaus werden Methoden entwickelt, die implizit gegebene Abhängigkeiten zwischen Produktmerkmalen, beispielsweise aufgrund physikalischer Zusammenhänge, erkennen, explizit darstellen und quantifizieren.

Grundvoraussetzung für die oben beschriebene Zielsetzung ist die Integration der Produktdatenmodelle der eingesetzten Produktmodellierungswerkzeuge in einem separaten Integrationssystem. Dieses verknüpft die Modellinhalte der externen Werkzeuge mit den Modellinhalten der Anwendungslogik des zu entwickelnden Systems. Das Integrationssystem wird so ausgelegt, dass es zum einen flexibel ist, was die abzubildenden Modellinhalte angeht, da nicht vorhergesehen werden kann, welche Produktaspekte und welche Arten von Produktdatenmodellen im späteren Einsatz des Systems integriert werden müssen. Zum anderen werden verschiedene Methoden für den Zugriff auf externe Modellinhalte unterstützt. Bevorzugt wird die direkte Anbindung über Programmierschnittstellen, da dies einen Online-Zugriff auf die Produktdatenmodelle ermöglicht und die Gefahr von Inkonsistenzen der beteiligten Modellinhalte minimiert. Eine Programmierschnittstelle ist aber nicht bei allen im industriellen Einsatz befindlichen Werkzeugen verfügbar. Daher wird das Integrationssystem so ausgelegt, dass es die externen Modellinhalte der darauf zugreifenden Anwendungslogik des zu entwickelnden Systems transparent zur Verfügung stellt.

2 Einführung, Stand der Technik

In diesem Kapitel werden zunächst die dem Konzept der vorliegenden Arbeit zugrunde liegenden Begriffswelten und Ansätze erläutert sowie bestehende Ansätze in den Gebieten Produktdatenintegration und Abbildung und Auswertung von Abhängigkeiten zwischen Produktmerkmalen im Bereich der Forschung und der kommerziell verfügbaren Systeme vorgestellt und bewertet.

2.1 Begriffsdefinitionen und -Erläuterungen

2.1.1 Die Grundlagen der Konstruktionsmethodik

Ziel des Konstruktionsprozesses ist - vereinfacht gesagt - ausgehend von einer Aufgabenstellung alle notwendigen Informationen zu erzeugen und zusammenzutragen, die für die Herstellung eines Produkts notwendig sind.

Die kreative und damit zunächst unstrukturierte und informelle Tätigkeit der Produktentwicklung wird durch die Mittel der Konstruktionsmethodik auf ein theoretischeres Fundament gestellt. Mit den Ansätzen der Konstruktionsmethodik sollen die in der Produktentwicklung ablaufenden Prozesse erforscht und dargestellt werden, um daraus lehr- und erlernbare Verfahren und Vorgehensweisen abzuleiten, die dem Produktentwickler helfen sollen, sowohl intuitive als auch diskursive Lösungen für Problemstellungen zu finden. Wesentlich dabei ist auch, dafür zu sorgen, dass die Problemstellung vollständig bekannt ist und gelöst wird. Das bedeutet, dass alle relevanten Anforderungen, die sich aus den der Produktentwicklung nachgelagerten Phasen ergeben, erkannt, dokumentiert und bei der Lösungsfindung berücksichtigt werden.

Der Lebenslauf eines Produkts lässt sich allgemein in die Phasen *Produktentstehung*, *Produktvertrieb*, *Produktnutzung* und *Produktreycling* einteilen. Die Phase der Produktentstehung lässt sich wiederum in die Phasen *Produktplanung*, *Konstruktion* (oder allgemeiner formuliert: *Produktentwicklung*), *Arbeitsvorbereitung* und *Produktherstellung* untergliedern [Rude-1998]. Aus allen der Konstruktion nachgelagerten Phasen des Produktlebenslaufs ergeben sich Anforderungen (die „Gerechtigkeiten“), die während der Konstruktion berücksichtigt werden müssen. Abbildung 1 verdeutlicht diese Zusammenhänge.

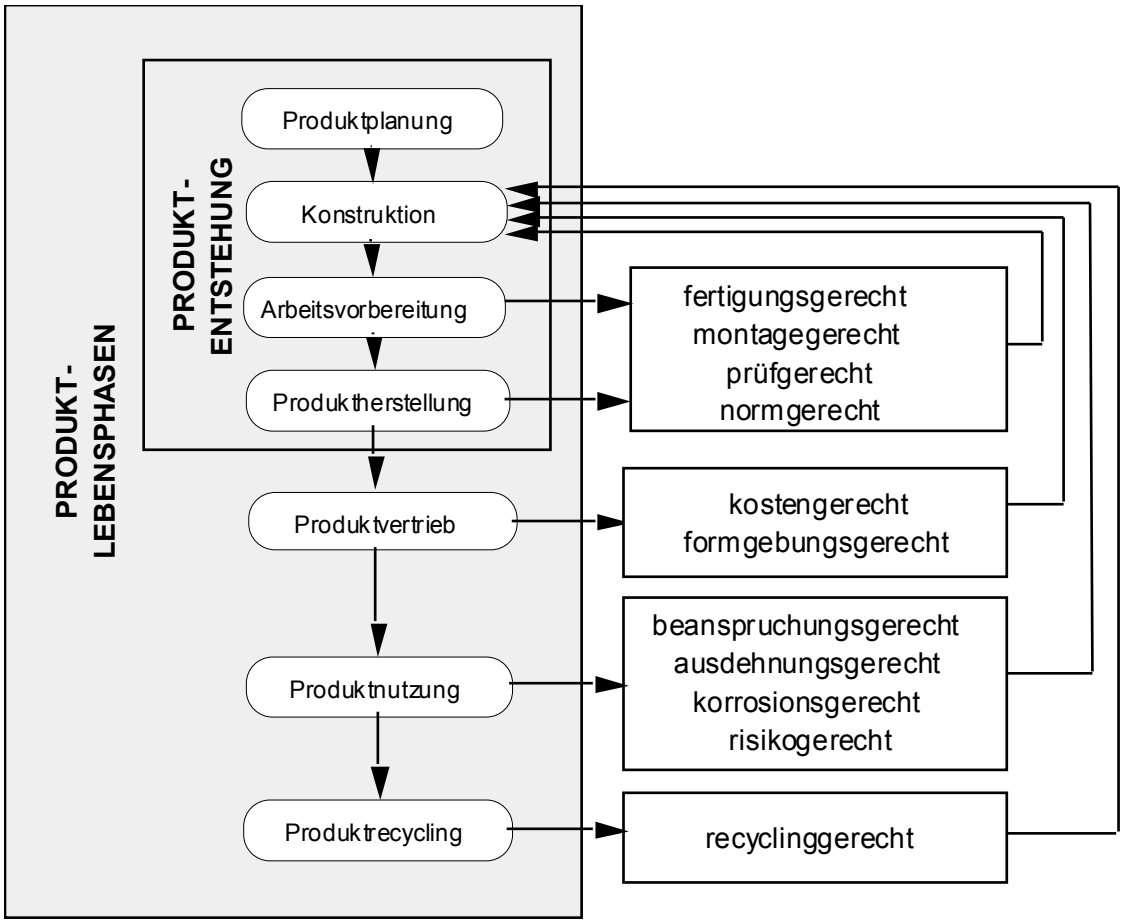


Abbildung 1: Lebensphasen eines Produkts [Rude-1998]

Die Konstruktionsmethodiken aus dem deutschsprachigen Raum, wie nach PAHL/BEITZ [PaBe-1997], ROTH [Roth-1994], EHRLENSPIEL [Ehrl-1995], RODENACKER [Rode-1991], KOLLER [Koll-1985] und HUBKA [Hubk-1984], die ihren Niederschlag in der VDI-Richtlinie 2221 gefunden haben, unterteilen die Phase der Konstruktion in weitere Phasen, die in der Regel iterativ durchlaufen werden. Abbildung 2 zeigt die Arbeitsschritte und ihre korrespondierenden Arbeitsergebnisse sowie ihre Zuordnung zu Konstruktionsphasen, wie sie in VDI-Richtlinie 2221 dargestellt sind [VDI-1993].

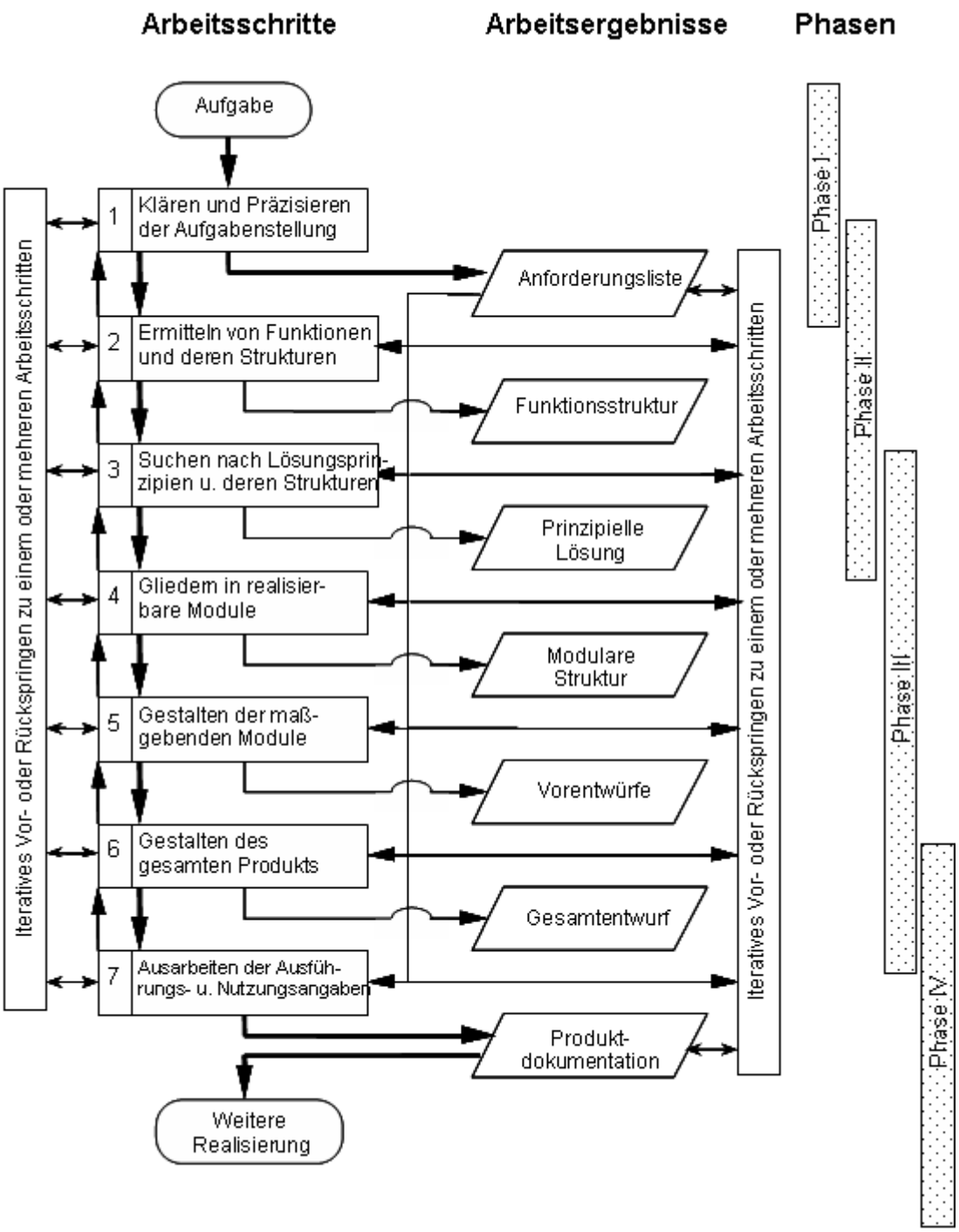


Abbildung 2: Phasen, Arbeitsergebnisse und Arbeitsschritte beim Konstruieren nach VDI 2221 [VDI-1993]

Eine der Grundlagen für die Richtlinie 2221 ist die VDI-Richtlinie 2210 aus dem Jahr 1975, die den Konstruktionsprozess in die vier Phasen *Anforderungsmodellierung*, *Funktionsmodellierung*, *Prinzipmodellierung* und *Gestaltmodellierung* (unterteilt in *Entwerfen* und *Ausarbeiten*).

- Die Phase der *Anforderungsmodellierung* dient zur vollständigen Klärung der Konstruktionsaufgabe. Hierunter wird das Zusammentragen aller Informationen

verstanden, die zum Entwurf und zur Herstellung eines gewünschten Produkts in der Zukunft bekannt sind. Ferner wird definiert, „was“ in der Zukunft geschehen soll, „wie“ es geschehen soll und welche Randbedingungen dabei jeweils zu beachten sind. In dieser Phase werden die notwendigen Produktanforderungen für die nachgelagerten Problemlösungsprozesse auf den weiteren Konkretisierungsstufen des Konstruktionsprozesses bereitgestellt .

- In der Phase der Funktionsmodellierung werden die Funktionen eines Produkts festgelegt und Überlegungen angestellt, wie die Funktionen des Produkts sinnvollerweise in Unterfunktionen unterteilt werden können. Dabei ergibt sich in der Regel eine hierarchische Struktur aus Funktionen, die über Funktionsflüsse miteinander in Beziehung stehen.
- In der Phase der Prinzipmodellierung wird festgelegt, durch welche physikalischen, chemischen, informationstechnischen oder vermehrt auch biologischen Prinzipien[4] die zuvor definierten Teilfunktionen umgesetzt werden sollen.
- In der Phase der Gestaltmodellierung wird die endgültige Produktbeschreibung erzeugt, die für die nachgelagerten Prozesse der Produktentstehung notwendig sind. Handelt es sich um ein mechanisches Produkt oder mechanische Komponenten, entsteht in dieser Phase die fertige Makro- und Mikrogeometrie (Toleranzen, Rauhigkeiten etc.). Bei mechatronischen Produkten werden in dieser Phase entsprechend Schaltpläne, Systemarchitekturen und Software fertiggestellt. Darüber hinaus werden in dieser Phase auch die Arbeiten zur Produktdokumentation abgeschlossen (z.B. Bedienungsanleitung, Montage-/Herstellungsanleitungen, Prüfanleitungen etc.).

2.1.2 Merkmale, Eigenschaften, Bedingungen und Anforderungen

Merkmale dienen im allgemeinen zur Unterscheidung von Objekten. Im Kontext dieser Arbeit dienen Merkmale insbesondere zur Unterscheidung von *produktbezogenen Modellentitäten* in den rechnerinternen Produktmodellen der Produktmodellierungswerkzeuge. Merkmale haben in jedem Fall eine eindeutige Bezeichnung und können, falls sie mit numerischen Werten ausgeprägt werden, eine vordefinierte Maßeinheit haben. Merkmale haben jedoch keine Ausprägung. Der Begriff „Merkmal“ dient im Zusammenhang dieser Arbeit als Oberbegriff für die Begriffe *Eigenschaft*, *Bedingung* und *Anforderung*.

Mit dem Begriff *Eigenschaften* werden hierbei die *Ist-Eigenschaften* eines Produkts bezeichnet. Darunter werden nicht nur die Eigenschaften eines tatsächlich vorhandenen Produkts verstanden, sondern insbesondere die Eigenschaften des virtuellen Produkts, d.h. eines Produkts, das noch nicht real existiert, sondern nur abstrakt in rechnerinternen Modellen abgebildet und beschrieben ist. Produkteigenschaften können auf vielfältige Weise ermittelt

---

4 Biologische Prinzipien werden beispielsweise in Sensoren zur Detektion von chemischen Substanzen verwendet, in denen lebende Zellen mit Elementen aus der Halbleitertechnik verbunden werden.

werden. Zum einen können sie direkt im Produktmodell vorliegen, wie beispielsweise die Dimensionen von Flächen und Kanten in Geometriemodellen, oder aber sie müssen auf andere Weise ermittelt werden, beispielsweise durch Simulationen, FEM[5]-Berechnungen oder physische Versuche.

Der Begriff *Bedingung* wird in dieser Arbeit im Sinne von *Bedingung zur Funktionsfähigkeit* benutzt und bezeichnet die Randbedingungen, die erfüllt sein müssen, damit eine Konstruktionslösung funktionsfähig ist, beispielsweise die maximale Betriebstemperatur einer Autobatterie. Im Unterschied zu Produktanforderungen dienen sie nicht direkt zur Überprüfung der Ist-Eigenschaften einer Produktkomponente; aus den Bedingungen eines Bauteils können aber indirekt Anforderungen an die Umgebung abgeleitet werden, die ihrerseits Komponenten des Produkts sein können.

Der Begriff *Anforderung* bezeichnet im Kontext dieser Arbeit die Soll-Eigenschaften eines Produkts, die zur Bewertung der Ist-Eigenschaften herangezogen werden. Diese Bewertung kann (und muss) schon sehr früh im Produktentwicklungsprozess geschehen, z.B. durch Berechnungen, Simulationen usw. Theoretisch steht jeder Produkteigenschaft mindestens eine Produktanforderung in einer 1:1-Beziehung gegenüber, anhand der sie überprüft werden kann. Der Zusammenhang zwischen Merkmalen, Anforderungen und Eigenschaften von Produktmerkmalen ist in Abbildung 3 dargestellt. In der Praxis ist es jedoch meist sinnvoller, n:m-Beziehungen zwischen Anforderungen und Eigenschaften zuzulassen.

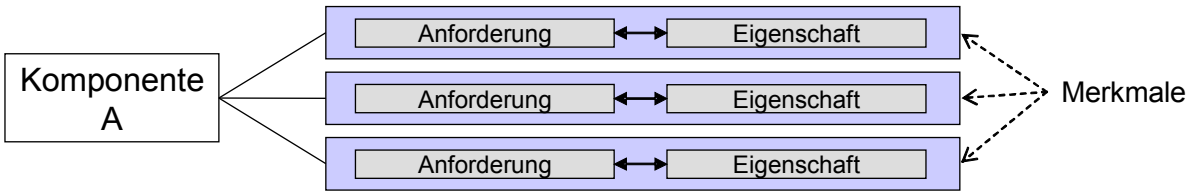


Abbildung 3: Merkmale als Oberbegriff für Anforderungen und Eigenschaften von Produktkomponenten

Anforderungen ergeben sich nicht nur aus dem Gebrauch des zu entwickelnden Produkts sondern auch aus allen anderen Phasen seines Lebenslaufs[6]. Des Weiteren resultieren Anforderungen aus verschiedenen Bereichen der Gesellschaft und Wirtschaft (siehe Abbildung 4). Anforderungen werden beispielsweise vom Gesetzgeber gestellt (z.B. Sicherheitsanforderungen oder Anforderungen aus dem Umweltschutz, Normen) oder gegebenenfalls von

5 FEM: Finite Elemente Methode. Vorgehensweise vor allem zur Simulation mechanischer Beanspruchungen von Bauteilen

6 In Wissenschaft und Wirtschaft ist der Begriff „Lebenszyklus“ oder „Produktlebenszyklus“ für die Abfolge der Lebensphasen eines Produkts (Entwicklung, Fertigung, Gebrauch etc.) verbreitet. Gleichwohl ist der Begriff des „Zyklus“ irreführend, da sich die Lebensphasen eines technischen Produkts i.d.R. nicht periodisch wiederholen.

entsprechenden Verbänden (z.B. VDI-Richtlinien). Die Gegebenheiten des eigenen Unternehmens geben ebenfalls Anforderungen vor, z.B. aus vorhandenen Fertigungsverfahren, Transportmitteln oder ähnlichem. Neben den gesetzlichen Anforderungen haben aber Anforderungen des Kunden, die entweder vom Kunden direkt formuliert oder indirekt (anonymer Kunde, z.B. bei Konsumgütern) durch das Marketing ermittelt werden, die höchste Priorität in der Produktentwicklung.

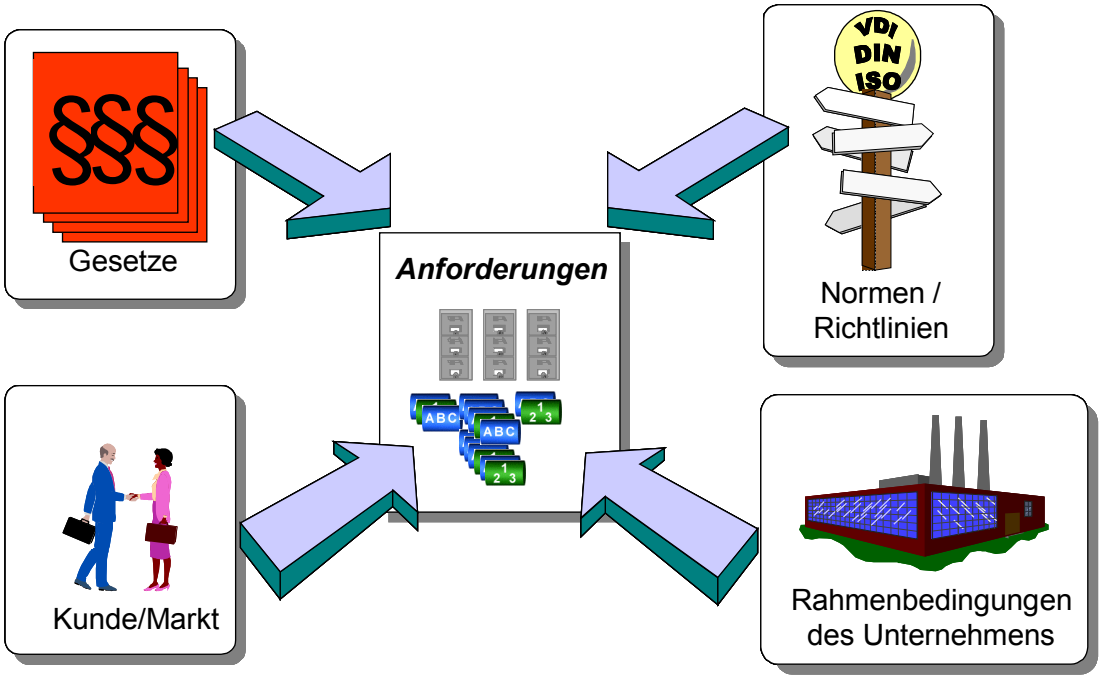


Abbildung 4: Herkunft von Anforderungen [Grab-2001]

Es werden verschiedene Arten von Anforderungen unterschieden. PAHL und BEITZ [PaBe-1997] sprechen von Forderungen und von Wünschen, wobei es sich empfiehlt, Wünsche entsprechend ihrer Bedeutung zu klassifizieren. ROTH [Roth-1994] unterscheidet Festforderungen, Zielforderungen und Wunschforderungen. Festforderungen sind als Punktforderungen und als ungezielte Grenz- und Bereichsforderungen formulierbar, während Zielforderungen in Mindest- und Höchstanforderungen sowie in gezielte Bereichsanforderungen und in nicht begrenzte Zielforderungen unterschieden werden können. Wunschforderungen sind zusätzliche Forderungen, die nicht fest vorgegeben sind, die aber wenn möglich ebenfalls erfüllt werden sollen. Die Festforderungen können formal beschrieben werden durch eine oder mehrere Gleichungen bzw. Ungleichungen; Zielforderungen durch keine, eine oder mehrere Ungleichungen zusammen mit einer Optimierungsrichtung.

Anforderungen werden vielfach noch nach anderen Kriterien unterschieden. Wichtig sind folgende Unterscheidungen:

- in qualitative und quantitative Anforderungen
- in externe und interne Anforderungen

- in explizite und implizite Anforderungen, sowie
- nach ihrer Herkunft aus den unterschiedlichen Produktlebensphasen

Unter qualitativen Anforderungen werden Merkmale eines Produktes verstanden, die nicht sinnvoll durch numerische Größen beschrieben werden können, z. B. ästhetisch oder tropenfest. Unter quantitativen Anforderungen werden Produktmerkmale verstanden, die sich dagegen durch numerische Größen beschreiben lassen. Für quantitative Anforderungen lassen sich Gleichungen oder Ungleichungen formulieren.

Bei externen Anforderungen handelt es sich um Anforderungen, die vom Kunden oder anderen Stellen außerhalb des Unternehmens, z.B. dem Gesetzgeber oder dem TÜV, vorgegeben werden, während interne Anforderungen vom Konstrukteur zusätzlich formuliert werden. Bei den internen Anforderungen werden unternehmensspezifische Randbedingungen oder beispielsweise Transportbedingungen auf dem Weg zum Einsatzort eines Produktes beschrieben.

Weiterhin kann zwischen expliziten und impliziten Anforderungen unterschieden werden. Implizite Anforderungen beruhen auf der Unausgesprochenheit von Anforderungen, die die am Problemlösungsprozess Beteiligten aufgrund ihres Hintergrundwissens berücksichtigen. Sie müssen erst durch Schlussfolgerungsprozesse abgeleitet werden. Explizite Anforderungen sind dagegen „erklärt“.

Entsprechend der Herkunft von Anforderungen aus unterschiedlichen Produktlebensphasen ist die Menge der Anforderungen, die ein Produkt erfüllen muss, erst am Ende des Produktlebens vollständig bekannt. Des Weiteren existieren Abhängigkeiten zwischen Anforderungen. Beispielsweise steht die Anforderung, ein Produkt so zu konstruieren, dass es möglichst kostengünstig hergestellt werden kann, offensichtlich im Widerspruch zu der Anforderung, eine optimale Qualität des Produkts zu erzielen. Abhängigkeiten zwischen Anforderungen sind allerdings oftmals schwer erkennbar oder werden erst bei zugeordneter Lösung offensichtlich. So ist im obigen Beispiel nicht auszuschließen, eine kostengünstigere Lösung zu finden, die trotzdem gleichzeitig eine höhere Qualität hat.

2.1.3 Anforderungsgetriebene, integrierte Produktentwicklung

Unter anforderungsgetriebener, integrierter Produktentwicklung soll in diesem Zusammenhang eine Vorgehensweise bei der Produktentwicklung verstanden werden, in der die in einem rechnerinternen Modell abgebildeten Produkteigenschaften computerunterstützt anhand von vorher definierten Produktanforderungen validiert werden. Diese Vorgehensweise verspricht ein unmittelbares Controlling des Produktentwicklungsprozesses ohne Medienbrüche. Es erlaubt so die Vermeidung von Entwicklungsfehlern, die durch die Nichtbeachtung von Produktanforderungen entstehen, sowie frühest mögliche Korrekturen am Produktmodell, wenn sich Anforderungen ändern. Zusätzlich sollen Abhängigkeiten zwischen Produktmerkmalen ermittelt, abgebildet und ausgewertet werden. Die Auswertung von Abhängigkeiten zwischen Produktmerkmalen ermöglicht es, die Auswirkungen von Änderungen von Eigen-

schaften von Produktmodellentitäten oder Änderungen von Anforderungen auf anderen Bereiche des Produktmodells zu erkennen. Im folgenden werden die Ansätze im Bereich der Anforderungsmodellierung erläutert, auf denen die in dieser Arbeit vorgestellten Forschungs- und Entwicklungsergebnisse aufbauen. Für eine detaillierte Darstellung des Standes der Technik und die dieser Arbeit zugrunde liegenden Ansätze im Bereich der Anforderungsmodellierung wird auf die im folgenden referenzierten Literaturstellen verwiesen.

2.1.3.1 Ansatz zur Anforderungsmodellierung des Instituts für Rechneranwendung in Planung und Konstruktion (RPK)

Am Institut für Rechneranwendung in Planung und Konstruktion (RPK) der Universität Karlsruhe wird seit Anfang der neunziger Jahre intensiv am Thema Anforderungsmodellierung geforscht[7].

Ansatz nach GEBAUER

Der Ansatz nach GEBAUER [Geba-2001] baut auf den Arbeiten von KLÄGER [Klae-1993] und RZEHORZ [Rzeh-1998] auf und stellt den letzten, in einer Dissertation dokumentierten Stand der Arbeiten des Instituts auf diesem Gebiet dar. Der Ansatz soll den Anwender im Produktentwicklungsprozess von der Anforderungsakquisition bis zur Produktentwicklung unterstützen. Zentraler Bestandteil des Konzepts sind *Anforderungsbibliotheken*. Dieser Ansatz wurde zum ersten Mal in der Arbeit von RZEHORZ vorgestellt. Anforderungsbibliotheken sind ihrerseits hierarchisch geordnet. Das bedeutet, dass Anforderungen, die beispielsweise allgemein für den Bereich Maschinenbau relevant sind, in einer Bibliothek zusammengefasst werden. Aus dieser Bibliothek können dann wiederum weitere Bibliotheken, etwa für die Bereiche Automobil-, Flugzeug- oder Schiffbau abgeleitet werden. Diese Bibliotheken können beliebig weiter konkretisiert werden in z.B. unternehmensspezifische bis hin zu produktgruppenspezifischen Anforderungsbibliotheken. Diese dienen als Basis für die Definition der Anforderungen an ein zu entwickelndes Produkt. Es können Anforderungen aus beliebig vielen Bibliotheken ausgewählt und zugeordnet werden.

In den *Anforderungsbibliotheken* sind vordefinierte, *Anforderungsmuster* hinterlegt, die abhängig vom Produktentwicklungskontext ausgewählt werden, um eine Produktentwicklungstätigkeit anzustoßen. Gebauer beschreibt *Anforderungsmuster* als Mengen von lösungsneutralen, d.h. noch keinem Produkt zugeordneten Anforderungen, die durch Beziehungen miteinander verknüpft sind. Anforderungen können hierbei durch *Abstraktionsbeziehungen*, *Bestandsbeziehungen*, *Unterstützende*, *Konkurrierende*, *Gegensätzliche* und *Erzeugende* Be-

7 Die Arbeiten zu diesem Thema wurden vor allem im Rahmen des Sonderforschungsbereichs (SFB) 346 „Rechnerintegrierte Konstruktion und Fertigung von Bauteilen“ durchgeführt [SFB-1999]. Des Weiteren wurde ein industrienaher Prototyp zur Anforderungsakquisition im Teilprojekt F6 „Entwicklung eines Anforderungsmodellierers“ im Transferbereich (TFB) 16 in Zusammenarbeit mit der Firma R.O.S.E Informatik GmbH in Heidenheim an der Brenz entwickelt [TFB-2003].

ziehungen verknüpft werden. *Abstraktionsbeziehungen* verknüpfen zwei Anforderungen, wenn eine Anforderung die andere abstrahiert. *Bestandsbeziehungen* beschreiben Zusammenhänge zwischen Anforderungen, wenn eine oder mehrere Anforderungen Bestandteil einer in dieser Beziehung übergeordneten Anforderung sind. Eine *Unterstützende Beziehung* existiert dann zwischen zwei Anforderungen, wenn die Optimierung einer Produkteigenschaft gemäß ihrer Anforderung auch die Optimierung einer anderen Produkteigenschaft gemäß ihrer Anforderung bewirkt. Stehen zwei Anforderungen in *Konkurrierender Beziehung* zueinander, kann eine Produkteigenschaft gemäß ihrer Anforderung nicht optimiert werden, ohne die Produkteigenschaft, die der anderen Anforderung in der Beziehung entspricht, gemäß ihrer Optimierungsrichtung zu verschlechtern. Stehen zwei Anforderungen in *Gegensätzlicher Beziehung*, können sie überhaupt nicht gleichzeitig erfüllt werden. *Erzeugende Beziehungen* sind gerichtet und drücken aus, dass die Zuordnung einer oder mehrerer Anforderungen zu einer Produktkomponente, eine oder mehrere andere, neue Anforderungen ebenfalls für diese Komponente relevant werden lassen.

Die Beziehungen der verschiedenen Abhängigkeitstypen verbinden die Anforderungen zu so genannten *Semantischen Anforderungsnetzen*, in denen die Abhängigkeiten die Kanten und die Anforderungen die Knoten darstellen. Die einzelnen *Semantischen Anforderungsnetze* der *Anforderungsmuster* können zu größeren *Semantischen Netzstrukturen* aggregiert werden. Da die *Semantischen Anforderungsnetze* in Bibliotheken abgelegt sind, repräsentieren sie das gesammelte Wissen über die Zusammenhänge und Abhängigkeiten in Anforderungsmengen lösungsneutral.

Gebauer führte darüber hinaus den Begriff der *Prozessmuster* ein. Prozessmuster werden wie Anforderungsmuster in Bibliotheken verwaltet und bearbeitet. Prozessmuster stellen Mengen von Entwicklungstätigkeiten dar, die erst durch eine gegebene, initiale Menge von Anforderungen angestoßen werden können, die als Anforderungsmuster vorliegt. Prozessmuster wirken sich auf die beteiligten Anforderungsmuster aus, d.h. sie überführen Anforderungsmuster von einem Zustand A_i in einen Zustand A_{i+1}.

Die lösungsneutrale Repräsentation von Wissen über Beziehungen zwischen Anforderungen hat als Nachteil, dass nur allgemeingültige Zusammenhänge abgebildet werden können, die weitgehend unabhängig von der konkreten Problemstellung gelten. Dies schränkt die Anwendbarkeit des Wissens über die Abhängigkeiten ein. Anforderungsbibliotheken könnten theoretisch zwar beliebig konkretisiert werden, sie verlieren dann aber ihren Vorteil der leichten Navigierbarkeit durch den Benutzer, da die Menge der zu konsultierenden Bibliotheken dann nicht mehr handhabbar ist. Die Problematik, wie Produkteigenschaften anhand der Anforderungen rechnerunterstützt evaluiert werden können, wurde nicht betrachtet.

Weitere Arbeiten des RPK

Die am RPK entwickelte Methodik zur Anforderungsmodellierung wurde in einem Software-Werkzeug umgesetzt[8]. Das entwickelte Programm **A**nforderungs**E**ntwicklungs**S**ystem (AES) ist in der Lage mit der Systemmodellierungssoftware RODON[9] über eine Dateischnittstelle zu kommunizieren. Mit AES ist es möglich, in RODON modellierte Ist-Eigenschaften eines Produkts anhand von Anforderungen zu evaluieren.

Bei der Modellierung von Anforderungen in AES werden diese den einzelnen Produktkomponenten (soweit bekannt) in einer logischen Produktstruktur zugeordnet. Die Produktstruktur spiegelt die Systemhierarchie, wie sie in RODON vorliegt, wider. Beim Export der Produktstruktur mit den zugeordneten Anforderungen wird in RODON eine der Produktstruktur entsprechende Systemstruktur aufgebaut und die Anforderungen in entsprechende Systemports umgesetzt. Dabei ist es zweckmäßig, zwischen *undefinierten* und *definierten* Anforderungen zu unterscheiden. Undefinierte Anforderungen entsprechen dem bloßen Wissen um die Relevanz einer Anforderung, ohne deren *Ausprägung* zu kennen, während definierte Anforderungen zusammen mit ihrer Ausprägung bekannt sind. Bei undefinierten Anforderungen werden lediglich Systemports in RODON erzeugt, die die gleiche Bezeichnung besitzen. Bei definierten Anforderungen werden hingegen zusätzlich die Maßeinheiten der entsprechenden Ports festgelegt sowie die durch die Anforderung erlaubten Wertebereiche. Während des Produktwicklungsprozesses wird durch die Erarbeitung und Auswahl von Lösungen auch die Menge der Anforderungen weiterentwickelt, d.h. undefinierte Anforderungen werden ausgeprägt, Ausprägungen von definierten Anforderungen werden geändert und zusätzliche, neue Anforderungen werden relevant. Dieser Vorgang wird mit *Anforderungsentwicklung* bezeichnet. Änderungen in dem in RODON modellierten System werden in AES zurückgeführt und entsprechend verarbeitet.

2.2 Ansätze zur Produktdatenintegration

Um die in einem Softwarewerkzeug erzeugten Informationen in einem anderen Werkzeug weiter verwenden zu können, müssen diese für das andere Werkzeug verständlich sein. Um dies zu erreichen, gibt es prinzipiell zwei Möglichkeiten: *Transformation* und *Integration* [Seil-1985].

Transformation: Die Informationen aus dem Informationsmodell des originären Systems werden durch geeignete Abbildungsvorschriften in das Informationsmodell des zweiten Systems abgebildet. Zwischen Entitäten der beiden Informationsmodelle bestehen keinerlei Beziehungen, d.h. sie sind *disjunkt*. Die Abbildungsvorschriften müssen für eine semantisch möglichst korrekte Interpretation der Informationen sorgen. Kennzeichnend für die Modelltransformation ist, dass nur eine Schnittmenge der Modellinhalte der beteiligten Modelle abgebildet werden kann. Diese Schnittmenge kann gleichwohl auch implizit gegebene Modellinhalte umfassen. Daraus ergeben sich zwei wesentliche Merkmale von

8 Im Rahmen des Teilprojekts F6 „Entwicklung eines Anforderungsmodellierers“ des TFB 16

9 Die Software wurde vom Systemhaus R.O.S.E Informatik entwickelt und vertrieben.

Modelltransformationen: Informationen sind redundant in beiden Modellen vorhanden und gleichzeitig gehen nicht redundante Informationen verloren, da sie nicht sinnvoll in das andere Modell abgebildet werden können.

Integration: die unterschiedlichen Teilaspekte eines Produkts werden in einem einheitlichen, integrierten Produktmodell zusammengefasst. Nach SEILER [Seil-1985] müssen integrierte Produktmodelle folgende Merkmale aufweisen:

- *Integrität*: Berücksichtigung semantisch korrekter, d.h. tätigkeits- und phasenübergreifender Zusammenhänge der Informationsobjekte im integrierten Produktmodell
- *Kohärenz*: Fortschreibung des Inhalts des Produktmodells ohne Modelltransformation
- *Akkumulation*: explizite Abbildung aller (relevanten) Informationen, die nicht aus anderen Informationen im Modell hergeleitet werden können
- *Assoziation*: Ableitbarkeit implizit enthaltener Informationen durch Ableitungsregeln. Das heißt, explizit vorhandene Informationen können Beziehungen zueinander haben.

Das Bestreben, Produktdaten zu integrieren, mündete in eine Reihe von Ansätzen, die zum Teil noch Gegenstand der Forschung, zum Teil in bereits kommerziell verfügbaren Lösungen implementiert sind. Dabei sind die oben geforderten Merkmale eines integrierten Produktmodells in den verschiedenen Ansätzen unterschiedlich stark ausgeprägt. Eine Bewertung diesbezüglich ist allerdings nicht im Fokus dieser Arbeit.

2.2.1 Kommerzielle Lösungen

Kommerzielle Lösungen zur Produktdatenintegration können unterschieden werden in Systeme, die sich auf die phasenübergreifende Verwaltung von Metainformationen[10] spezialisiert haben, ohne auf die eigentlichen produktbeschreibenden Daten zuzugreifen und in Systeme, die zusätzlich eine einheitlich Darstellung von Nutzdaten anstreben. Exemplarisch seien im folgenden PDM[11]-, bzw. PLM[12] Systeme als Vertreter für Systeme zur Metadatenverwaltung im allgemeinen beschrieben und das System von UGS als Vertreter für derartige Systeme zur Nutzdatenverwaltung. Außerdem wird das neutrale Dateiformat STEP beschrieben, das sich im Laufe der letzten Jahre als Standard für den Austausch von Produktdaten etabliert hat.

10 Metainformationen sind Informationen zur Verwaltung von anderen Informationen. Die verwalteten Informationen sind häufig produktbeschreibende Informationen, wie z.B. Geometriemodelle.

11 PDM: Produktdatenmanagement

12 PLM: Product Lifecycle Management

2.2.1.1 Produktdatenmanagementsysteme, Product-Lifecycle-Management-Systeme

In modernen, produktionsorientierten Unternehmen müssen eine Vielzahl von Daten verwaltet werden. Sie entstehen bei der Produktentwicklung, bei der Fertigung und bei der Verwaltung des Unternehmens als solches. Die bei der Produktentwicklung entstehenden Daten können in produktbeschreibende Daten (*Nutzdaten*) und in produktverwaltende Daten (*Metadaten*) unterteilt werden. Nutzdaten beschreiben Aussehen und Funktion des Produkts und werden beispielsweise mit Hilfe von CAD-Systemen[13] erstellt. Metadaten dienen dagegen zur Verwaltung der Nutzdaten. Sie umfassen Angaben über verschiedene Versionen der Nutzdaten, deren Freigabestatus, Benutzerrechte etc. und werden von so genannten Produktdatenmanagementsystemen (PDM-Systeme) zur Produktdatenverwaltung benutzt [EiSt-2004]. Typische Funktionen von PDM-Systemen umfassen [EiSt-2004]:

- **Strukturmanagement:** Verwaltung der Produktstruktur eines Produkts. Komplexe, technische Produkte setzen sich üblicherweise aus Komponenten zusammen, die ihrerseits entweder Einzelteile sein können oder Baugruppen, d.h. aus weiteren Komponenten bestehen. Das Strukturmanagement dient auch zur Erstellung von Stücklisten und Teile-, bzw. Baugruppenverwendungsnachweisen.
- **Konfigurationsmanagement:** Verwaltung von Produktkonfigurationen mit allen zugehörigen, zu einem bestimmten Zeitpunkt gültigen Daten und Dokumenten für alle Produktkomponenten in der betreffenden Produktkonfiguration.
- **Dokumentenmanagement:** Verwaltung der physischen Dateien im PDM-System mit Hilfe entsprechender Metadaten, wie z.B. Versionsnummern, Freigabestatus etc. oder weiterer Attribute, wie beispielsweise Dokumententyp (Angebotstext, Pflichtenheft etc.), die primär zur Klassifizierung und damit zur einfacheren Suche und Wiederverwendung von Dokumenten dienen.
- **Datensicherungs- und Sicherheitsmanagement**: Das Sicherungsmanagement ist verantwortlich für die Vermeidung von Datenverlusten, verursacht z.B. durch technisches Versagen, Unglücksfälle etc., während das Sicherheitsmanagement für die Vermeidung des Datenzugriffs durch Unbefugte verantwortlich ist.
- **Änderungsmanagement:** Das Änderungsmanagement regelt die Durchführung von Konstruktionsänderungen auf Dokumentenebene sowie von bestimmten Daten im PDM-System selbst, wie z.B. Änderungen an der Produktstruktur. Dazu gehört die Verwaltung des Freigabestatus eines Konstruktionsdokuments oder einer Produktkonfiguration, die Versionskontrolle, die Zugriffskontrolle, das Einchecken und Auschecken von Konstruktionsdokumenten etc.
- **Klassifikationsmanagement:** Das Klassifikationsmanagement erlaubt die vereinfachte Suche und Wiederverwendung von im PDM-System verwalteten Dokumenten und Daten z.B. auf der Basis von Sachmerkmalleisten oder anderen speziellen Dokumentattributen.

---

13 CAD, Computer Aided Design: rechnerunterstützte Vorgehensweise zum Produktentwurf

- **Versionsmanagement:** Kontrolle der Entstehungs- und Änderungsgeschichte von ausgewählten Daten und den vom PDM-System verwalteten Dokumenten.
- **Freigabemanagement:** Kontrolle der Freigabeabläufe von Dokumenten und relevanten Daten im PDM-System. Es wird festgelegt, welche Freigabestati erlaubt sind und ob und wie in den einzelnen Freigabestati die Daten modifiziert werden können.
- **Workflowmanagement:** die Erstellung, die Verwaltung und der Ablauf von Geschäftsprozessen werden im Workflowmanagement eines PDM-Systems kontrolliert. Geschäftsprozesse umfassen dabei festgelegte Abläufe beispielsweise zur Änderung von Produktmodellen.

Die einzelnen Funktionen sind auf vielfältige Weise miteinander verzahnt. So greift beispielsweise das Konfigurationsmanagement auf Funktionen des Strukturmanagements, des Dokumentenmanagements, des Freigabemanagements, des Versionsmanagements sowie des Sicherungs- und Sicherheitsmanagements zurück.

PDM-Systeme wurden seit ihrer Einführung in den 1980er Jahren ständig weiterentwickelt und um neue Funktionen und Aufgaben erweitert, die für die der Produktentwicklung und Arbeitsvorbereitung nachgelagerten Produktlebensphasen relevant sind. Daher hat sich für diese Systeme die Bezeichnung Product Lifecycle Management (PLM) etabliert. Neue Funktionen umfassen beispielsweise:

- **Supply Chain Management (SCM) :** Unterstützung von unternehmensübergreifenden Geschäftsprozessen zwischen Herstellern und Zulieferern. Dies umfasst sowohl die Kopplung der internen Geschäftsprozesse an die Geschäftsprozesse des/der Partner als auch die gegenseitige Bereitstellung der relevanten Produktdaten.
- **Enterprise Ressource Planning (ERP):** Planung der unternehmensinternen Ressourcen. Dies umfasst sowohl die Personalressourcen als auch die Fertigungsmittel, die Materialplanung, Einkauf etc. ERP-Systeme haben sich häufig aus Produktionsplanungs- und Steuerungssystemen (PPS) entwickelt.
- **Maintenance-, Repair- and Overhaul-Management (MRO):** Ergänzt die Produktdatenverwaltung um Funktionalität zur Ersatzteilplanung, die Festlegung und Verwaltung von Wartungsintervallen etc.
- **Requirement Traceability Management (RTM):** Dokumentiert und verwaltet Produktanforderungen.
- **Customer Relationship Management (CRM):** Unterstützt die Pflege von Kundenbeziehungen. Dazu gehört beispielsweise die Verwaltung von Kundendaten, die Rückführung und Dokumentation von Kundenreaktionen auf das Produkt (Customer Feedback) etc.

Zusammenfassend lässt sich feststellen, dass PLM-Systeme häufig eine Integration von Funktionalität der ehemals getrennten Systemklassen PDM und PPS darstellen. Die meisten Hersteller dieser ehemals getrennten Systemklassen bieten inzwischen selbst PLM Systeme an, wobei sie häufig den Schwerpunkt auf die angestammten Funktionsbereiche ihrer ursprünglichen Produkte legen. Ein weiterer Trend ist die Öffnung der Systeme. Das bedeutet,

es wird verstärkt dem Umstand Rechnung getragen, dass Kompetenzen und Ressourcen von den Firmen ausgelagert und individuell und projektbezogen von externen Partnern eingekauft werden. Dazu wird ein effektiver Informationsaustausch über die Grenzen der beteiligten Partner hinweg benötigt.

Das Manko von PLM-Systemen ist, dass sie sich in ihrer Funktionalität auf die Verwaltung von Metadaten beschränken. Das bedeutet, dass produktbeschreibende Daten nicht miteinander verknüpft werden können, sondern allein dem Zugriff durch das jeweilige Autorensystem unterliegen. Damit ist auch keine Abbildung und Auswertung der Abhängigkeiten zwischen den produktbeschreibenden Merkmalen möglich.

2.2.1.2 UGS Teamcenter

UGS Teamcenter von UGS [UGS-2005] zielt auf die Integration aller in der Produktentstehung erzeugten Informationen ab. Damit erhebt es den Anspruch, über die im vorigen Abschnitt beschriebene PDM/PLM-Funktionalität hinaus, auch produktbeschreibende Informationen miteinander zu verknüpfen. Durch den Zukauf entsprechender Softwareunternehmen konnte UGS ein Produktportfolio aufbauen, das den gesamten Engineeringbereich abdecken und in einer integrierten Lösung angeboten werden soll. Dazu sollen verschiedene, ehemals isolierte Engineering-Werkzeuge miteinander verknüpft werden [Samp-2003]. Die Palette umfasst System Engineering Systeme, PDM-Systeme, CAD/CAE-Systeme, ERP-Systeme, Projektmanagementsysteme sowie Systeme zur Unterstützung von teamübergreifender Kollaboration.

Diese Werkzeuge sollen auf ein gemeinsames *Lifecycle Repository* von Produktinformationen zugreifen. Die Funktionalität soll soweit ausgebaut werden, dass Abhängigkeiten zwischen Elementen, die in den verschiedenen Werkzeugen modelliert sind, abgebildet und ausgewertet werden können. Damit lassen sich auch Anforderungen mit entsprechenden Produkteigenschaften verknüpfen und die Auswirkungen von Änderungen von Anforderungen auf die Produkteigenschaften ermitteln. Umgekehrt kann bei Änderungen an Produkteigenschaften ermittelt werden, ob mit den geänderten Produkteigenschaften die Produktanforderungen noch erfüllt werden.

Der oben genannte Anspruch zur Integration von Nutzdaten wird jedoch von der derzeitigen Version des Produkts nicht erfüllt. Die Produktpalette von UGS hat sich grundlegend geändert. Derzeit im Angebot von UGS ist *Teamcenter Requirements*, das die Funktionalität zur Modellierung von Anforderungen bereitstellt [UGS-2005b]. Es existieren Verbindungen zu *Teamcenter Engineering* (PDM-Funktionalität), *Teamcenter Project* (Projektmanagementfunktionalität) und *Teamcenter Enterprise* (Workflowmanagement, PDM-Funktionalität). Damit sind lediglich Systeme zur Verwaltung von Metadaten angebunden. Eine Verknüpfung von Anforderungen mit modellierten Produkteigenschaften existiert nicht. Ein weiteres Produkt von UGS, das Anforderungsmodellierung unterstützt, ist *Teamcenter SLATE* [UGS-2005a]. Mit dem Werkzeug können Systeme hierarchisch aus sogenannten *Building Blocks* zusammengestellt werden, die die Produktmodule repräsentieren. Diese Building

Blocks können in verschiedenen Sichten zusammengestellt werden. Das Werkzeug bietet zusätzlich Unterstützung zur Erzeugung von Produktdokumentation zur Verwaltung von Produktgruppen (*Product Portfolio Management*) und besitzt Schnittstellen zu Teamcenter Enterprise. Dem oben geschilderten Anspruch zur Abbildung von Abhängigkeiten zwischen Produktmerkmalen aus unterschiedlichen Modellierungssystemen nach [Samp-2003] wird dieses Werkzeug nicht gerecht.

2.2.1.3 Das Produktmodell von STEP

STEP ist in der ISO[14]-Norm ISO 10303 standardisiert. STEP wurde mit der Zielsetzung entwickelt, Produktinformationen unabhängig von der verwendeten Hard- und Software der beteiligten Systeme auszutauschen. Es handelt sich um ein neutrales Format zum Austausch von Produktdaten. In STEP können theoretisch Produktdaten aus allen Phasen der Produktentstehung abgebildet werden. Es weist alle in Abschnitt 2.2 genannten Merkmale eines integrierten Produktmodells auf [GrAP-1994a][GrAP-1994b][GrAP-1994c].

Um Informationen aus externen Modellen in STEP abzubilden, werden anwendungsspezifische Programme (sogenannte STEP-Prozessoren) verwendet, die die Modellinhalte durch entsprechende Abbildungsvorschriften in Modellinhalte von STEP und umgekehrt, STEP-Modelle in die Modelle der jeweiligen Anwendung transformieren. Für diese Transformationen gelten ebenfalls die in 2.2 beschriebenen Implikationen.

Durch die Mächtigkeit und die Komplexität des STEP-Modells gibt es in den meisten Fällen eine Vielzahl von plausiblen Abbildungsmöglichkeiten. Für die Interoperabilität von verschiedenen Anwendungen auf STEP-Basis muss daher eine einheitliche Interpretation des STEP Modells sichergestellt werden. Dazu wurden sogenannte Application Protocols (AP) entwickelt, die für jeden Anwendungsbereich eine einheitliche Interpretation des STEP-Modells sicherstellen sollen. Beispielsweise ist AP 214 für den Bereich Automobilbau konzipiert worden. Aber auch dieser Ansatz bietet noch genügend Spielraum für Interpretationen und kann daher keine zufrieden stellende Standardisierung gewährleisten. Die Mächtigkeit von STEP bedingt zudem, dass eine funktionierende Integration von Systemen sehr aufwendig ist und nur von Experten auf diesem Gebiet vorgenommen werden kann.

2.2.2 Ansätze aus der Forschung

2.2.2.1 Das integrierte Produkt- und Produktionsmodell (PPM)

Im Rahmen des Sonderforschungsbereichs (SFB) 346 wurde ein Konzept zur integrierten Darstellung aller Produktinformationen unter der Bezeichnung PPM (integriertes Produkt- und Produktionsmodell) entwickelt und prototypisch implementiert [SFB-1999]. In ihm können Informationen aus allen Bereichen der Produktentstehung abgebildet werden. Die Pa-

14 International Organisation for Standardisation

lette der Teilmodelle umfasst die Angebotsbearbeitung, die Phasen der Konstruktion bis hin zu Fertigung und Montage. Darüber hinaus werden die Organisationsstrukturplanung, die Personaleinsatzplanung sowie die Betriebsmittel und Materialflussplanung unterstützt. Das besondere am PPM ist, dass Produkt und Produktionsmittel in einheitlicher, integrierter und anwendungsunabhängiger Form repräsentiert werden [SFB-1999].

Während der Arbeiten im SFB 346 wurde erkannt, dass ein umfassendes, akkumulatives Produktmodell sehr komplex und unhandlich ist. Daher wurden geeignete Methoden entwickelt, um die Komplexität zu beherrschen und einen transparenten Zugriff durch den Rechner, aber auch durch den Menschen, zu ermöglichen. Dazu wurden unter anderem Methoden zur Sichtenbildung entwickelt. Das bedeutet, dass alle wesentlichen Objektstrukturen im sogenannten Kernmodell verfügbar sind, und aus den Elementen dieses Kernmodells anwendungsspezifische Teilmodelle (Sichten) erstellt werden. Durch geeignete Mechanismen wird sichergestellt, dass die in den jeweiligen Sichten verfügbaren Informationen teilmodellübergreifend kohärent sind und miteinander in semantisch korrekten Beziehungen stehen.

2.2.2.2 Der Metamodell-Ansatz aus DRAGON

Im Rahmen des Projekts *DRAGON* [15] wurden Methoden für die Unterstützung verteilter, interkultureller Produktentwicklungspartnerschaften entwickelt und in einem Engineering Portal implementiert [DRAG-2004]. Als informationstechnisches Fundament des Portals dient ein generisches, metamodellbasiertes Datenmodellierungsframework, das über Webservice-Schnittstellen angesprochen wird [Ehrl-2004]. Unter Metamodellen sind Datenmodelle zu verstehen, die der Beschreibung von Datenmodellen dienen. Das Metamodell muss dazu alle nötigen Konstrukte bereitstellen, die zur allgemeingültigen und vollständigen Beschreibung der betrachteten Datenmodelle notwendig sind. Der Ansatz dient vor allem dazu, die Informationsmodelle externer Systeme flexibel abzubilden und miteinander zu verknüpfen.

Der wichtigste Datentyp des Metamodells ist das *Objekt* (*Object*). Die Eigenschaften eines Objekts werden durch *Attributwerte* (*Attribute Value*) beschrieben. Objekte sind einem *Objekttyp* (*Object Type*) zugeordnet, der über eine *Attributdefinition* (*Attribute Definition*) festlegt, welche *Attributtypen* (*Attribute Type*) ein ihm zugeordnetes Objekt haben kann. Abbildung 5 zeigt die Zusammenhänge zwischen den Konstrukten zur Repräsentation der instanziierten Daten (linker Block in der Graphik) und den Konstrukten zur Repräsentation des zugehörigen Objektmodells, d.h. der Strukturierung und Typisierung der Objekte (rechter Block in der Graphik).

---

15 ***D****evelopment of an inte****RA****ctive En****G****ineering Portal for* ***O****pen* ***N****etworks*, IST-2000-29366.

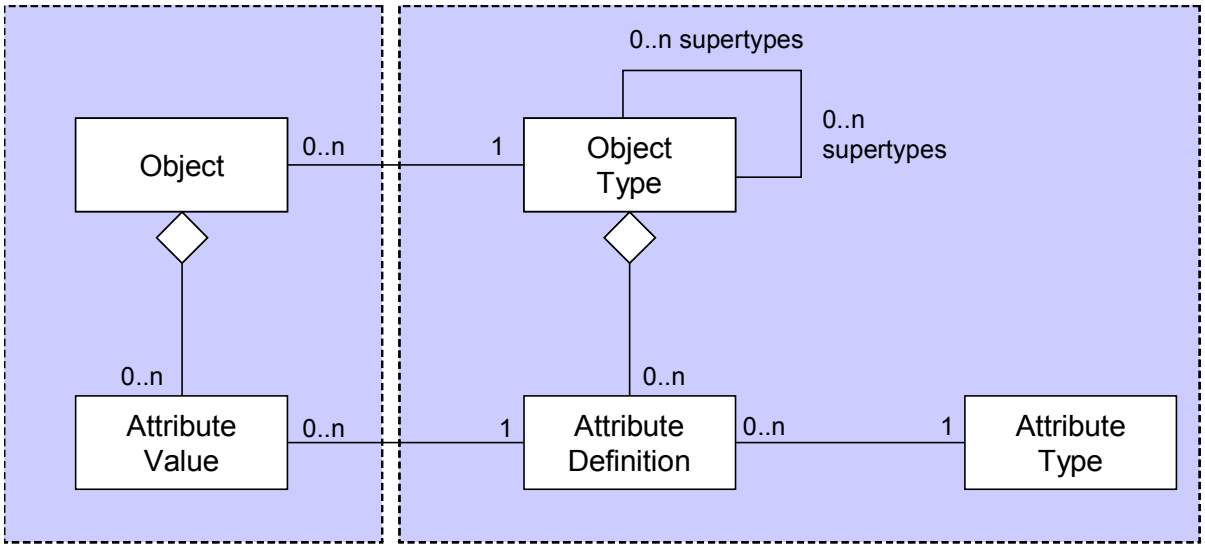


Abbildung 5: Objekte und Objekttypen nach EHRLER [Ehrl-2004]

Zwei Objekte können durch eine *Relation* (*Relationship*) - genauer gesagt durch ein Relationsobjekt - miteinander verknüpft werden. Relationen werden ähnlich wie Objekte repräsentiert d.h., es gibt eine Ebene zur Repräsentation der instanziierten Relationen und eine zugehörige Ebene zur Definition der gültigen Relationen. Jede Relation ist genau einem Relationstyp (*Relationship Type*) zugeordnet, der festlegt, welche Objekttypen diese Relation verknüpfen kann und welche Attributtypen der Relation zugeordnet werden können. Abbildung 6 veranschaulicht die Zusammenhänge zwischen der Repräsentation der instanziierten Relationen (linker Block in der Graphik) und der Repräsentation des Relationenmodells (rechter Block in der Graphik).

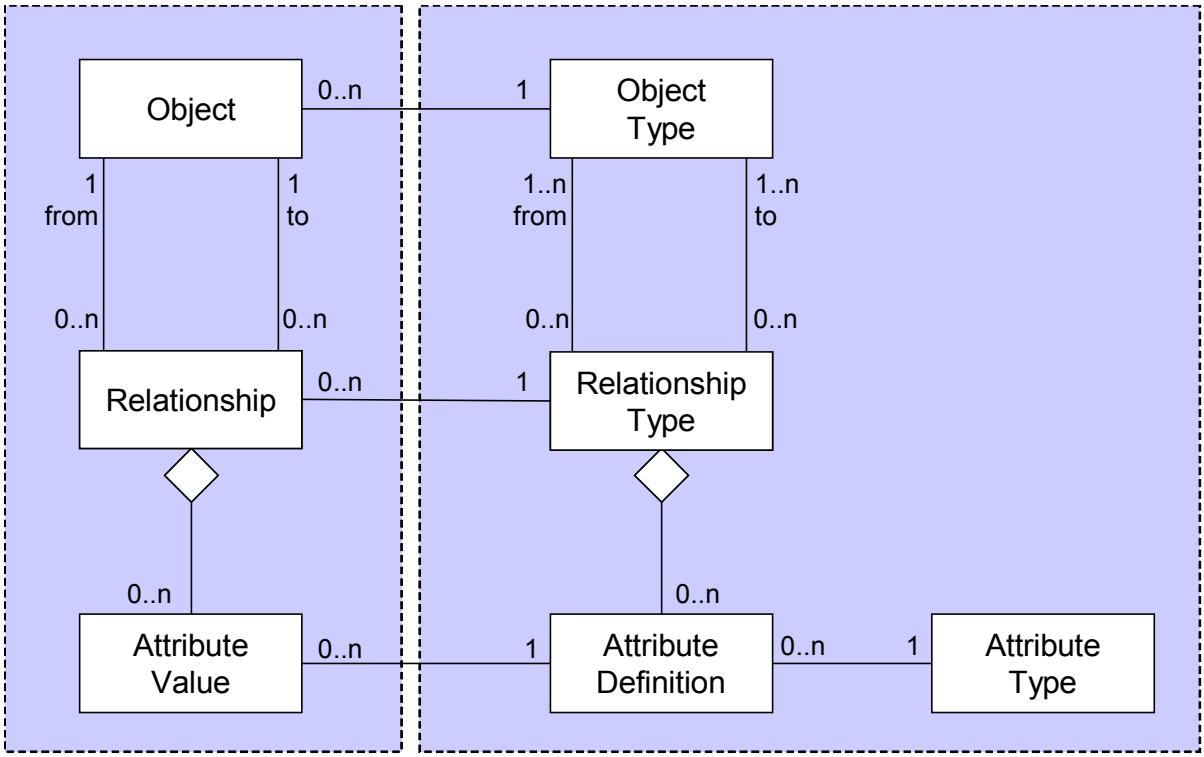


Abbildung 6: Relationen und Relationstypen nach Ehrler [Ehrl-2004]

Im Rahmen von DRAGON wurde auch ein Konzept erarbeitet und umgesetzt, das die Frage der Anbindung externer Datenquellen, wie z.B. PDM- oder CAD-Systeme an das Datenmodellframework beantwortet [MaEH-2004]. Dazu müssen für jedes anzubindende System spezielle Adapter implementiert werden, die die Konstrukte der Datenmodelle der externen Systeme in das metamodellbasierte Datenmodell der *unternehmensübergreifenden Integrationsplattform* (*UIP*) übersetzen (Mapping) und deren Inhalte dort verfügbar machen. Abbildung 7 zeigt schematisch die Anbindung externer Datenquellen an die Integrationsplattform.

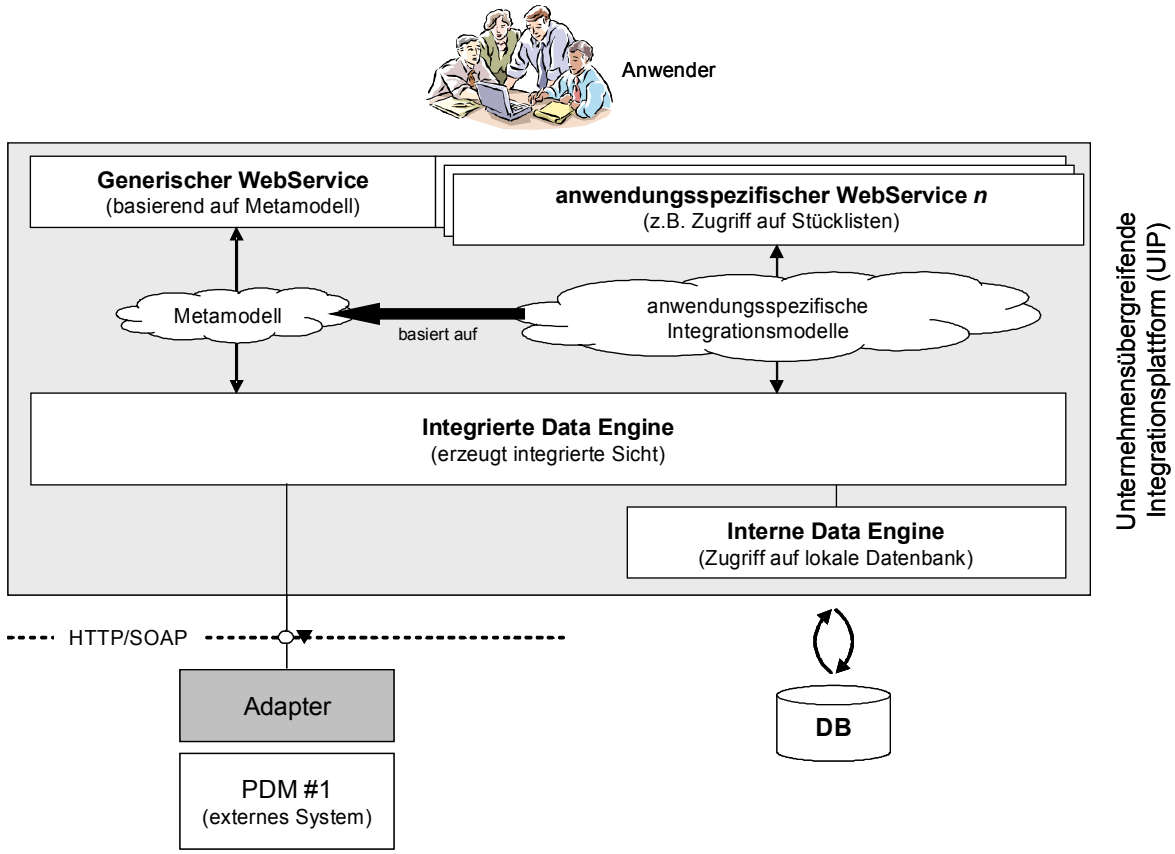


Abbildung 7: Anbindung externer Datenquellen an die Integrationsplattform UIP [MaEH-2004]

Die Integrationsplattform ist in mehreren Schichten aufgebaut. Zu unterst sind die eigentlichen Datenquellen angesiedelt. Dies können zum einen angebundene, externe Datenquellen sein, wie beispielsweise PDM-Systeme, die über einen Adapter, der für jedes System implementiert werden muss, angesprochen werden. Er bereitet die Daten aus seinem System auf und stellt sie über vorgegebene Protokolle der Integrationsplattform zur Verfügung. Zum anderen ist auf der gleichen Ebene die interne Datenbank angesiedelt, die die von der Integrationsplattform erzeugten Informationen verwaltet. Darüber liegt die *Integrierte Data Engine*. Sie ist dafür zuständig, die Daten der verschiedenen, darunter liegenden Systeme zu verknüpfen. Anwendungsspezifische Webservices greifen auf die aufbereiteten Informationen der Data Engine zu, interpretieren sie und bereiten sie, jeweils für einen bestimmten Anwendungszweck, für den Benutzer auf. Parallel dazu ist ein generischer Webservice angesiedelt, mit dessen Hilfe auf die Entitäten des Metamodells zugegriffen werden kann. Er interpretiert die Daten nicht, sondern zeigt lediglich Modellentitäten, sowie deren Verknüpfungen mit anderen Entitäten an.

Im DRAGON-Projekt wurden Konzepte für die Abbildung und Verknüpfung von Entitäten externer Informationsmodelle entwickelt. Konzepte für eine selbständige, rechnerunterstützte Erkennung von Abhängigkeiten zwischen Modellentitäten wurden nicht entwickelt.

2.2.2.3 Der Ansatz aus FIPER

In dem Projekt *Federated Intelligent Product Environemnt*[16] (*FIPER*) wurden neue Ansätze zur Unterstützung unternehmensübergreifender Entwicklungsprozesse entwickelt und implementiert [AIAA-2000]. Zentrales Konzept ist die Bereitstellung einer Plattform zur Verwaltung und Koordination föderativer Dienste. Das bedeutet, dass die eigentlichen, wertschöpfenden Tätigkeiten von spezialisierten Softwarekomponenten geleistet werden, die von der FIPER-Plattfom angesteuert und koordiniert werden. Dazu müssen diese Dienste gewisse Voraussetzungen erfüllen, um in das Framework eingebunden werden zu können. Sollen bestehende, fremde Systeme (sog. „*Legacy Systems*“), wie z.B. CAD-Systeme oder FEM-Software, eingebunden werden, so muss für diese ein sogenannter *Wrapper*[17] geschrieben werden, der die Systemfunktionalität nach außen für die FIPER-Plattform zugänglich macht. Der Ansatz von FIPER sieht kein integriertes Produktmodell vor, sondern beschränkt sich darauf, Informationen, die in einem System erzeugt wurden, anderen Systemen in einem dem konsumierenden System verständlichen Dateiformat zur Verfügung zu stellen. Hierbei handelt es sich also um eine Modelltransformation nach [Seil-1985]. Die FIPER-Plattform ist dabei sehr flexibel skalierbar: Dienste, die der Plattform zur Verfügung gestellt werden sollen, melden sich einfach in dem Framework an, woraufhin die nötigen Zugangsdaten in der *Library* gespeichert werden. Danach können entsprechende Workflows definiert und angestoßen werden.

Die FIPER-Plattform dient zur Koordination und Automatisierung von Teilaspekten der Produktentwicklung. Dazu wird die Verarbeitung der Informationen durch das konsumierende System von FIPER automatisch angestoßen und die Ergebnisse wieder eingesammelt Die FIPER-Plattform koordiniert diese Abläufe durch eine *Workflow-Engine*, in der die Prozesse abgebildet und kontrolliert werden. Der *Context Manager* stellt die benötigten Informationen für das jeweils nachfolgende, konsumierende System in einer Form, die vom nachfolgenden verarbeitet werden kann, zusammen. Der *Job Dispatcher* entscheidet, welcher Dienst die aufbereiteten Daten verarbeiten soll und schickt die Informationen an die entsprechende Adresse. Der *Results Manager* sammelt die Ergebnisse nach Verarbeitung durch einen Dienst wieder ein und stellt sie der Plattform für den folgenden Arbeitsschritt zur Verfügung.

Eine wirkliche Datenintegration findet im Ansatz von FIPER nicht statt. Es ist vielmehr eine Integration von Diensten, um Entwicklungsprozesse automatisiert zu koordinieren. Die einzelnen Dienste greifen dabei immer noch auf isolierte Datenmodelle zurück, die lediglich durch Modelltransformation miteinander gekoppelt sind.

---

16 Projektlaufzeit von 1999 bis 2003 [Engi-2005b] wurde vom US-amerikanischen *National Institute for Standards* (*NIST*) gefördert [NIST-2005]. Am Projekt waren Engineous Software [Engi-2005a], General Electric [GE-2005], BFGoodrich [BFGo-2005], Ohio Aerospace Institute (OAI) [OAI-2005], Parker Hannifin [Park-2005] und die Universität von Ohio [OU-2005] beteiligt.

17 Wrapper sind Programme, die eine bestehende Software kapseln und den Zugriff von außen nach innen und von innen nach außen steuern und ggf. die ausgetauschten Informationen für den jeweiligen Adressaten verständlich aufbereiten.

2.2.2.4 Pan Galactic Engineering Framework (PGEF)

Ein weiteres, im Rahmen dieser Arbeit relevantes Projekt, das sich mit der Integration unterschiedlicher Produktentwicklungswerkzeuge beschäftigt, ist das *Pan Galactic Engineering Framework*[18] (*PGEF*) [Wate-2004].

Das PGEF ist ein Client-Server-System. Hauptaufgabe der Serverkomponente ist die Integration von Modellinformationen aus allen Ingenieursdisziplinen (Elektrik, Mechanik, Analyse, System Engineering etc.). Dazu wird ein standardbasierter Verzeichnisdienst für den Import und das Management von Datenverzeichnissen[19] bereitgestellt. Außerdem bietet der Server eine einheitliche Sicht auf alle Modellierungsparameter (die im Datenverzeichnis aufgeführt sind) über den gesamten Produktlebenslauf. Eine weitere Aufgabe ist die Bereitstellung von Standardmodellbibliotheken und –Archiven, um die Wiederverwendung von Entwürfen zu ermöglichen [Wate-2003].

Der Client stellt Funktionalität für die Suche nach und Präsentation von Modellinformationen aus allen Ingenieursdisziplinen bereit. Über den Client können die Informationen in den Datenverzeichnissen auch erstellt und verändert werden. Zusätzlich kann der Client die Modellparameter überprüfen sowie ggf. Änderungsbenachrichtigungen auslösen. Die Erstellung von verschiedenen Arten von Konzept-, Funktions-, Verhaltens- und physikalischen Produktstrukturen für Simulationen, Digital Mock-Up etc. ist eine weitere Aufgabe des Clients [Wate-2003].

Die Informationen werden serverseitig in einem Datenmodell gespeichert, das auf STEP basiert. Dazu werden sie in einem *Integrated STEP Master Model* auf dem Server gespeichert und mittels Mapping in geeignete Anwendungsprotokolle (STEP-AP) abgebildet, die dann in die einzelnen, angebundenen Produktentwicklungswerkzeuge importiert, bzw. von ihnen in den PGEF-Server exportiert werden. Die Kommunikation mit dem Client findet direkt über das gemeinsam zugrunde liegende Modell statt. Abbildung 8 zeigt das Framework im Kontext der angebundenen Werkzeuge und des Clients.

---

18 PGEF wurde ursprünglich von der National Aeronautics and Space Administration (NASA) [NASA-2005a] als Testapplikation für STEP-Anwendungen entwickelt (*NASA STEP Testbed)* [NASA-2005b] und wird als Open Source Projekt weiterentwickelt [Wate-2005b].

19 Der Begriff *Datenverzeichnis* ist eine Übersetzung des englischen Begriffs *Data Dictionary* und bezeichnet eine Sammlung von Metadaten, die zum Datenmanagement benötigt werden, z.B. Angaben über Bezeichnungen, Anwender, Rollen, Zugriffsrechte, Integritätsbedingungen etc.

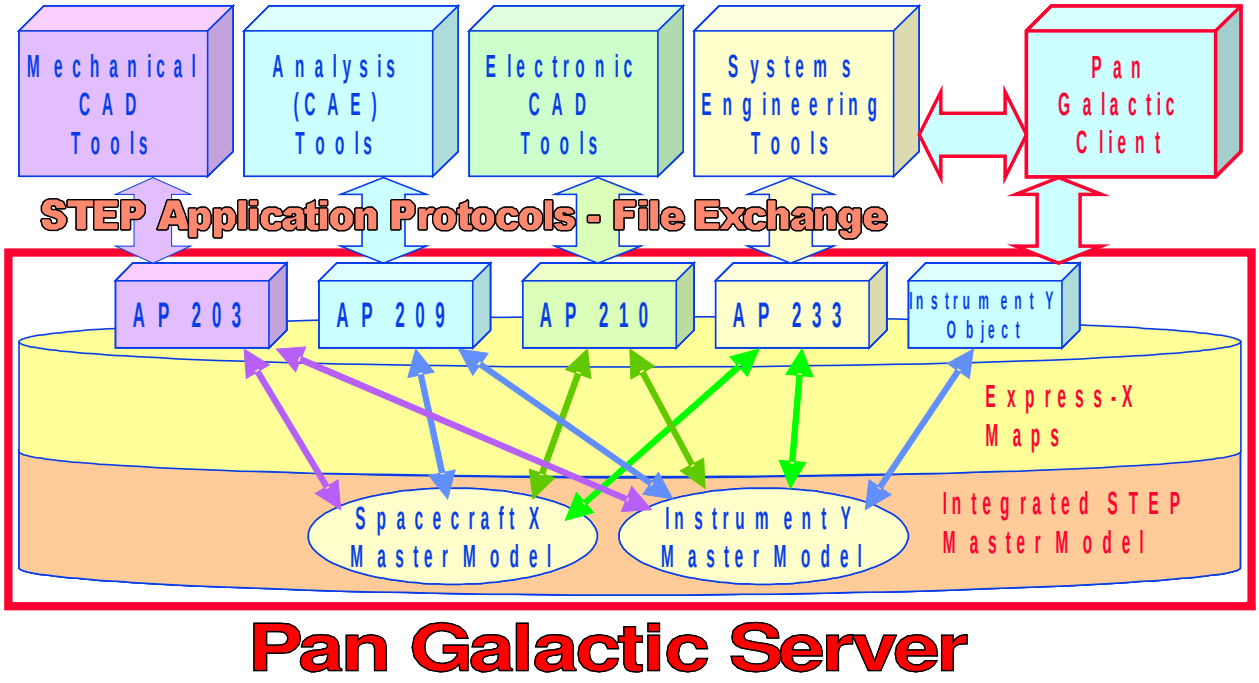


Abbildung 8: Architektur des PGEF [Wate-2003]

Da das System auf STEP basiert, gelten prinzipbedingt auch dessen Einschränkungen (siehe Abschnitt 2.2.1, Seite 18). Diese treten in ihrer Bedeutung jedoch nicht so zutage, da es sich bei dem PGEF um ein speziell für einen Anwendungszweck und auf dessen Rahmenbedingungen zugeschnittenes System handelt. Eine Rechnerunterstützung für die Erkennung von Abhängigkeiten zwischen Produktmerkmalen ist ebenfalls nicht gegeben.

2.3 Systeme zur Abbildung von Abhängigkeiten zwischen Produkteigenschaften

Es gibt eine Vielzahl von Beispielen für Produktmodellierungswerkzeuge, die Abhängigkeiten zwischen Produkteigenschaften abbilden. Im folgenden werden dafür beispielhaft System Engineering Werkzeuge und parametrische CAD-Systeme vorgestellt.

2.3.1 Computer Aided Design (CAD) Werkzeuge

3D-CAD-Systeme (im folgenden als „CAD-System“ bezeichnet) stellen die zentralen und herausragenden Werkzeuge in der Produktentwicklung der Maschinenbaubranche dar [FFGR-2003]. Die mit ihrer Hilfe erstellten Modelle enthalten die Beschreibung eines zu fertigenden Produkts, d.h. in den CAD-Dateien sind neben den reinen geometrischen Informationen auch weitergehende Beschreibungen der Produkteigenschaften vorhanden, wie z.B. Materialeigenschaften. Aus den Informationen im CAD-System werden weitere, für die Produktfertigung relevante Informationen abgeleitet, wie z.B. NC-Code für die Steuerung von Werkzeugma-

schinen in CAM-Systemen. Dies führte dazu, dass CAD-Dateien inzwischen papierbasierte Unterlagen als Grundlage für Fertigungs- und Entwicklungsaufträge, sowohl unternehmensintern als auch bei der Beauftragung externer Firmen, weitgehend abgelöst haben.

Ein wesentlicher Bestandteil eines CAD-Systems ist das Geometriemodell. Von ihm hängt die realisierbare Funktionalität eines CAD-Systems in entscheidendem Maße ab. In dieser Arbeit werden wegen ihrer Dominanz in der Maschinenbaubranche ausschließlich 3D-Volumenmodelle betrachtet. Es lassen sich mehrere Ansätze für die Abbildung dreidimensionaler Objekte unterscheiden [Grae-1989].

- **Akkumulative Modelle:** Bei den akkumulativen Modellen ist die Menge der Modellinformationen getrennt vom Erzeugungsprogramm als Datenstruktur abgelegt. Die Konsistenz muss durch spezielle Programme überprüft werden. Bei diesen Modellen kann direkt auf jedes Datenelement der Datenstruktur, wie z.B. Flächen oder Kanten zugegriffen werden. Die wichtigsten Vertreter der Akkumulativen Modelle sind Geometrisch-Topologische Strukturmodelle, auch *B-Rep* oder *Boundary Representation* Modelle genannt.
- **Generative Modelle:** Bei den generativen Modellen ist die Information in Form einer Erzeugungsvorschrift des Modells, d.h. eines Programms im Speicher abgelegt. Alle Modellinformationen sind hierbei implizit vorhanden. Das bedeutet, dass nicht direkt auf die Datenelemente des Modells zugegriffen werden kann. Nach außen hin werden meist nur Darstellungsinformationen sichtbar. Das Programm gewährleistet die Konsistenz des Modells. Die wichtigsten Vertreter der Generativen Modelle sind *Constructive Solid Geometry* (*CSG*) Modelle.
- **Hybride Modelle:** CAD-Systeme vereinigen meistens beide oben genannten Modelltypen zu sogenannten Hybridmodellen, wobei die Entstehungshistorie in einem CSG-Modell abgebildet wird, das mit einem B-Rep-Modell synchronisiert ist. In diesem werden Operationen wie beispielsweise das Auswählen von Geometrieelementen (*Picking*) durchgeführt.

Im Rahmen dieser Arbeit sind vor allem akkumulative Modelle und im speziellen Geometrisch-Topologische Strukturmodelle interessant.

2.3.1.1 Geometrisch-Topologische Strukturmodelle (GT- bzw. B-Rep Modelle)

Boundary-Representation Modelle basieren auf der Definition der äußeren Hülle, die das Volumen eines Körpers umschließt. Die Modelle setzen sich aus einzelnen Geometrieelementen wie Flächen, Kanten und Punkten zusammen. Parallel zur Geometrie werden topologische Beziehungen zwischen den Geometrieelementen abgebildet. Einzelne Geometrieelemente werden durch die topologischen Beziehungen zu einer konkreten Gestalt verbunden [Grae-1989].

Die Topologie in B-Rep Modellen stellt eine komplexe, nicht streng hierarchische, netzwerkartige Struktur dar. Die Struktur des B-Rep Modells wird getrennt von der Erzeugungslogik

gespeichert. Dadurch ist eine ständige Überprüfung der Konsistenz des erzeugten Modells erforderlich. Bei B-Rep Modellen entfällt im Vergleich zum CSG Modell die ständige Neuberechnung des ganzen Modells nach jeder Modifikation, da die Modifikationen als topologische Operationen an der B-Rep Struktur durchgeführt werden und die Struktur nur lokal neu berechnet werden muss. Vorteilhaft ist ebenfalls die Möglichkeit des direkten Zugriffs auf alle Geometrieelemente des Modells (z.B. Picken von Kanten oder Punkten) [Grae-1989].

Geometrieelemente wie Linien (Kurven), Flächen und Volumina können analytisch oder parametrisch beschrieben werden. Analytische Beschreibungsformen beziehen sich auf ein kartesisches Koordinatensystem in dem die Elemente mit Hilfe der Vektorrechnung mathematisch beschrieben werden können. Das einfachste Geometrieelement ist der Punkt. Beispiele für analytische Geometrieelemente sind im eindimensionalen Fall Geraden, Strecken sowie Kegelschnittkurven. Im zweidimensionalen Fall gehören Flächen wie Ebenen, Zylinder oder Kugelflächen zu den analytisch beschreibbaren Elementen. Analytisch beschreibbare, dreidimensionale Volumen sind z.B. Quader, Kugeln und Zylinder.

Die parametrische Beschreibung von Geometrieelementen ist durch die Anforderung entstanden, beliebig gekrümmte Linien, Flächen und Volumina zu definieren, die analytisch nicht oder nur sehr aufwendig beschreibbar sind. Die parametrischen Beschreibungsverfahren ermöglichen die mathematische Definition beliebiger Krümmungen (Freiformkrümmungen) von Linien, Flächen und Volumina. Die parametrische Beschreibung einer Kurve kann wie folgt angegeben werden:

$P(t) = (x(t), y(t)), t \epsilon [0,1]$

Bei parametrischen Beschreibungsverfahren für Geometrieelemente (Kurven, Flächen, Volumina) bezieht sich die Definition der Geometrieelemente nicht auf ein Koordinatensystem sondern auf Größen (Parameter), die eine Eigenschaft des Elements verkörpern. Das heißt die Werte für die Parameter *t* sind nicht aus dem Koordinatensystem des Zielraums (*x, y, z*). Die einzelnen Koordinaten können auch von mehreren Parametern abhängig sein, sowie eine lineare, quadratische, oder andere Abhängigkeit von diesen besitzen. [BuGZ-2002].

Die topologische Beschreibung eines Geometriemodells ergänzt die geometrische Beschreibung durch Nachbarschaftsbeziehungen zwischen den Elementen des Gegenstands und beschreibt den Aufbau des Geometriemodells aus den Geometrieelementen und damit seine Struktur. Geometrische Elemente können verändert werden, ohne dass sich die Topologie des Modells ändert. Die Topologie verwendet, ebenso wie die Geometrie, grundlegende Elemente zur Beschreibung von Objekten. Dabei handelt es sich um *Eck*- oder *Endpunkte* (engl. *Vertex*), *Kanten* (engl. *Edge*), *Berandungen* (engl. *Loop*), *Oberflächen* (engl. *Face*) und *Körper* (engl. *Body*). Diese Elemente korrelieren mit den geometrischen Elementen *Punkt*, *Linie*, *Fläche* sowie *Volumen*.

Zwischen den verschiedenen Elementen bestehen bestimmte Bedingungen. So hat eine Kante beispielsweise immer zwei Endpunkte und wird immer von zwei Flächen geteilt. Eine Fläche wird immer von mindestens einer Berandungslinie begrenzt, die sich wiederum aus verschiedenen Kanten zusammensetzt. Diese und andere Beziehungen werden unter anderem dazu

benutzt, die Gültigkeit eines Körpers zu bestimmen. Ein Körper ist, vereinfacht gesagt, dann gültig, wenn er in seiner Form auch in der Natur vorkommen könnte.

Zwischen den geometrischen und topologischen Elementen bestehen vielfältige logische Zusammenhänge. So haben z. B. Punkte, Kanten und Flächen eines Produkts sowohl eine geometrische als auch eine topologische Bedeutung. So entsprechen z. B. die Oberflächen (faces) in der Topologie den Flächen in der Geometrie. Ebenso verhalten sich die Kanten zu Linien und die End- bzw. Eckpunkte zu den geometrischen Punkten (siehe Abbildung 9).

Die Topologie wird in modernen CAD-Systemen mit der Produktstruktur verknüpft. Das bedeutet, dass ein Geometriemodell ein Einzelteil beschreibt, dass wiederum zu einem Produkt gehört, das in Baugruppen und andere Einzelteile gegliedert ist. Abbildung 9 zeigt den Zusammenhang zwischen Topologie, Geometrie und Produktstruktur in B-Rep-basierten CAD-Systemen.

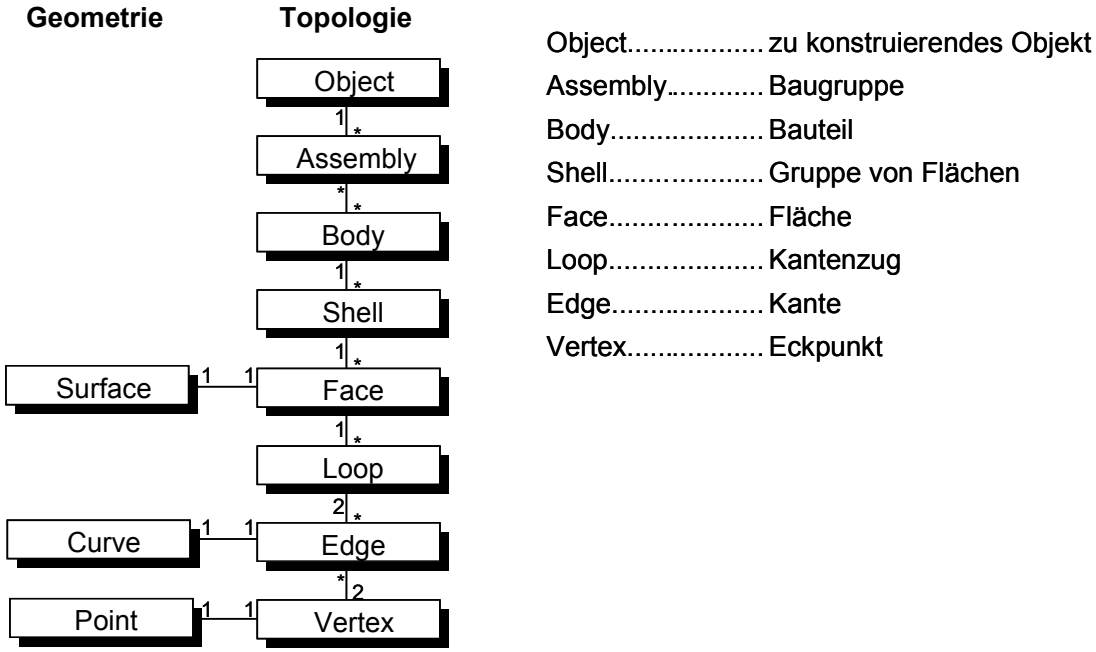


Abbildung 9: B-Rep Modell und Produktstruktur [Grab-2001]

Features, Parametrik und constraintbasierte Modellierung in 3D-CAD-Modellen

Unter *Features* werden Aggregationen von Merkmalen zu gemeinsamen Objekten verstanden [EnTa-1994]. Features können beliebig komplex sein, d.h. sie können aus mehreren Objekten aufgebaut sein, die ihrerseits wiederum Features oder geometrische Elemente sein können. Es werden verschiedene Arten von Features unterschieden:

- *Body Features* tragen nur geometrische Informationen und entsprechen Primitivvolumina (z. B. *Block*, *Wellenabschnitt*) oder Freiformvolumenelementen.

- *Form Features* tragen eine gewisse technische Bedeutung, d.h. neben geometrischen Informationen, die ihr Aussehen definieren tragen sie auch Informationen über ihren technischen Verwendungszweck, wie z.B. Passfedernuten
- *Operation Features* sind mit einem Bearbeitungsschritt verknüpft, z. B. Abrundungen und Fasen.
- *Enumerative Features* stellen mehrfach nach bestimmten Mustern angeordnete Objekte dar. Muster können z.B. kreisförmige Anordnungen von Objekten sein. Die angeordneten Objekte können ihrerseits einfache geometrische Elemente oder Features sein.

Die zunehmende Bedeutung und Integration der Feature- Technologie liegt in der einfachen und schnellen Erzeugung konstruktionsrelevant zusammengefasster Geometrien [SpKr-1997]. Ein Formfeature lässt sich aus verschiedenen Gesichtspunkten (z.B. Konstruktion, Fertigung, Montage) definieren und die Semantik hängt dabei primär vom Kontext der Featureverwendung ab.

Innerhalb des Produkt- und Geometriemodells (siehe Abbildung 9) fügen sich Features zwischen den Elementen Body (Bauteil) und Shell (Flächenverbund) und damit als neue Gliederungsebene ein. Sie stellen im CAD-Modell Objekte dar, die bei ihrer Erzeugung mit konkreten Attributwerten (Parameter) aus vorgegebenen Featuretypen gebildet werden. Dabei wird sowohl die Gestalt des neu zu erzeugenden Features als auch seine Lage, Orientierung und Platzierung innerhalb des Modells festgelegt.

Da nicht für alle Anforderungen standardisierte Features zur Verfügung stehen, bieten 3D-CAD-Systeme auch die Möglichkeit der *benutzerdefinierten Features* (*User Defined Features, UDF*). Features können neben der Semantik auch technologische Attribute, Regeln und Zwangsbedingungen (Constraints) beinhalten. Somit tragen Features implizites Konstruktionswissen, das nicht nur die Geometriemodellierung beschleunigt, sondern auch die nachfolgenden Phasen des Produktentstehungsprozesses vereinfachen kann [ShMä-1995].

Bei der *parametrischen Modellierung* in 3D-CAD-Systemen werden Geometrie und Bemaßung bidirektional gekoppelt. Dabei wird das rechnerinterne Modell mathematisch so beschrieben, dass Maße und andere die Geometrie bestimmende Größen (*Parameter*) über Bedingungen und mathematische Beziehungen (*Constraints*) miteinander verknüpft sind. Das heißt, die Geometrie beeinflusst die Bemaßung und umgekehrt beeinflusst eine Veränderung der Bemaßung die zugehörige Geometrie [ShMä-1995].

Parameter sind veränderliche Größen, die im Modell abgespeichert werden. Sie beschreiben im Allgemeinen geometrische Größen (Längen, Winkel, Koordinaten etc.), können aber auch nichtgeometrische Objekteigenschaften, wie z.B. Werkstoff oder maximal zulässige Torsionsmomente mit einem veränderlichen Wert definieren.

Constraints sind Zwangs- und Randbedingungen, die Abhängigkeiten zwischen Parametern definieren. Durch die Möglichkeit nichtgeometrische Parameter zu definieren und mittels Constraints zu verknüpfen, können Zusammenhänge zwischen einzelnen Modellobjekten be-

schrieben werden, die weit über die eigentliche Repräsentation der geometrischen Gestalt hinausgehen. Damit ist es möglich, Teilbereiche der Produkt- und Konstruktionslogik („Design Intent“) im Modell abzubilden. Constraints können durch mathematische Beziehungen zwischen Parametern ausgedrückt werden oder als zunächst nichtmathematische Zwangsbedingungen, wie z.B. Parallelität, Konzentrizität oder Orthogonalität. Für die Auswertung werden diese aber unter Zuhilfenahme der Vektorrechnung in mathematische Beziehungen systemintern umgewandelt. Eine Begrenzung des Potentials zur Informations- und Wissensabbildung im 3D CAD-System ergibt sich durch die Syntax und die Semantik der im 3D CAD-System vorhandenen Objekt- und Constrainttypen [ShMä-1995].

2.3.2 System Engineering (SE) Werkzeuge

System Engineering bedeutet nach der Definition der University College of London, Centre for System Engineering (UCLse) [UCLS-2004] (aus dem Englischen):

“System Engineering ist der Bereich der Ingenieurswissenschaften, der sich mit der Entwicklung großer und komplexer Systeme befasst, wobei unter einem System eine Menge von miteinander in Beziehung stehenden Elementen verstanden wird, die zur Bewältigung einer gemeinsamen Aufgabe zusammenarbeiten.“

System Engineering konzentriert sich dabei auf:

- Die Aufgaben solcher Systeme, die Dienstleistungen dieser Systeme sowie die Beschränkungen und Randbedingungen, denen diese Systeme unterliegen
- Die präzise Spezifikation der Systemstruktur und seines Verhaltens sowie die Realisierung der Spezifikationen
- Die Tätigkeiten, um sicherzustellen, dass die Spezifikationen und die Aufgaben durch ein System erfüllt werden
- Die Evolution solcher Systeme über die Zeit und über Systemgrenzen hinweg
- Die Prozesse, Methoden und Werkzeuge zur ökonomischen und zeitgerechten Entwicklung von Systemen

Die Disziplin des System Engineering wird im Produktentwicklungsprozess mit zunehmender Komplexität technischer Produkte immer wichtiger. In der Folge wächst der Bedarf an entsprechenden Werkzeugen in den entsprechenden Branchen, wie beispielsweise Automobil- und Luft und Raumfahrtindustrie stetig an.

Aus Sicht der Konstruktionstheorie nach VDI-Richtlinie 2221 [VDI-1993] umfassen SE-Werkzeuge vor allem Funktionalität zur Unterstützung der Phasen Funktions- und Prinzipmodellierung.

2.3.2.1 System Modeling Language (SysML)

Die System Modeling Language (SysML) stellt einen Ansatz für eine einheitliche, allgemein gültige, graphische Notation zur Beschreibung von Systemen dar [SML-2005]. Die Sprache unterstützt die Spezifikation, die Analyse, den Entwurf, die Verifikation sowie die Validierung von Systemen. Die abgebildeten Systeme können z.B. Hardware, Software, Informationen, Prozesse, Personal oder Produktionsmittel beinhalten. SysML basiert auf dem UML Standard 2.0 [UML-2003a][UML-2003b][UML-2003c][UML-2003d], der von der Object Management Group (OMG) [OMG-2005a] zur Zeit standardisiert wird. SysML wird zur Zeit in gemeinsamer Anstrengung von INCOSE (International Council on Systems Engineering) [INCO-2005] und der OMG definiert.

SysML ist eine Abwandlung von UML. Es wurden die Architektur und für den Entwurf von Systemen relevante Elemente von UML übernommen, wie z.B. Klassendiagramme, Sequenzdiagramme und Zustandsdiagramme sowie die in ihnen verwendeten Symbole für Entitäten und Beziehungen. Erweitert wurden diese durch zusätzliche Symbole für Systemelemente und den Beziehungen zwischen diesen. Zusätzlich wurde eine Reihe von neuen Konstrukten entwickelt, die wie in UML in *Strukturkonstrukte* (*Structural Constructs*), *Verhaltenskonstrukte* (*Behavioral Constructs*) und für beide gültige, *übergreifende Konstrukte* (*Crosscutting Constructs*) unterteilt werden können.

Zu den neuartigen Strukturkonstrukten gehören *Baugruppenkonstrukte* (*Assemblies*) und *Parametrische Konstrukte* (*Parametrics*). Ein Baugruppenkonstrukt beschreibt ein System als eine Menge von Komponenten, die jeweils spezielle Aufgaben im Kontext einer übergeordneten Aufgabe haben. Die Zugehörigkeit der Komponenten zu einem Baugruppenkonstrukt definiert die Systemgrenze des Konstrukts. In Baugruppenkonstrukten werden auch die Verbindungen dargestellt, über die die Komponenten miteinander interagieren können. Komponenten können Verbindungen nach außerhalb der Systemgrenze haben. Sie werden dann als *Ports* bezeichnet. Komponenten können selbst wiederum Baugruppenkonstrukte sein, sodass sich eine hierarchische Schachtelung von Systemen ergibt (siehe Abbildung 10).

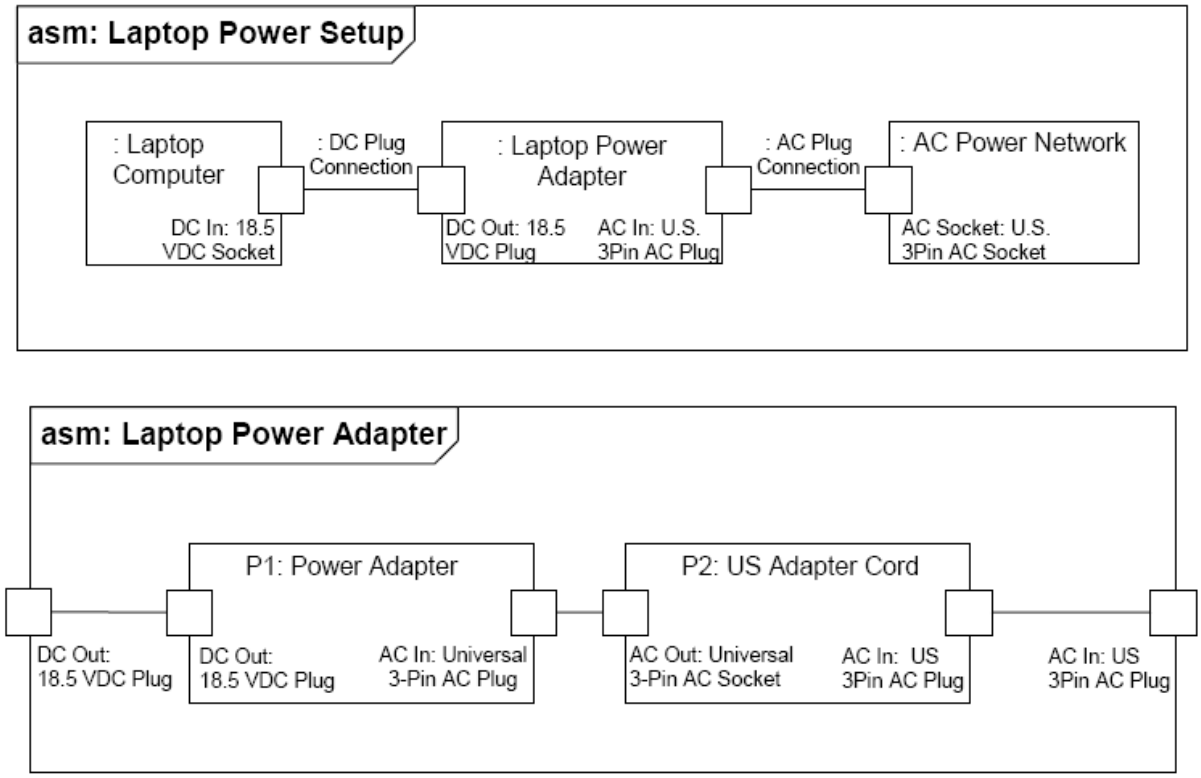


Abbildung 10:Stromversorgung eines Laptops modelliert als Baugruppenkonstrukt [SML-2005]

Parametrische Konstrukte beschreiben Netzwerke von parametrischen Abhängigkeiten zwischen den Eigenschaften eines Systems. Das bedeutet, dass sich damit die Auswirkungen einer Änderung an einem Parameter auf andere, damit verbundene Parameter beschreiben und auswerten lassen. Es entsteht ein Constraintnetz, das auch als mathematisches Gleichungssystem dargestellt werden kann. Abbildung 11 zeigt exemplarisch, wie Eigenschaften eines Baugruppenkonstrukts mit Parametrischen Konstrukten in Beziehung gebracht werden können.

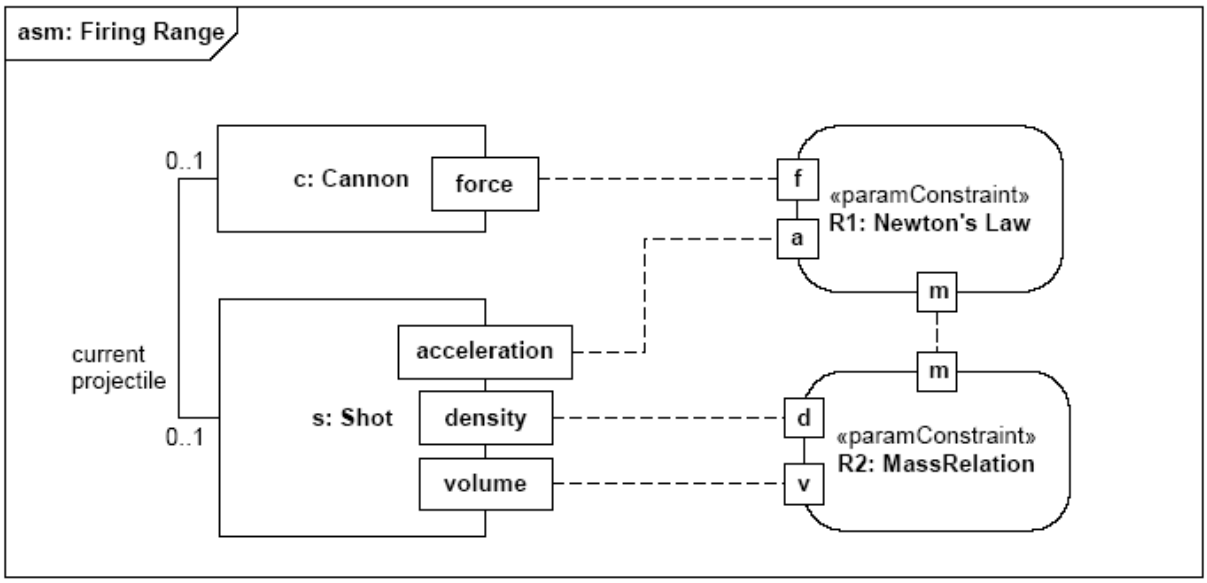


Abbildung 11: Constraintnetz in einem Assembly [SML-2005]

Anforderungen (*Requirements*) beschreiben Fähigkeiten oder Zustände, die ein System innehaben muss, bzw. soll. Dazu werden sie mit Systemelementen aus den Bereichen Analyse, Entwurf und Implementierung mit Hilfe der Beziehungsart *Erfüllung* (*Satisfaction* oder *satisfy*) verknüpft. Anforderungen können mittels Beziehungen des Typs *Überprüfung* (*Verification* oder *verify*) mit Testfällen verknüpft werden. Anforderungen können außerdem in weitere Unter-Anforderungen (*Subrequirements*) unterteilt werden, sodass sich eine Baumstruktur von zusammengesetzten Anforderungen (*Compound Requirements*) ergibt. Darüber hinaus können Anforderungen mit weiteren Anforderungen über die Beziehungsart *Ableitung* (*Derivation* oder *derive*) in Beziehung stehen. Abbildung 12 zeigt beispielhaft die Verknüpfung von Anforderungen mit Entwurfselementen.

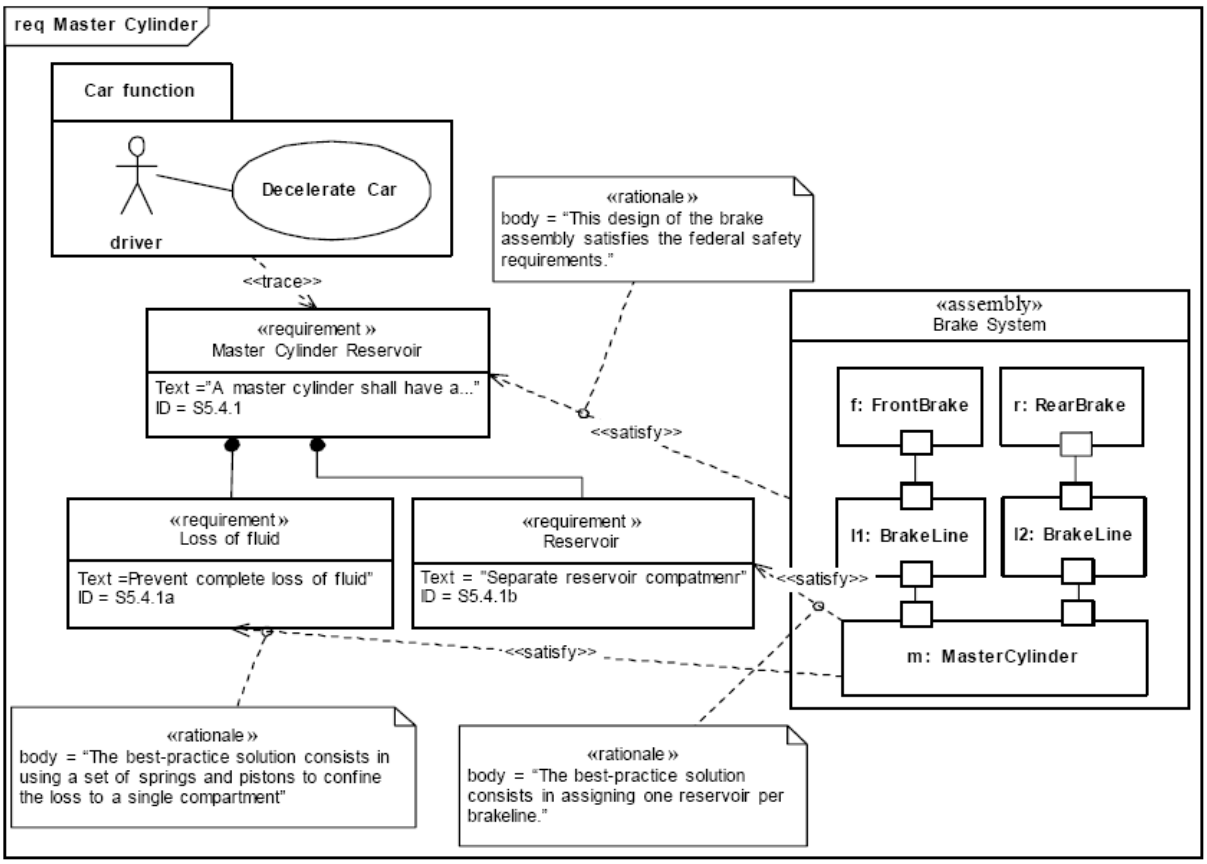


Abbildung 12: Verknüpfung zwischen Anforderungen und Entwurf [SML-2005]

Bei SysML handelt es sich um eine graphische Beschreibungssprache, die für eine einheitliche Beschreibung von Systemen aller Art sorgen soll und damit in erster Linie die Kommunikation von Menschen erleichtern soll. Eine rechnerunterstützte Abbildung und Auswertung von Abhängigkeiten zwischen Produktmerkmalen ist damit noch nicht gegeben.

2.3.2.2 Das System RODON® der Firma R.O.S.E Informatik

Das Softwaresystem RODON wird von der Firma R.O.S.E Informatik GmbH entwickelt und vertrieben [Rose-2005]. Es wurde ursprünglich zur Nachbildung von bereits existierenden Systemen und anschließender Simulation, Fehlerdiagnose und FMEA[20]-Analysen konzipiert. Genauso können aber auch Systeme in den frühen Entwurfsphasen der Funktions- und Prinzipfindung mit RODON modelliert und gefundene (modellierte) Lösungen frühzeitig evaluiert werden.

Mit dem Werkzeug lassen sich Funktionsstrukturen modellieren, die hierarchisch aus Subsystemen und Einzelkomponenten aufgebaut sind. Systeme und Komponenten können beliebig viele Ein- und Ausgänge besitzen, so genannte Input- und Output-Ports. Jeder Output-Port kann mit jedem Input-Port innerhalb des selben Systems verbunden werden, sofern die über-

20 FMEA: Failure Modes and Effects Analysis. Systematische Risikoanalysemethode

tragenen Größen kompatibel zueinander sind. Das heißt, ein Ausgang, der elektrische Spannung führt, kann nur an einen Eingang angeschlossen werden, der elektrische Spannung erwartet. Abbildung 13 zeigt RODONs Editor zur Modellierung von Systemen.

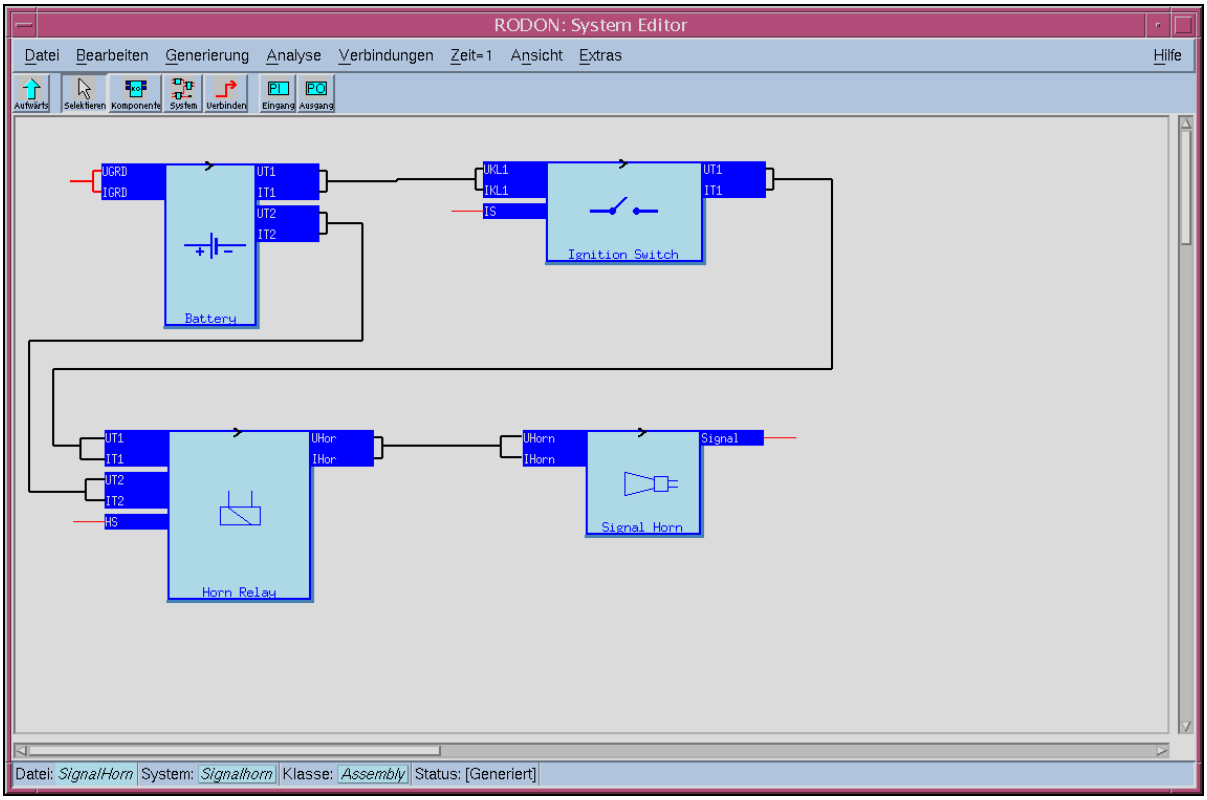


Abbildung 13: Der Systemeditor von RODON

In RODON können hierarchisch gegliederte Komponentenklassen definiert werden, aus denen Komponenten erzeugt (instanziiert) werden. Komponenten einer Klasse können die Eigenschaften ihrer übergeordneten Klassen erben. Komponenten werden in einem speziellen Editor modelliert. In diesem Model-Editor kann spezifiziert werden, wie die einzelnen Eingänge mit den Ausgängen „verschaltet“ werden. Das heißt, es lassen sich beispielsweise mathematische Formeln angeben, die spezifizieren, wie eine oder mehrere Eingangsgrößen in eine oder mehrere Ausgangsgrößen abzubilden sind. Es können aber auch Wertetabellen als Abbildungsvorschriften angegeben werden. Zusätzlich lassen sich verschiedene Betriebszustände der Komponenten mit ihren eigenen Abbildungsvorschriften definieren. Auf diese Weise lassen sich beispielsweise das Nominal- und das Fehlerverhalten einer Komponente abbilden, was unter anderem bei Simulationen ausgenutzt wird.

In RODON sind keine Mechanismen zur automatischen Erkennung von impliziten Abhängigkeiten gegeben. Das System ist vielmehr ein Werkzeug, um Abhängigkeiten explizit zu modellieren und auszuwerten.

2.3.3 Ansätze aus der Forschung

2.3.3.1 Ermittlung von Produktfunktionen aus 3D-CAD-Gestaltmodellen

Das Verfahren von KUNZE [Kunz-2002] erlaubt es, implizit in einem 3D-CAD-Gestaltmodell vorhandene Informationen aus den frühen Konstruktionsphasen Funktionsmodellierung und Prinzipmodellierung zu rekonstruieren und mit den Elementen des Gestaltmodells zu verknüpfen. Der Ansatz empfindet die natürliche, menschliche Vorgehensweise zur Ermittlung der Funktionsweise von unbekannten, physischen Objekten nach, indem Bewegungen der modellierten, mechanischen Komponenten von außen virtuell initiiert und die Reaktionen des Systems interpretiert werden. Dadurch werden zusätzlich zu möglicherweise im CAD-System bereits abgebildeten Beziehungen (Constraints) zwischen Produktkomponenten weitere Abhängigkeiten ermittelt und abgebildet. Das Verfahren beruht auf sechs Schritten:

Im ersten Schritt wird die Produktgeometrie analysiert und höherwertige semantische Informationen über das Produkt ermittelt. Im zweiten Schritt werden virtuelle Interaktionen mit dem Produkt ausgelöst. Dazu werden translatorische und/oder rotatorische Bewegungen über die zuvor ermittelten Angriffspunkte initiiert und die Bewegung über die sich berührenden Einzelteile propagiert um die *Kinematische Wirkstruktur* zu bestimmen. Die Ermittlung der Prinzipstruktur für jeweils einen Interaktionsfall als dritter Schritt erfolgt durch die Zuhilfenahme einer Lösungsmusterbibliothek. In dieser Bibliothek werden Teilgraphen der Kinematischen Wirkstruktur mit dort abgelegten Mustergraphen verglichen und die den Mustergraphen zugeordneten Wirkprinzipien (z.B. *Hebel*, *Feder*, *Keil*) bei Übereinstimmung der Graphen ausgewählt. Im vierten Schritt wird aus der Prinzipstruktur die Funktionsstruktur für den jeweiligen Interaktionsfall ermittelt. Vorausgesetzt wird dazu, dass zwischen Prinzipien und Funktionen eine 1:m Beziehung besteht, d.h. dass eine Funktion genau einem Prinzip zugeordnet werden kann (umgekehrt kann ein Prinzip mehreren Funktionen zugeordnet werden). Aus der Funktionsstruktur wird im fünften Schritt die Aufgabenstruktur des aktuellen Interaktionsfalls ermittelt. Dazu wird die Funktionsstruktur vereinfacht und zusammengefasst und aus Einzelfunktionen übergeordnete Gesamtfunktionen ermittelt. Im sechsten Schritt werden die Ergebnisse der vorherigen Schritte für die einzelnen Interaktionsfälle zusammengefasst und daraus die wahrscheinliche Gesamtfunktion des zu untersuchenden Produkts ermittelt.

Aus dem Ansatz nach Kunze können über den Weg der Funktionsinterpretation Abhängigkeiten zwischen Geometrieelementen erkannt werden, die zunächst nur implizit im untersuchten Modell vorlagen. Die Erkennung und Explizierung von Abhängigkeiten zwischen Geometrieelementen lag jedoch nicht im Fokus der Arbeiten.

2.3.3.2 Der Ansatz aus iViP

In iViP[21] sollten neue Ansätze zur integrierten, virtuellen Produktentstehung entwickelt und in einem Framework, das aus einer Integrationsplattform sowie zugehörigen Werkzeugen besteht, implementiert werden. Dazu wurde eine Client-Server-Framework entwickelt, das im wesentlichen aus der sogenannten *iViP-Integrationsplattform* und verschiedenen *iViP-Werkzeugen* besteht. Die Integrationsplattform besteht aus dem iViP-Client, der die Interaktion mit dem Benutzer steuert und verschiedenen Systemdiensten, die den Informationsaustausch innerhalb der Integrationsplattform und zwischen der Integrationsplattform und ihrer Umgebung koordinieren (Session-Management, Benutzerverwaltung etc.). Die Integrationsplattform kommuniziert über CORBA[22] mit den iViP-Werkzeugen, die die eigentliche Funktionalität zur virtuellen Produktentstehung bereitstellen (siehe Abbildung 14)[KrTA-2002].

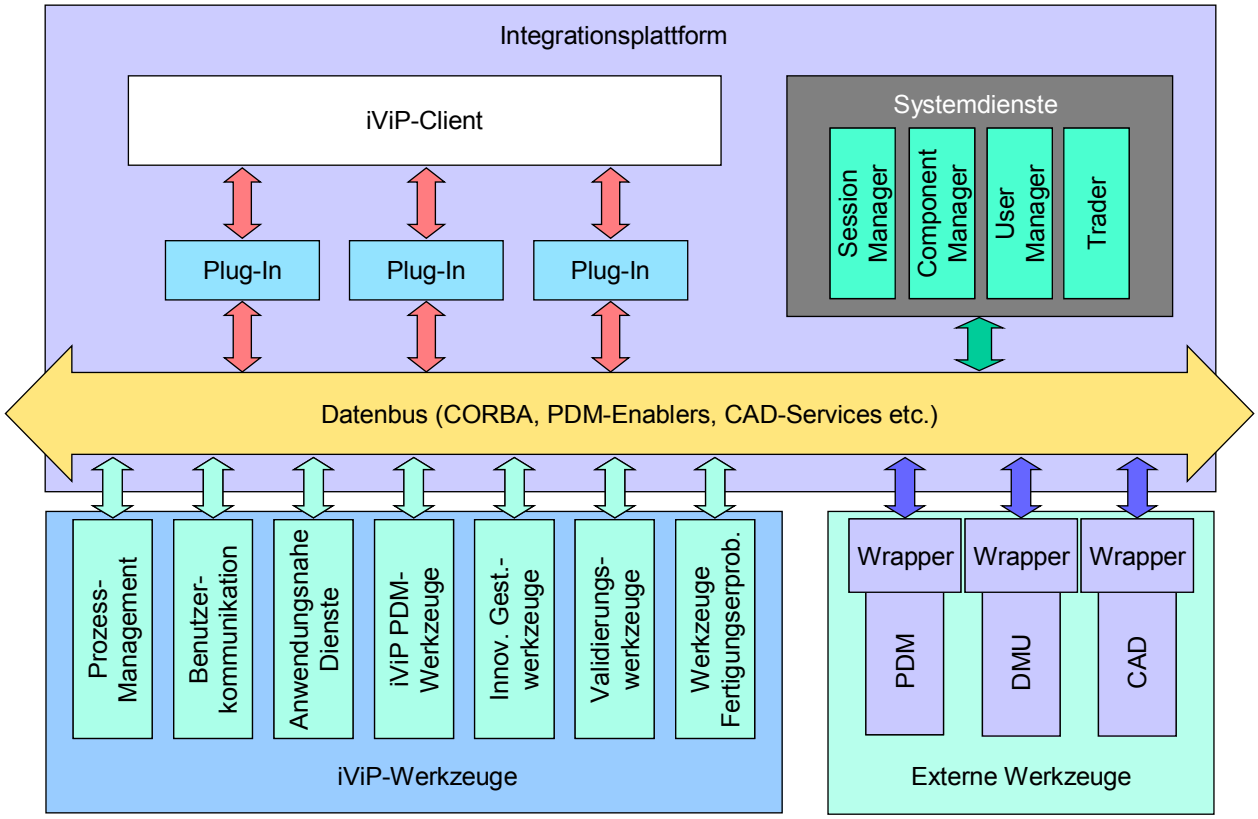


Abbildung 14: iViP-Systemarchitektur [KrTA-2002]

Zusätzlich können fremde Softwaresysteme wie z.B. PDM-Systeme oder CAD-Systeme angebunden werden, die ebenfalls über CORBA mit der Integrationsplattform kommunizieren. Dazu müssen sogenannte *Wrapper* für jedes anzubindende System implementiert

21 Das Leitprojekt *Innovative Technologien und Systeme für die integrierte, virtuelle Produktentstehung* (iViP) wurde durch das Bundesministerium für Bildung und Forschung (BMBF) gefördert (Projektlaufzeit 1998-2002) [KrTA-2002].

22 CORBA: Common Object Request Broker Architecture. Plattform für die Interaktion verschiedener Programme.

werden, die eine semantisch korrekte Abbildung ihrer Modellinhalte auf die verwendeten Standards gewährleisten. Für den Austausch von Daten zur Produktverwaltung wurde der OMG-Standard *PDM-Enablers* benutzt und für den Austausch von Geometriedaten der OMG-Standard *CAD-Services*.

Wie oben erwähnt, steuert der iViP-Client die Interaktion mit dem Benutzer des Systems. Dabei besitzt der Client nur rudimentäre Logik, die sich auf allgemeine Funktionen zur Anbindung der iViP-Werkzeuge, sowie auf die Darstellung von Informationen beschränkt. Das eigentliche Verhalten des Client wird durch die angebundenen Werkzeuge bestimmt. Der Client dient somit lediglich als Benutzeroberfläche, das heißt, er stellt die Informationen aus den Werkzeugen auf dem Bildschirm dar und gibt die Benutzereingaben, sei es über Maus oder Tastatur, an die jeweils angebundenen Werkzeuge weiter. Dazu wurde für jedes Werkzeug eine spezielle Benutzeroberfläche erstellt, die über eine definierte Schnittstelle in den Client eingebunden und von diesem verwaltet wird (Plug-In).

Besonders interessant im Rahmen dieser Arbeit ist das Werkzeug zum Funktionsorientierten Entwerfen *FOD (Function-oriented Design*). Im FOD-Werkzeug sind Assistenzsysteme für die Definition von Produktanforderungen, von Funktionen und Funktionsstrukturen sowie von Bauteilstrukturen (Strukturmodelle) zusammengefasst. Die den Assistenzsystemen zugrunde liegenden Teilmodelle sind über ein übergreifendes Constraintmodell miteinander verknüpft (siehe Abbildung 15), das mit Hilfe eines eigens dafür entwickelten Assistenzsystems verwaltet und gepflegt werden kann. Geometriemodelle aus externen CAD-Systemen werden über Verknüpfungen mit Elementen des Strukturmodells eingebunden. Softwaretechnisch ist die Anbindung der externen CAD-Systeme über die CAD-Services Schnittstelle realisiert.

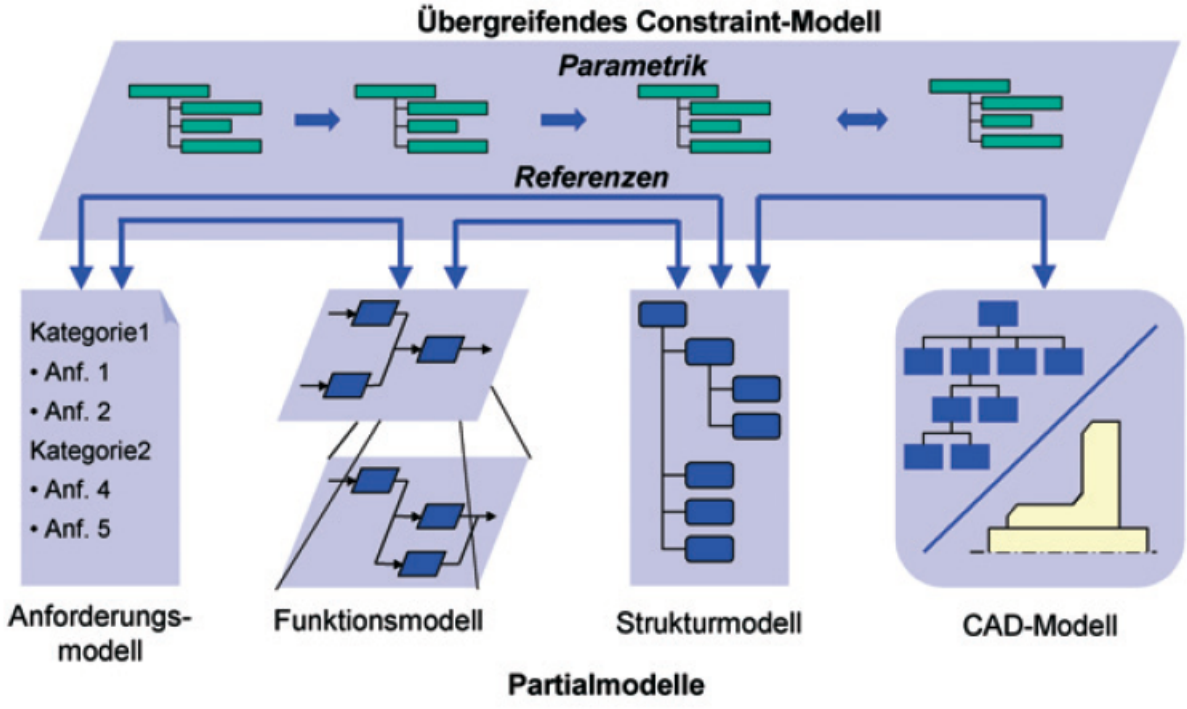


Abbildung 15: Die Teilmodelle des FOD-Werkzeugs [KrTA-2002]

Im FOD werden Abhängigkeiten zwischen Produktmerkmalen explizit definiert. Eine selbständige Erkennung von Abhängigkeiten lag nicht im Fokus des Forschungsprojekts. Daher ist die Vorgehensweise bei Neukonstruktionen mit größerem Aufwand verbunden.

2.3.3.3 Property Driven Development/Design (PDD)

Der Ansatz des Property Driven Development/Design (PDD) wurde maßgeblich am Lehrstuhl für Konstruktionstechnik/CAD (LKT) der Universität des Saarlandes in Saarbrücken entwickelt [WeDe-2003].

Kernpunkt des Ansatzes ist die Unterscheidung zwischen *Merkmalen* und *Eigenschaften* eines Produkts. Die Merkmale eines Produkts, wie z.B. Gestalt, Abmessungen, verwendete Werkstoffe etc. definieren das Produkt. Sie werden vom Konstrukteur direkt festgelegt. Eigenschaften dagegen beschreiben im weitesten Sinne das Verhalten eines Produkts und treten als Wirkung der Merkmale und anderer Faktoren in Erscheinung. Sie können daher nur indirekt durch die definierten Merkmale vom Konstrukteur festgelegt werden. Zu den Eigenschaften eines Produkts werden beispielsweise das Gewicht, die ästhetische Wirkung, die Montagegerechtheit, die Prüfgerechtheit oder die Gebrauchsgerechtheit gezählt. Kunden interessieren in erster Linie die Eigenschaften eines Produkts. Eigenschaften werden weiter in geforderte Eigenschaften (Soll-Eigenschaften) und tatsächliche Eigenschaften unterschieden (Ist-Eigenschaften). Aufgabe des Konstrukteurs ist es daher, die Merkmale des Produkts so zu definieren, dass es die geforderten Eigenschaften besitzt.

Zwischen Merkmalen und Eigenschaften bestehen zwei grundlegende Zusammenhänge:

- **Analyse**: Aus den Merkmalen eines Produkts werden dessen Eigenschaften ermittelt oder, falls das Produkt noch nicht physisch existiert, durch Simulation, Berechnung oder andere Verfahren vorhergesagt.
- **Synthese**: Aus den geforderten Eigenschaften werden die erforderlichen Merkmale des Produkts bestimmt.

Merkmale und Eigenschaften stehen durch *Relationen* miteinander in Beziehung. Sie beschreiben, wie Merkmale Eigenschaften beeinflussen und wie sich Eigenschaften auf Merkmale auswirken.

In diesem Ansatz werden Vorgehensweisen und Modellkonstrukte vorgeschlagen, um Abhängigkeiten zwischen Produktmerkmalen (nach der dieser Arbeit zugrunde liegenden Terminologie) abzubilden. Die rechnerunterstützte Ermittlung von Abhängigkeiten wurde bisher nicht untersucht.

2.3.3.4 Der Ansatz aus dem Design Repository Project (DRP)

Das *Design Repository Project* (*DRP*), das vom US-amerikanischen National Institute of Standards and Technology (NIST) [NIST-2005] durchgeführt wurde, beschäftigte sich mit der Frage, wie unternehmensübergreifende, kollaborative Entwicklungsprozesse durch die Abbildung und den Austausch von Konstruktionswissen unterstützt werden können [SzSR-2001], [Szyk-2000].

Dazu wurde das Konzept der *Design Repositorys* (*DR*) entwickelt. DR sind objektorientierte Konstrukte und beinhalten neben der Beschreibung des zu entwickelnden Produkts (*Artefakte, Artifacts*) zusätzlich Wissen[23] über Anforderungen (*Requirements*), Spezifikationen (*Specifications*), Konstruktionsbegründungen (*Design Rationale*), Zwangsbedingungen (*Constraints*) und allgemeine Beziehungen (*Relationships*) zwischen Elementen der DR. Artefakte wiederum werden durch geometrische Informationen (*Form*), Informationen über seine Funktion (*Function*) und sein Verhalten (*Behaviour*) beschrieben. Die Elemente von DR können mit Referenzen (*Reference*) zu beliebigen externen Informationen versehen werden, wie z.B. CAD-Dateien. Funktionen können hierarchisch zu Funktionsstrukturen aufgebaut werden und können Stoff-, Energie- und Informationsflüsse übertragen. Des Weiteren sind sie mit Elementen der Artefaktrepräsentation verbunden. Abbildung 16 zeigt beispielhaft die Darstellung einer Funktionsstruktur in DRP.

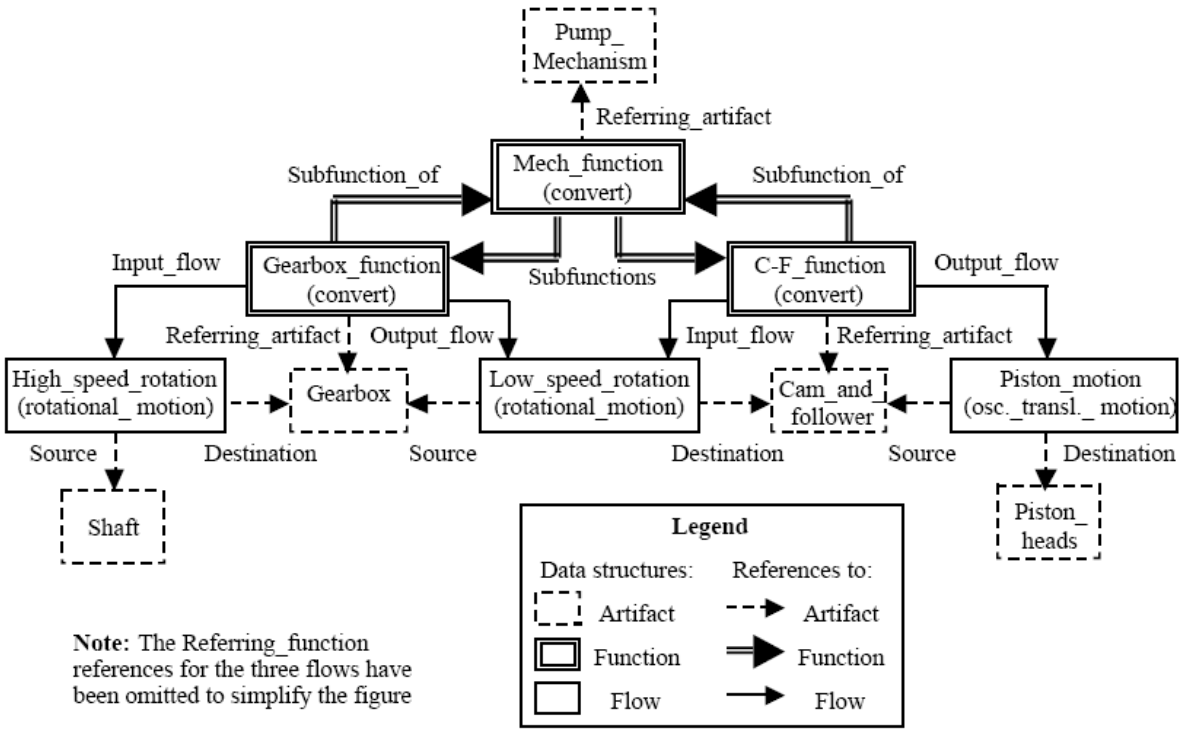


Abbildung 16: Beispiel einer Funktionsdarstellung nach dem Ansatz von DRP [Szyk-2000]

23 Obwohl in den diesbezüglichen Veröffentlichungen zu diesem Projekt von Wissen (engl. *Knowledge*) die Rede ist, entsprechen die dort aufgeführten Sachverhalte eher dem auch dieser Arbeit zugrunde liegenden Begriff der *Information* nach RUDE [RUDE-1998].

Um die Interaktion verschiedener Produktentwicklungswerkzeuge zu ermöglichen, wurde die Notwendigkeit einer neutralen Repräsentation von Artefakten erkannt. Die Beschreibung von Artefakten erfolgt in Anlehnung an das Anwendungsprotokoll AP 203 des STEP Datenmodells (siehe Abschnitt 2.2.1, Seite 18). Die relevanten Produktmodellinhalte der anzubindenden Werkzeuge müssen dazu in dieses Datenformat abgebildet werden. Die Konstrukte zur Repräsentation von Artefakten wurden in einem Kernmodell mit der Bezeichnung *Common Core Object* (*CCO*) zusammengefasst. Im CCO sind auch Konstrukte zur Abbildung von Spezifikationen enthalten. Ergänzt wird das CCO durch das Paket *Common Core Relationship* (*CCR*), in dem Konstrukte zur Beschreibung der Beziehungen zwischen den Entitäten der DR zusammengefasst sind [Szyk-2001]. Abbildung 17 zeigt schematisch den Aufbau des Kernmodells.

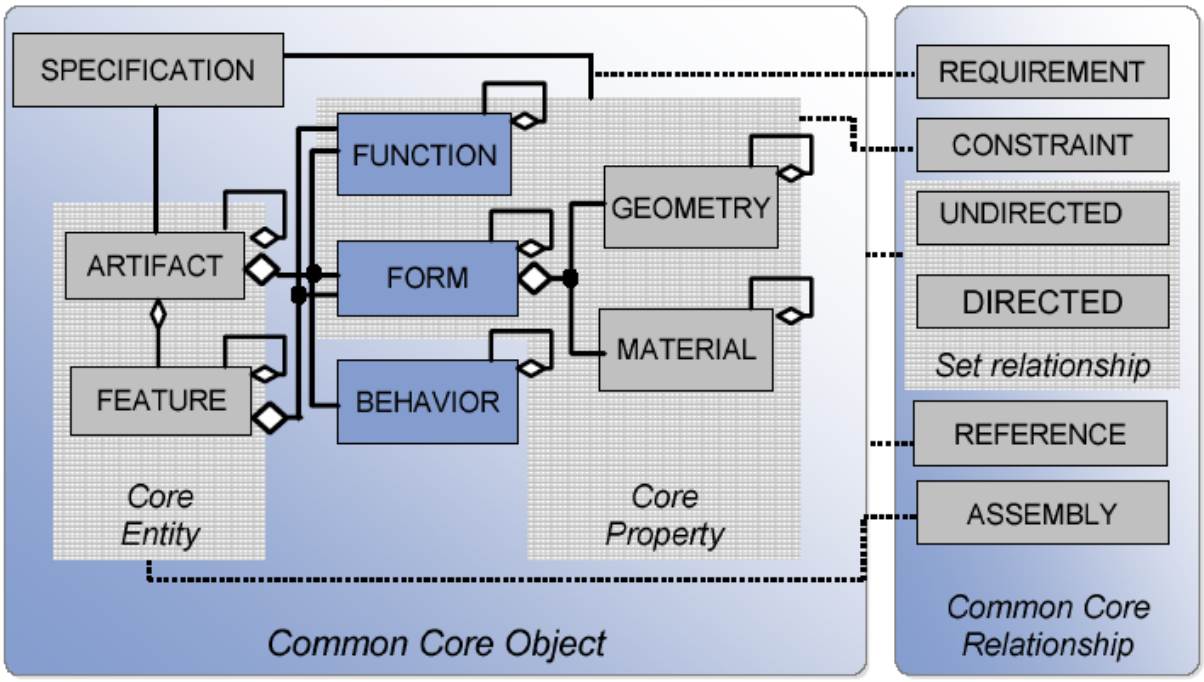


Abbildung 17: Das Kernmodell von DRP [Szyk-2001]

In diesem Projekt wurden ebenfalls Datenmodellkonstrukte und Vorgehensweisen zur Abbildung von Abhängigkeiten zwischen Produktmerkmalen vorgeschlagen. Eine rechnerunterstützte Ermittlung von implizit gegebenen aber nicht explizit modellierten Abhängigkeiten wird nicht betrachtet.

2.3.3.5 Konfigurations- und Verträglichkeitsmatrix (K-V-Matrix) nach Bongulielmi

Bongulielmi [Bong-2002] schlägt mit dem Konzept der *Konfigurations- und Verträglichkeitsmatrix* (*K-V-Matrix*) ein Matrizensystem zur Verknüpfung von Produktanforderungen aus Kundensicht und den technischen Eigenschaften eines Produkts vor. Das System dient zur Unterstützung von Konfigurationsproblemen, bei denen aus Kundenanforderungen direkt technische Lösungen konfiguriert werden sollen. Das Konzept setzt daher eine bestehende,

modulare Produktarchitektur mit entsprechend vorentwickelten, modularen Produktkomponenten voraus, aus denen das fertige, auf den Kunden zugeschnittene Endprodukt zusammengestellt werden soll.

Die *Verträglichkeitsmatrix*, *oder V-Matrix*, stellt die Kombinierbarkeit der Eigenschaften eines Produkts dar. Das heißt, dass die Reihen und Spalten der Matrix symmetrisch mit den selben Eigenschaften belegt sind und die Kombinierbarkeit jeder Eigenschaft mit jeder der anderen Eigenschaften durch eine „1“ für „kombinierbar“ oder einer „0“ für „nicht kombinierbar“ in der jeweiligen Zelle gekennzeichnet ist. Abbildung 18 zeigt als Beispiel die Verträglichkeitsmatrix für ein (fiktives) Fahrrad.

		Anhänger-Bereifung				Anhänger-Farbe				Anhänger-Typ		Bereifung				Federung		Kindersitz		Lenker
tmlName2	tmlaName2	hohes Profil	Slicks	normales Profil	keine	blau	gelb	rot	keine	Anhänger Typ A	nicht vorhanden	Alu - hohes Profil	Alu - normales Profil	Alu - Slicks	Stahl - hohes Profil	Federung Typ 1	nicht vorhanden	Kindersitz Typ 1	nicht vorhanden	Mountain
Anhänger-Bereifung	hohes Profil					1	1	1	1	1	1	1	1	1	1	1			1	1
	Slicks					1	1	1	1	1	1	1	1	1	1	1			1	1
	normales Profil					1	1	1	1	1	1	1	1	1	1	1			1	1
	keine					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Anhänger-Farbe	blau	1	1	1	1					1	1	1	1	1	1	1			1	1
	gelb	1	1	1	1					1	1	1	1	1	1	1			1	1
	rot	1	1	1	1					1	1	1	1	1	1	1			1	1
	keine	1	1	1	1					1	1	1	1	1	1	1	1	1	1	1
Anhänger-Typ	Anhänger Typ A	1	1	1	1	1	1	1	1			1	1	1	1	1			1	1
	nicht vorhanden	1	1	1	1	1	1	1	1			1	1	1	1	1	1	1	1	1
Bereifung	Alu - hohes Profil	1	1	1	1	1	1	1	1	1	1					1		1	1	1
	Alu - normales Profil	1	1	1	1	1	1	1	1	1	1					1		1	1	1
	Alu - Slicks	1	1	1	1	1	1	1	1	1	1						1			1
	Stahl - hohes Profil	1	1	1	1	1	1	1	1	1	1					1		1	1	1
Federung	Federung Typ 1	1	1	1	1	1	1	1	1	1	1	1	1		1			1	1	1
	nicht vorhanden				1				1		1			1						1
Kindersitz	Kindersitz Typ 1				1				1		1	1	1		1	1				1
	nicht vorhanden	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1
Lenker	Mountain	1	1	1	1	1	1	1	1	1	1	1			1	1		1	1	1
	Renn				1				1		1			1			1			1
	Tour	1	1	1	1	1	1	1	1	1	1		1			1		1	1	1
Rahmen-Grösse	26"	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	28"	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	30"	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Rahmen-Typ	Damen	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Herren	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Schaltung	18-Gang	1	1	1	1	1	1	1	1	1	1		1			1		1	1	1
	24-Gang				1				1		1			1			1			1
	27-Gang	1	1	1	1	1	1	1	1	1	1	1			1	1		1	1	1

Abbildung 18: Verträglichkeitsmatrix der Eigenschaften eines Fahrrads[Bong-2002]

Die in der V-Matrix betrachteten Eigenschaften des Produkts stellen immer eine bestimmte Sicht auf das Produkt dar, z.B. eine eher technische Sicht, wie in Abbildung 18.

Sollen zwei Sichten miteinander verknüpft werden, wird die *Konfigurationsmatrix*, *oder K-Matrix*, eingesetzt. Im einfachsten Fall verknüpft die K-Matrix die technische Sicht der Pro-

dukteigenschaften mit der Sicht der Kundeneigenschaften. Sie bildet dann die Ist-Eigenschaften auf die Soll-Eigenschaften eines Produkts ab. Abbildung 19 zeigt beispielhaft die Konfigurationsmatrix zur Verknüpfung der technischen Produkteigenschaften mit den für Kunden relevanten (Soll-) Eigenschaften eines Fahrrads. Es können aber beliebig komplexe Matrizensysteme erstellt werden, die verschiedene Sichten auf das Produkt (bzw. den Produktbaukasten) - repräsentiert durch entsprechende V-Matrizen - und deren Abbildung aufeinander - repräsentiert durch K-Matrizen - darstellen. So lassen sich beispielsweise auch Sichten aus parametrierten CAD-Modellen in Verträglichkeitsmatrizen abbilden, eine softwaretechnische Anbindung vorausgesetzt.

		Fahrer		Personengrösse			Ausführung			Anhänger			Anhängerfarbe				Kindersitz	
		Dame	Herr	< 170 cm	170 cm - 185 cm	> 185 cm	Komfort	Renn	Mountain	geländegängig	nicht vorhanden	strassengängig	rot	blau	gelb	keine	vorhanden	nicht vorhanden
Anhänger-Bereifung	hohes Profil									1								
	Slicks											1						
	normales Profil											1						
	keine										1							
Anhänger-Farbe	blau													1				
	gelb														1			
	rot												1					
	keine															1		
Anhänger-Typ	Anhänger Typ A						1		1	1		1						
	nicht vorhanden						1	1	1		1							
Bereifung	Alu - hohes Profil								1									
	Alu - normales Profil						1											
	Alu - Slicks							1										
	Stahl - hohes Profil								1									
Federung	Federung Typ 1						1		1									
	nicht vorhanden							1										
Kindersitz	Kindersitz Typ 1						1		1								1	
	nicht vorhanden						1	1	1									1
Lenker	Mountain								1									
	Renn							1										
	Tour						1											
Rahmen-Grösse	26"			1														
	28"				1													
	30"					1												
Rahmen-Typ	Damen	1																
	Herren		1															
Schaltung	18-Gang						1											
	24-Gang							1										
	27-Gang								1									

Abbildung 19: Konfigurationsmatrix eines Fahrrads[Bong-2002]

Der Ansatz der K-V-Matrizen eignet sich vor allem, wenn einfache Zusammenhänge abgebildet werden sollen. Nicht geeignet ist das Verfahren, wenn komplexere Zusammenhänge

abgebildet und untersucht werden müssen. Wie die Abhängigkeiten, abgesehen von manuellen Tätigkeiten durch den Menschen, ermittelt werden sollen, wird in dem Ansatz nicht betrachtet.

2.4 Fazit

Es gibt eine Vielzahl von Softwarewerkzeugen, die den Konstrukteur in unterschiedlichen aber isolierten Aspekten der Produktmodellierung unterstützen. In ihnen werden Eigenschaften des virtuellen (noch zu fertigenden) Produkts festgelegt und in einem rechnerinternen Modell abgebildet. Dabei gibt es für die unterschiedlichen Modellierungsmethoden jeweils spezialisierte Werkzeuge, die als Insellösungen verfügbar sind und Datenaustausch meist nur mit Werkzeugen ihrer eigenen Domäne erlauben. Dazu zählen beispielsweise CAD-Systeme, die über Neutralfileschnittstellen Geometrieinformationen für den Import durch andere CAD-Systeme ausleiten können. Eine Ausnahme bildet STEP, das aber die in Abschnitt 2.2.1. auf Seite 18 beschriebenen Probleme bereitet. Nach dem heutigen Stand der Technik sind keine kommerziellen Werkzeuge verfügbar, die einen systemübergreifenden und integrierten Zugriff auf Eigenschaften von Produktmodellinhalten in Systemen aus unterschiedlichen Domänen erlauben. Ausnahmen bilden FEM-Werkzeuge und Kinematiksimulationsprogramme, die allerdings als hybride Werkzeuge mit starkem Geometriebezug anzusehen sind.

In der Forschung sind die theoretischen Grundlagen zur Anforderungsmodellierung, zur Bewertung von Lösungen auf Grund von Anforderungen, zur Lösungsfindung auf Basis von Anforderungen sowie die Abhängigkeiten zwischen Produktmerkmalen untereinander gut beschrieben und teilweise auch in Softwareprototypen umgesetzt. Sie setzen aber voraus, dass alle entsprechenden Informationen, sowie das benötigte Wissen bereits vorhanden ist. Außer acht gelassen wurde bisher dagegen die interessante Frage, wie diese Informationen und das Wissen über Abhängigkeiten algorithmisch akquiriert werden können.

3 Konzept

Aufgrund der in Abschnitt 2.4 dargelegten Defizite gerade im Bereich der Forschung soll in dieser Arbeit ein Konzept vorgestellt werden, mit dem sich die hohen Erwartungen, die an die anforderungsgetriebene, integrierte Produktentwicklung gestellt werden, auch in der Praxis erfüllt werden können. Das bedeutet, dass dem Konstrukteur ein System an die Hand gegebenen werden soll, das ihn bei der Formulierung der Produktanforderungen sowie bei der Evaluierung von Produkteigenschaften anhand der Anforderungen effektiv unterstützt. Des Weiteren soll das System eine domänenübergreifende, integrierte Sicht auf das zu entwickelnde Produkt bieten, die es erlaubt, Auswirkungen von isolierten Konstruktionsentscheidungen auf alle Bereiche des Produkts zu erkennen und zu quantifizieren. Das System soll in einem Unternehmen verwendete Produktmodellierungswerkzeuge ergänzen und integrieren aber nicht ersetzen. Es soll lediglich im Bereich der Anforderungsdefinition als Autorensystem fungieren.

3.1 Anforderungen an ein System zur anforderungsgetriebenen, integrierten Produktentwicklung

Die Anforderungen aus dem Anspruch, eine anforderungsgetriebene, integrierte Produktentwicklung mit dem Rechner zu unterstützen, lassen sich in drei Teilbereiche untergliedern: Um Produkteigenschaften anhand von Anforderungen zu evaluieren, müssen diese in isolierten Modellierungswerkzeugen vorliegenden Informationen miteinander in Beziehung gebracht und verknüpft werden. Die Integration verschiedener Ingenieursdisziplinen in komplexen Produkten erfordert außerdem eine domänenübergreifende Sicht auf das Gesamtprodukt und die Erkennung, Abbildung und Auswertung von Abhängigkeiten zwischen Produktmerkmalen aus den unterschiedlichen Domänen. Um die beiden ersten Anforderungen zu erfüllen, muss als drittes die grundlegende Aufgabe gelöst werden, auf die Informationen in den anzubindenden Systemen zuzugreifen und gleichzeitig das Portfolio der angebundenen Werkzeuge flexibel erweitern zu können, ohne unnötige Eingriffe am System erforderlich zu machen.

3.1.1 Verknüpfung von korrespondierenden Produktmerkmalen

Grundlage für eine rechnerunterstützte, anforderungsgetriebene Produktentwicklung ist, dass die vom Konstrukteur definierten Produkteigenschaften rechnerintern anhand der Anforderungen evaluiert werden können, auf Basis derer sie ausgeprägt wurden. Dies setzt zunächst voraus, dass ein geeignetes Datenmodell zur Verfügung steht, dass sowohl Produkteigenschaften als auch Anforderungen abbilden kann, sowie die Verknüpfungen zwischen ihnen. Um Informationsverluste zu vermeiden, muss das Modell bei Ergänzungen oder anderen Änderungen der Informationsmenge fortgeschrieben werden (Modellintegration). Da die Produkteigenschaften in unterschiedlichen Werkzeugen definiert werden, deren Produktmodelle nicht von vorneherein bekannt sind, muss das Datenmodell so flexibel sein, dass es mit

minimalem Aufwand an neue zu integrierende Modelle angepasst und erweitert werden kann. Zu berücksichtigen ist außerdem, dass Produktmerkmale aus verschiedenen Modelldomänen nicht unbedingt 1:1 aufeinander abgebildet werden können, sondern über komplexe Beziehungen miteinander in Verbindung stehen. Dies gilt beispielsweise bei Produkteigenschaften, die die Geometrie eines Bauteils beschreiben und Produktanforderungen. In heute gebräuchlichen CAD-Systemen, die für die Definition der Produktgeometrie eingesetzt werden, liegen die Informationen auf einem sehr hohen Detaillierungsgrad vor. Anforderungen werden dagegen in der Regel auf einer sehr viel tieferen Detaillierungsebene definiert. Das bedeutet, dass die Geometrieinformationen für eine Evaluierung basierend auf Anforderungen entsprechend aufbereitet werden müssen.

Ein weiterer wichtiger Gesichtspunkt ist die Unterstützung bei der Identifikation der korrespondierenden Produktmerkmale aus unterschiedlichen Modelldomänen. Da es gilt, eine große Menge von Informationseinheiten in Beziehung zueinander zu bringen, ist der Aufwand dafür sehr hoch. Daher muss das Konzept Lösungen für eine effiziente Erkennung und Verknüpfung der korrespondierenden Merkmale bieten.

3.1.2 Abbildung und Erkennung von Abhängigkeiten zwischen Produktmerkmalen

Komplexe technische Produkte integrieren Technologien aus unterschiedlichen Ingenieurswissenschaftlichen Domänen. Die einzelnen Teilbereiche sind dabei eng miteinander verzahnt. Das bedeutet, dass sich Änderungen in einem Teilbereich des Produkts auf die anderen Teilbereiche auswirken können. Beispielsweise können Änderungen an der Elektronik eines Kraftfahrzeugs eine Erhöhung der Stromlast auf einem Kabelbaum hervorrufen, die wiederum Auswirkungen auf die Kabelbaumdurchführungen haben kann, dann nämlich, wenn der Leiterquerschnitt erhöht werden muss. Alle drei betroffenen Bereiche (Elektronik, Elektrik und Karosserie) werden mit Hilfe unterschiedlicher, isolierter Modellierungswerkzeuge entworfen. Das wiederum bedeutet, dass die Änderungen an der Elektronik nur schwer und mit großer Verzögerung weiterpropagiert werden können. Aus diesem Grund muss das Konzept Lösungen für die Abbildung komplexer Zusammenhänge zwischen Produktmerkmalen aus unterschiedlichen Disziplinen und Modellierungswerkzeugen bereitstellen. Auch hier gilt, dass die Informationsmengen, die analysiert und ausgewertet werden müssen, sehr groß sind. Daher müssen im Rahmen des Konzepts zusätzlich Lösungen bereitgestellt werden, die eine effiziente Erkennung von Abhängigkeiten zwischen Produktmerkmalen ermöglichen.

3.1.3 Anforderungen an die Flexibilität der Systemanbindung

Die Vielfalt der im industriellen Einsatz befindlichen Autorensysteme erfordert von einem System zur Nutzdatenintegration und –Verknüpfung eine hohe Flexibilität, sowohl was die Abbildung der Modellinhalte der anzubindenden Werkzeuge angeht, als auch was die von den

Systemen bereitgestellten Möglichkeiten für einen Zugriff auf die Modellinhalte anbelangt. Es muss daher dem Umstand Rechnung getragen werden, dass die unterschiedlichen Systeme verschiedene Ansätze für den Zugriff von außen auf ihre Datenmodellinhalte verfolgen. Man kann prinzipiell unterscheiden zwischen einem Online-Zugriff und einem Offline-Zugriff. Online bedeutet, dass das anzubindende System während des Zugriffs auf sein Datenmodell ebenfalls ausgeführt wird und Informationen direkt vom laufenden System abgefragt werden. Dabei ist es prinzipiell unerheblich, ob es auf demselben Rechner läuft wie das Integrationssystem oder auf einem anderen, über geeignete Netzwerkprotokolle, wie z.B. TCP/IP, angebundenen Rechner. Offline bedeutet, dass das anzubindende System Informationen über Dateien zur Verfügung stellt. Dies können entweder standardisierte Austauschformate, wie beispielsweise STEP, sein oder proprietäre Formate. Wie der Zugriff auf das Datenmodell eines angebundenen Systems erfolgt, muss transparent sein. Das heißt, die Art des Zugriffs darf keine Auswirkungen auf die Implementierung oberhalb der Zugriffsschicht für das jeweilige System haben. Falls jedoch ein Online-Zugriff möglich ist, ist diese Art der Anbindung vorzuziehen, da nur dadurch eine Unterstützung in Echtzeit durch das System möglich ist. Bei einer Offline-Anbindung tritt zusätzlich das Problem der Konsistenzsicherung auf, da nicht garantiert werden kann, dass die jeweils in Bearbeitung befindlichen Modellinhalte aktuell sind.

Da bei der Entwicklung des Systems nicht vorhersehbar ist, wie die Systemlandschaft beschaffen ist, in die das System eingebunden werden soll, ist ein weiterer wichtiger Aspekt dessen Plattformunabhängigkeit.

3.2 Merkmalsgewinnung aus externen Produktmodellierungssystemen

Um eine durchgängige, anforderungsgetriebene Produktentwicklung zu ermöglichen, ist neben einer Unterstützung der Anforderungsaufnahme, eine Beurteilung der modellierten Produkteigenschaften anhand der definierten Anforderungen notwendig. Die Bestimmung einer Produkteigenschaft kann dabei auf unterschiedliche Weise erfolgen. Die Möglichkeiten reichen von physischen Tests an realen Prototypen über rechnerunterstützte Simulationen bis hin zu einfachen Abfragen von explizit modellierten Eigenschaften virtueller Produkte. In diesem Lösungsansatz sollen vor allem Eigenschaften des virtuellen Produkts betrachtet werden, das noch nicht real existiert, sondern abstrakt in rechnerinternen Modellen von Produktmodellierungswerkzeugen abgebildet ist. Da diese Werkzeuge in der Regel keine integrierte Funktionalität zur Erfassung von Anforderungen aufweisen und entsprechend auch keine integrierte Beurteilung der modellierten Produkteigenschaften anhand von Anforderungen zulassen, müssen die Produkteigenschaften einem externen System zur Anforderungsmodellierung und Produkteigenschaftsvalidierung zugänglich gemacht werden, das diese Aufgaben übernimmt.

Um Produkteigenschaften anhand von Anforderungen evaluieren zu können, müssen diese miteinander verknüpft werden. Dabei kann die Kombination mehrerer quantitativer Eigenschaften bzw. der Eigenschaften mehrerer Teilsysteme eine einzelne Anforderung erfüllen. Beispielsweise können mehrere zusammenhängende Kantenlängen in der Geometriebeschreibung eines Bauteils auf eine Anforderung an die Gesamtlänge des Bauteils zeigen.

Da sich Anforderungen in der Regel auf Produkteigenschaften auswirken, die in verschiedenen Modellierungswerkzeugen modelliert bzw. ermittelt werden (z.B. Bauteilgeometrie in CAD-Systemen, Bauteilstabilität in FEM-Systemen), ist es notwendig, ein einziges **S**ystem zur **A**nforderungsbasierten **E**valuierung von **P**rodukteigenschaften zu entwickeln, im folgenden mit *SAEP* abgekürzt, das alle in unterschiedlichen Werkzeugen modellierten Eigenschaften in einem kohärenten Modell erfassen und evaluieren kann (siehe Abbildung 20).

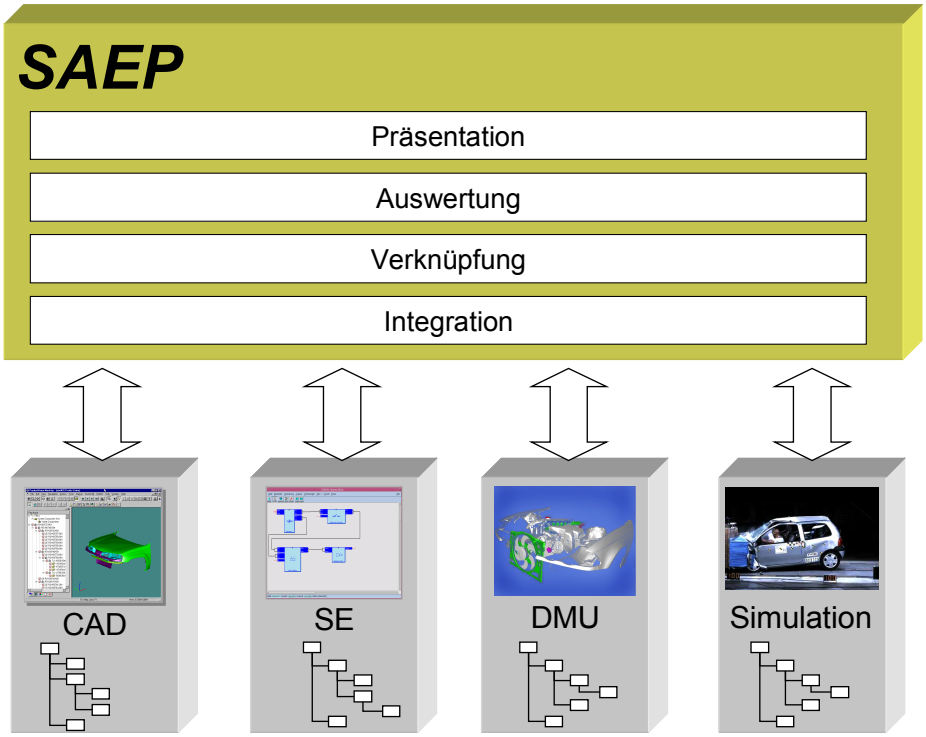


Abbildung 20: SAEP im Umfeld anderer Produktmodellierungswerkzeuge

Der vorgestellte Ansatz zur Merkmalsgewinnung setzt voraus, dass die rechnerinternen Modelle der Modellierungswerkzeuge das abgebildete Produkt als hierarchisches System beschreiben und dass dem Gesamtsystem und seinen Teilsystemen Eigenschaften zugeordnet werden können, die das Produkt beschreiben. Unter welchen Gesichtspunkten die hierarchischen Systeme aufgebaut sind und welche Arten von Eigenschaften zugeordnet werden können, ist für den Ansatz nebensächlich. Eine solche Darstellungsform ist beispielsweise die Produktstruktur in BRep-Modellen. Hier ergibt sich die Hierarchie aus Bestandsbeziehungen (Baugruppen enthalten Unterbaugruppen und Einzelteile), und die Komponenten haben Eigenschaften wie Masse, Volumen etc.

3.2.1 Produktmerkmalsbibliotheken

Um Produktmerkmale, seien es Anforderungen, Eigenschaften oder Sensibilitäten, Produktkomponenten zuzuordnen, müssen sie zunächst definiert werden. Um diesen Vorgang zu vereinfachen, wurde in den Arbeiten von RZEHORZ und GEBAUER das Konzept der Anforderungsbibliotheken eingeführt. Dieser Ansatz wurde in der vorliegenden Arbeit aufgegriffen und erweitert.

Zunächst werden in den Bibliotheken nicht nur Anforderungen verwaltet sondern allgemein Produktmerkmale. Das bedeutet, dass es zunächst unerheblich ist, ob es sich um eine Anforderung, eine Eigenschaft oder eine Sensibilität handelt. Erst im Kontext mit einer konkreten Entwicklungsaufgabe entscheidet der Benutzer, um welche Art von Merkmal es sich handelt. Des Weiteren lassen sich Sichten auf die hierarchische Struktur definieren, in die die Produktmerkmale eingeordnet sind. Dies erlaubt es, die Merkmalsmenge benutzerspezifisch aufzubereiten. Das kann zum einen der Benutzungsfreundlichkeit dienen, indem die Menge sinnvoll eingegrenzt wird, um für den Benutzer uninteressante Aspekte auszublenden, und zum anderen können Außendienstmitarbeiter bei der Anforderungsaufnahme Produktmerkmale verbergen, die der Kunde nicht zu Gesicht bekommen soll.

3.2.2 Sichten auf die Gesamtproduktstruktur

Da in den Modellierungswerkzeugen das zu fertigende Produkt unter unterschiedlichen Gesichtspunkten betrachtet wird (Geometrie, Funktionsstruktur etc.), ist auch die hierarchische Gliederung des Produkts in seine Teilsysteme in den verschiedenen Werkzeugen unterschiedlich. Es ist daher zweckmäßig, in SAEP ebenfalls unterschiedliche Arten der Gliederung der Teilsysteme des Gesamtprodukts zu unterstützen. Diese unterschiedlichen Arten der Gliederung werden als *Sichten auf die Gesamtproduktstruktur* oder kurz: *Sichten* bezeichnet. Die einzelnen Sichten können, müssen aber nicht an die Produktstrukturen in den entsprechenden Werkzeugen angelehnt sein. Es können beliebig viele, beliebig strukturierte Sichten auf das Gesamtprodukt definiert werden. Sichten werden definiert durch die Menge der verbundenen Teilsysteme, sowie den Beziehungen, mit denen die Teilsysteme untereinander verknüpft werden. Auch aus PDM-Systemen sind unterschiedliche Sichten auf das Produkt bekannt. Häufig werden z.B. die Konstruktionssicht („as-designed“) und die Montagesicht („as-build“) unterschieden.

3.2.3 Beziehungen zwischen Produktstrukturkomponenten

Es können verschiedene Beziehungen zwischen den Komponenten einer Produktstruktur bestehen. Die Beziehungen in einer Produktstruktur gehören bestimmten Beziehungsklassen an. Die Beziehungsklassen können prinzipiell frei definiert werden, um möglichst allen unternehmensspezifischen Anforderungen genügen zu können. Es kann generell unterschieden

werden zwischen Beziehungen, die Hierarchien aufbauen und Beziehungen, die nicht-hierarchische Strukturen erzeugen.

Beziehungen, die Hierarchien aufbauen, verbinden immer Teilsysteme unterschiedlicher Hierarchieebenen miteinander. Das bedeutet umgekehrt, dass die Existenz einer Beziehung zwischen zwei Entitäten diese auf verschiedene Hierarchiestufen stellt. Diese Beziehungen sind daher gerichtet und binär. In den meisten Fällen sind die Beziehungsklassen *Aggregation* und *Implementierung* ausreichend:

- Beziehungen der Klasse Aggregation verbinden Entitäten im Sinne von „Ist Teil von", z.B. „Motor ist Teil von Antriebsstrang", „Antriebsstrang ist Teil von Fahrzeug", „Fläche XYZ ist Teil von Volumen XXYYZZ".
- Beziehungen der Klasse Implementierung verbinden Entitäten im Sinne von „Realisiert", z.B. „Fliehkraftregler $X_1Y_1Z_1$ realisiert Funktion Drehzahlregelung" oder „Elektronischer Regler $X_2Y_2Z_2$ realisiert Funktion Drehzahlregelung".

Eine wichtige Beziehungsklasse im Kontext dieses Lösungsansatzes ist die der *Nachbarschaftsbeziehungen* oder *Adjazenzbeziehungen*. Beziehungen dieser Art verknüpfen Komponenten miteinander, die in einem bestimmten Kontext aufeinander Einfluss nehmen können. Adjazenzbeziehungen sind oft implizit in Produktmodellierungswerkzeugen gegeben. Beispiele dafür sind die räumliche Nähe von Komponenten in einem gemeinsamen Raum in CAD- bzw. DMU[24]-Modellen oder die Zugehörigkeit von verschiedenen Teilsystemen zu einem gemeinsamen, übergeordneten System in SE-Modellen. Beziehungen dieser Klasse erzeugen nicht-hierarchische Strukturen. Unter welchen Bedingungen Komponenten noch Adjazenzbeziehungen haben können, wird später behandelt.

3.2.4 Vorgehensweise zur Identifikation und Organisation korrespondierender Produktteilsysteme aus unterschiedlichen Produktmodellierungswerkzeugen

Der Ansatz zur Merkmalsgewinnung beruht auf einer zweistufigen Vorgehensweise: Zunächst müssen korrespondierende Teilsysteme identifiziert und entweder direkt aufeinander abgebildet werden (Mapping), oder in geeignete übergeordnete Systeme eingegliedert werden. Wenn das gelungen ist, können die Eigenschaften der identifizierten Systeme geeignet aufeinander abgebildet werden. Bei der Interaktion von Produktmodellierungswerkzeugen und SAEP sind grundsätzlich drei Vorgehensweisen möglich, mit denen der zweistufige Ansatz zur Merkmalsgewinnung umgesetzt werden kann.

Die Interaktion zwischen den beteiligten Werkzeugen kann dabei entweder ereignisgesteuert oder im Batchmodus (Stapelverarbeitung) erfolgen. Das heißt, jede Operation im steuernden System wird sofort an das gesteuerte System weitergegeben und direkt verarbeitet, oder aber

---

24 In DMU-Systemen (Digital Mock-Up) werden geometrisch beschriebene Produktmodelle virtuell zusammengebaut. Wichtige Aufgaben, die mit Hilfe von DMU-Systemen gelöst werden, sind Kollisionsuntersuchungen in mechanischen Baugruppen sowie Einbauuntersuchungen.

ein ganzer Satz von Operationen wird gesammelt weitergegeben und als Ganzes verarbeitet. Die direkte Kopplung hat den Vorteil, dass Änderungen am Modell sofort erkannt und verarbeitet werden können. Diese Art der Kopplung sollte im Produktentwicklungsprozess bevorzugt eingesetzt werden, da nur sie eine unmittelbare Kontrolle der Produkteigenschaften erlaubt. Sie setzt eine Onlineschnittstelle voraus, d.h. das Produktmodellierungswerkzeug arbeitet parallel zu SAEP, und beide Systeme kommunizieren direkt miteinander. Die Kopplung im Batchmodus ist dann sinnvoll, wenn eine Vielzahl von Modifikationsoperationen ohne Kontrolle des SAEP durchgeführt werden kann. Die Kommunikation über eine Dateischnittstelle wie z.B. das Neutraldateiformat STEP ist ebenfalls als eine Kommunikation im Batchmodus anzusehen.

3.2.4.1 Externes System steuert SAEP

Im externen System wird eine Produktstruktur vordefiniert und das Produkt und seine Komponenten mit Eigenschaften versehen. In SAEP werden daraufhin entsprechende Produktkomponenten mit Eigenschaften erzeugt und diesen entsprechende Anforderungen zugeordnet, die aber noch ausgeprägt werden müssen. Ein Szenario für diesen Fall wäre, dass in einem CAD-System ein Produkt bzw. eine seiner Komponenten bereits modelliert ist. In SAEP wird daraufhin die topologisch-geometrische Struktur nachgebildet und den Eigenschaften der Strukturelemente (z.B. Längen von Kanten) entsprechende Anforderungen zugeordnet. Diese Anforderungen sind in diesem Stadium entweder noch nicht ausgeprägt und müssen erst noch in einem weiteren Schritt „per Hand“ mit Werten versehen werden, oder sie werden mit den Werten der Eigenschaften initialisiert. Im ersten Fall wird davon ausgegangen, dass nicht die für eine konkrete Aufgabe relevanten Anforderungen bei der Konstruktion vorlagen. Dies ist beispielsweise dann der Fall, wenn eine Konstruktionslösung aus einer früheren Aufgabenstellung wieder verwendet werden soll. Die Konstruktion muss also im Nachhinein überprüft werden. Im zweiten Fall wurden bei der Konstruktion für die konkrete Aufgabenstellung relevante Anforderungen berücksichtigt, die aber in einem anderen, nicht von *SAEP* verarbeitbaren Format oder Medium vorliegen. In beiden Fällen muss aber in der Regel aus Gründen, die später noch erläutert werden, davon ausgegangen werden, dass die in SAEP erzeugten Anforderungen noch nachbereitet werden müssen.

3.2.4.2 SAEP steuert externes System

Im SAEP wird die Produktstruktur vordefiniert und das Produkt sowie seine Komponenten mit Anforderungen versehen. Im externen System wird daraufhin eine entsprechende Produktstruktur angelegt und den Anforderungen entsprechende Eigenschaften vordefiniert. Diese Eigenschaften müssen aber im externen System selbst ausgeprägt werden. So kann beispielsweise der Aufwand für die Modellierung von anforderungsgerechten Systemen in System Engineering Werkzeugen verringert werden, indem die Systemkomponenten und deren hierarchische Struktur aus SAEP als Vorlage für die hierarchische Systemstruktur benutzt wird. Für die Systemkomponenten können darüber hinaus den jeweiligen Anforderungen entspre-

chende Ports vordefiniert werden, deren Eigenschaften sich aus dem Verhalten der Komponenten ergibt. Dieses wird weiterhin mit Hilfe des angebundenen System Engineering Werkzeugs definiert und modelliert.

3.2.4.3 Korrespondierende Produktstrukturen in SAEP und externem System

Produktstrukturen in SAEP und in einem externen System existieren bereits und müssen in SAEP integriert dargestellt werden. Hierbei können die Produktkomponenten entweder direkt aufeinander abgebildet werden (Mapping) oder müssen, falls das nicht möglich ist, in einer übergeordneten Komponente zusammengefasst werden.

Die Struktur der übergeordneten Systeme kann unter unterschiedlichen Gesichtspunkten aufgebaut werden. Sinnvoll erscheint z.B. eine Gliederung nach aufgabenbezogenen, funktionalen Aspekten. Beispielsweise kann eine Komponente „Kraftfahrzeugbremssystem“ physische Baugruppen und Bauteile wie das Bremsleitungssystem oder die Bremsscheibe und gleichzeitig elektronische Komponenten wie das ABS-Steuergerät sowie die zugehörige Steuerungssoftware zusammenfassen. Um eine möglichst hohe Flexibilität bei der Integration verschiedener Produktmodellierungswerkzeuge zu erreichen, ist es möglich, verschiedene Gliederungsschemata parallel zu verwalten und durch verschiedene Sichten auf die Systemstruktur zu repräsentieren (siehe Abschnitt 3.2.2).

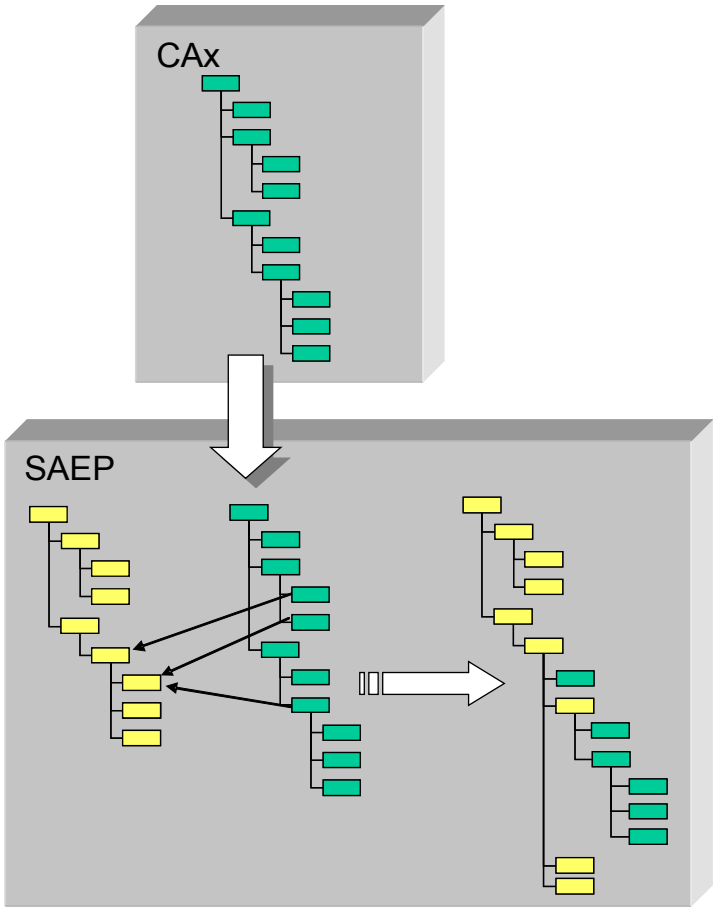


Abbildung 21: SAEP und externes System müssen abgeglichen werden

3.2.5 Vorgehensweise zur Verknüpfung von Produktkomponenten unterschiedlicher Systeme

Wenn Produktkomponenten aus verschiedenen Produktmodellierungswerkzeugen mit ihren untergeordneten Komponenten in SAEP aufeinander abgebildet werden sollen, ist zu lösen, wie die korrespondierenden Komponenten identifiziert und einander zugeordnet werden können. Die geeignete Vorgehensweise hängt dabei von der Art des einzubindenden Produktmodellierungswerkzeugs ab und davon, wie die Komponenten und ihre Eigenschaften verwaltet werden. In CAD-Systemen werden beispielsweise benutzerdefinierte Namen gewöhnlich nur bis auf Bauteilebene unterstützt. Elemente auf einer höheren Detaillierungsstufe, wie Volumina, Flächen, etc., erhalten dagegen durch das System definierte, eindeutige Bezeichnungen (IDs). Das Konzept setzt daher voraus, dass parametrische CAD-Systeme angebunden werden[25]. In diesen lassen sich beliebige Parameter definieren. Dies muss allerdings vom Benutzer manuell durchgeführt werden.

---

25 Diese Voraussetzung ist zulässig, da marktgängige CAD-Systeme die parametrische Modellierung unterstützen

3.2.5.1 Automatische Analyse der Modellinhalte von Produktmodellierungswerkzeugen und SAEP

Falls benutzerdefinierte bzw. einheitliche Namen in den angebundenen Produktmodellierungswerkzeugen verfügbar sind, ist zunächst ein einfacher Textvergleich der Teilsystemnamen mittels *Regulärer Ausdrücke* naheliegend [Frie-2003]. Eine weitere Möglichkeit besteht im Vergleich der Namen der Eigenschaften und der Anforderungen der einzelnen Teilsysteme, ebenfalls mit Hilfe *Regulärer Ausdrücke*. Im industriellen Einsatz von CAD-Systemen gelten häufig standardisierte und unternehmensweit verbindliche Regeln zur Benennung von Bauteilen und Baugruppen, sodass diese Vorgehensweise in diesem Bereich erfolgversprechend ist. Sind die im Produktmodellierungswerkzeug abgebildeten Eigenschaften der Produktkomponenten heterogen, d.h. werden sie mit Hilfe unterschiedlicher Maßeinheiten gemessen, ist es zielführend, die Maßeinheiten der Eigenschaften und Anforderungen zu vergleichen. Diese Vorgehensweise ist bei der Integration von System Engineering Werkzeugen sinnvoll, da dort oft heterogene Größen betrachtet werden, die beispielsweise Energieflüsse, Stoffströme, Drücke, elektrische Spannungen, etc. eines Teilsystems beschreiben und unterschiedliche Maßeinheiten aufweisen. Nicht zielführend ist der Vergleich von Maßeinheiten dagegen bei der Einbindung von CAD-Systemen, in denen weitgehend homogene, geometrische Größen betrachtet werden.

3.2.5.2 Schablonen zur Integration von Modellinhalten von Produktmodellierungswerkzeugen und SAEP

In vielen Branchen beruht ein großer Anteil der Entwicklung von neuen Produkten auf Variationen von bereits existierenden Komponenten. Meist ist nur ein kleiner Anteil eines zu entwickelnden Produkts eine wirkliche Neukonstruktion, bei der auf keine bekannten Komponenten zurückgegriffen werden kann. Das heißt, zwischen vielen Elementen besteht eine Ähnlichkeit. Die Ähnlichkeit der Varianten kann dazu ausgenutzt werden, um Modellinhalte von Produktmodellierungswerkzeugen und SAEP zu verknüpfen, indem die Gemeinsamkeiten einmalig modelliert und nur variantenspezifische Informationen neu erzeugt werden. Dazu wurde das Konzept der *Schablonen* entwickelt

Schablonen bezeichnen abstrakte Vorlagen, die zur Generierung des integrierten Modells benutzt werden. In Schablonen werden Systemstrukturen ähnlicher Komponenten auf abstrakter Ebene vordefiniert. Zum Beispiel ist das Hydrauliksystem einer Fahrzeugbremsanlage oft aus ähnlichen Komponenten aufgebaut, die in ähnlichen Strukturen angeordnet sind. Die Hydraulikanlage besteht beispielsweise in den meisten Fällen aus einem Hauptbremszylinder, der über Hydraulikleitungen mit den Radbremsen verbunden ist. Sind die Radbremsen als Scheibenbremsen ausgeführt, bestehen sie immer aus einer Bremsscheibe, die über hydraulische Aktoren gebremst wird.

Aus einer Schablone können konkrete, integrierte Modelle instanziiert werden. Schablonen sind umso nutzbringender, je konkreter sie definiert werden können und je weniger sie an konkrete Anwendungsfälle angepasst werden müssen, d.h. je ähnlicher sich die abzuleitenden

Varianten sind. Dieser Umstand bestimmt auch die möglichen Einsatzgebiete, bzw. Randbedingungen für eine sinnvolle Anwendung von Schablonen. Produktfamilien, die zumindest eine ähnliche Systemstruktur aufweisen, sind für den Schablonenansatz besonders geeignet, selbst wenn sie sich in ihren Eigenschaften signifikant unterscheiden.

In einer Schablone ist eine Systemstruktur mit ihren Teilsystemen und Komponenten vordefiniert. Auf die Systemstruktur können in der Schablone dabei auch unterschiedliche Sichten vordefiniert sein (siehe 3.2.2). Sichten können zur Generierung von Modellen in externen Produktmodellierungswerkzeugen benutzt werden. Das heißt, es werden, angepasst an die typischen Produktstrukturen in den jeweiligen Modellierungswerkzeugen, Sichten vordefiniert, aus denen provisorische Modellinhalte der Modellierungswerkzeuge erzeugt werden, die im weiteren Verlauf der Produktentwicklung fertig entwickelt werden müssen.

Den Teilsystemen sind Merkmale in unterschiedlichem Abstraktionsgrad zugeordnet. Das bedeutet, ein Merkmal kann z.B. ganz abstrakt eine Anforderung sein, die noch nicht ausgeprägt ist, d.h. deren exakter Wert auf der Abstraktionsebene der Schablone nicht bekannt ist, wohl aber deren generelle Relevanz für die Produktentwicklung. Sie muss dann während der Produktentwicklung mit einem konkreten Wert belegt werden. Ein einfaches Beispiel dafür ist die maximal zulässige Abmessung eines Bauteils. Dieser Wert kann vorher nicht definiert werden, da er von anderen Werten abhängt, die aufgabenspezifisch definiert werden. Eine Anforderung kann aber auch vorher ausgeprägt sein, beispielsweise, wenn es sich um gesetzliche Vorgaben handelt, die von allen Produkten, die von der Schablone abgeleitet werden sollen, erfüllt werden müssen. Beispielsweise ist der maximale Bremspedalweg in einem PKW gesetzlich vorgegeben und muss von allen Bremsanlagen in diesem Bereich eingehalten werden.

Zur Verwaltung von Schablonen werden so genannte *Schablonenbibliotheken* eingeführt. Schablonenbibliotheken enthalten Mengen von vordefinierten Schablonen, die durch zwei Arten von Beziehungen geordnet sind:

- **Abstraktions-/Konkretisierungsbeziehungen**: durch diese Beziehungen werden Hierarchien innerhalb der Schablonenmenge aufgebaut, deren Ebenen durch den Grad der Abstraktion bestimmt sind, wobei der Abstraktionsgrad von oben nach unten abnimmt. Das bedeutet, Schablonen auf weiter unten liegenden Hierarchieebenen sind auf einer konkreteren Ebene als weiter oben liegende. Schablonen lassen sich aus anderen Schablonen nach unten, d.h. in Richtung geringerer Abstraktion in der Hierarchiestruktur herleiten. Zum Beispiel könnte eine Schablone „Faustsattelscheibenbremse“ von einer Schablone „Scheibenbremse“ abgeleitet werden. Das bedeutet, die neue Schablone kopiert die Struktur der Originalschablone und kann anschließend für den ihr bestimmten, konkreteren Anwendungsfall angepasst werden, indem Irrelevantes gestrichen und neue, für den erwarteten Anwendungsfall relevante Informationen hinzugefügt werden.
- **Aggregationsbeziehungen** drücken aus, dass Schablonen Bestandteile anderer Schablonen sind. Diese Hierarchie ist äquivalent zu den Baugruppen/Bauteilehierarchien in

Produktstrukturen. Beispielsweise kann eine Schablone „Bremssystem“ Teil der Schablone „Fahrwerk“ sein.

So wie konkrete, integrierte Produktmodelle von Schablonen abgeleitet werden können, ist es auch möglich, von konkreten, bereits existierenden Modellkonfigurationen und Modellausprägungen Schablonen abzuleiten. Dies ist beispielsweise bei abgeschlossenen Neukonstruktionen sinnvoll, auf deren Basis später Weiterentwicklungen erfolgen sollen. Die Weiterentwicklungen stellen Variationen der Neukonstruktion dar und sind daher einander ähnlich. Somit bietet es sich an, die Gemeinsamkeiten vorzudefinieren, um später lediglich die variantenspezifischen Informationen neu erzeugen zu müssen. Auf diese Weise wird das Wissen über die Verknüpfung der Modellinhalte der unterschiedlichen Modellierungswerkzeuge in wieder verwendbarer Form festgehalten und kann später unmittelbar für ähnliche Problemstellungen wieder benutzt werden. Durch das Wechselspiel von Produktmodellinstanziierung aus Schablonen und der Rückführung von instanziierten Produktmodellkonfigurationen in Schablonen entsteht ein zyklischer Prozess, mit dem die Qualität und der Umfang des Schablonenbestands ständig weiterentwickelt werden kann.

3.2.5.3 Manuelle Integration von Modellinhalten von Produktmodellierungswerkzeugen und SAEP

Da das System zur Integration von Nutzdaten dem Benutzer bei dieser Tätigkeit immer nur assistieren kann, sind ihm flexible und benutzungsfreundliche Schnittstellen für die manuelle Konfiguration und Manipulation bereit zu stellen, um ein korrektes Ergebnis der Integration sicherstellen zu können. Die Benutzungsschnittstelle wird im Folgenden als Graphical User Interface (GUI) bezeichnet. Für die manuelle Integration von Modellinhalten muss der Benutzer über die SAEP-eigene GUI Zugriff auf die Modellinhalte der anzubindenden externen Produktmodellierungswerkzeuge haben und sie in einer geeignet aufbereiteten Form zu einem integrierten Modell konfigurieren können. Darüber hinaus wird Funktionalität für die Definition und Ausprägung von Produktmerkmalen benötigt, um beispielsweise Produktanforderungen zu formulieren und entsprechenden Produkteigenschaften zuzuordnen.

Die manuelle Konfiguration und Manipulation kann von Anfang an notwendig sein, das heißt, die Produktmodellkonfiguration muss ohne intelligente Unterstützung durch SAEP allein vom Anwender durchgeführt werden. Dies ist dann der Fall, wenn es noch keine passenden Schablonen für einen bestimmten Anwendungsfall gibt und die automatische Analyse der externen Produktmodelle keine Ergebnisse zeitigt. Die Funktionalität zur manuellen Konfiguration und Manipulation ist gleichfalls zur Nachbearbeitung und Verfeinerung von aus Schablonen instanziierten, integrierten Produktmodellen notwendig. Die Änderungen können wieder in die ursprüngliche Schablone zurückgeführt werden. Ebenso müssen integrierte Produktmodelle, die auf Grund der automatischen Analyse von Modellinhalten von Produktmodellierungswerkzeugen gewonnen wurden, in der Regel manuell nachbearbeitet werden.

3.2.6 Verknüpfung von Merkmalen

Wie in Abschnitt 3.2 auf Seite 48 bereits beschrieben, ist es notwendig, einzelne quantitative Eigenschaften, die nicht sinnvoll einer Anforderung zugeordnet werden können, zu einer geeigneten *aggregierten Eigenschaft* zusammenzufassen, die einer Anforderung zugeordnet werden kann. Abbildung 22 zeigt dies am Beispiel der Repräsentation der Teillängen einer Welle zu einer Gesamtlänge und deren Zuordnung zu einer Anforderung.

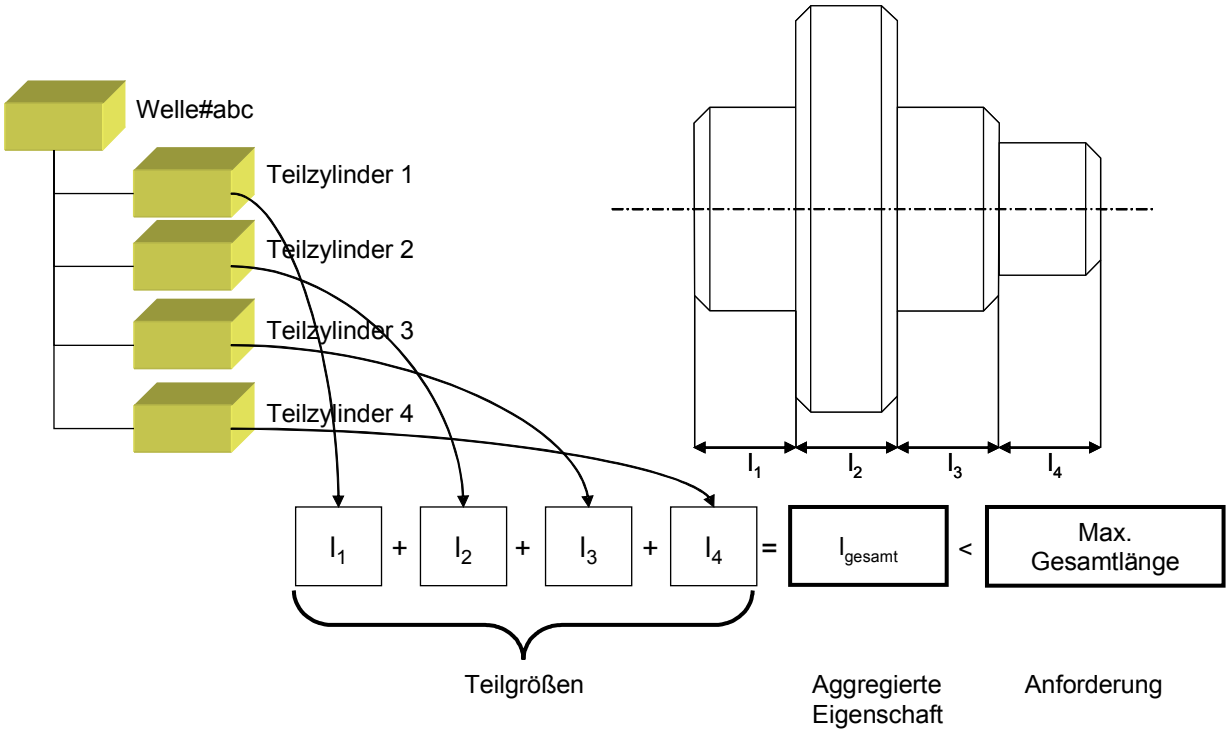


Abbildung 22: Abbildung mehrerer Teilgrößen zu einer Aggregierten Eigenschaft und deren Zuordnung zu einer Anforderung

Modelltechnisch wird dazu ein Binärbaum benutzt, wie er üblicherweise zur Repräsentation von mathematischen Ausdrücken verwendet wird. In den Knoten sind dabei binäre Operatoren abgebildet, während in den Blättern numerische Werte oder Variablen abgelegt sind. Abbildung 23 zeigt die Repräsentation der Gleichung „$l_1 + l_2 + l_3 + l_4 = l_{gesamt}$“ in einem Binärbaum. Bei der Verknüpfung von einzelnen Eigenschaften zu einer aggregierten Eigenschaft werden die zu verknüpfenden Eigenschaften in den Blättern repräsentiert und mit entsprechenden Operatoren verknüpft. Mit diesem Ansatz lassen sich auch komplexe Verknüpfungen abbilden.

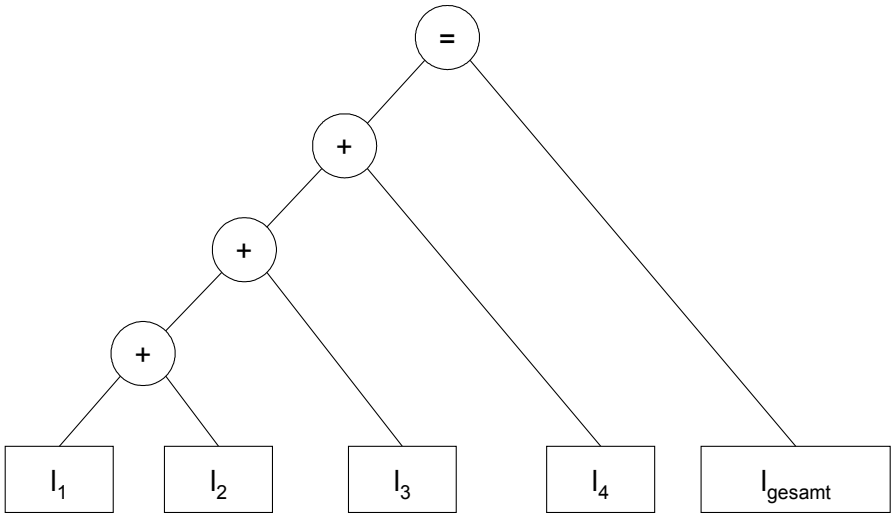


Abbildung 23: Repräsentation der Gleichung „l_1+ l_2+ l_3+ l_4 = l_{gesamt}“ in einem Binärbaum

3.2.6.1 Darstellung und Analyse von Maßeinheiten

Binärbäume eignen sich auch zur Abbildung von Maßeinheiten. Zusammengesetzte Einheiten werden in Einheitenformeln abgebildet. Dabei sind in den Knoten die mathematischen Operatoren abgebildet, während die SI-Einheiten, bzw. andere zusammengesetzte Einheiten in den Blättern abgebildet werden. Abbildung 24 zeigt die Darstellung der Einheit Volt V = m^2 * kg * s^{-2} * C^{-1} als Binärbaum mit SI-Einheiten.

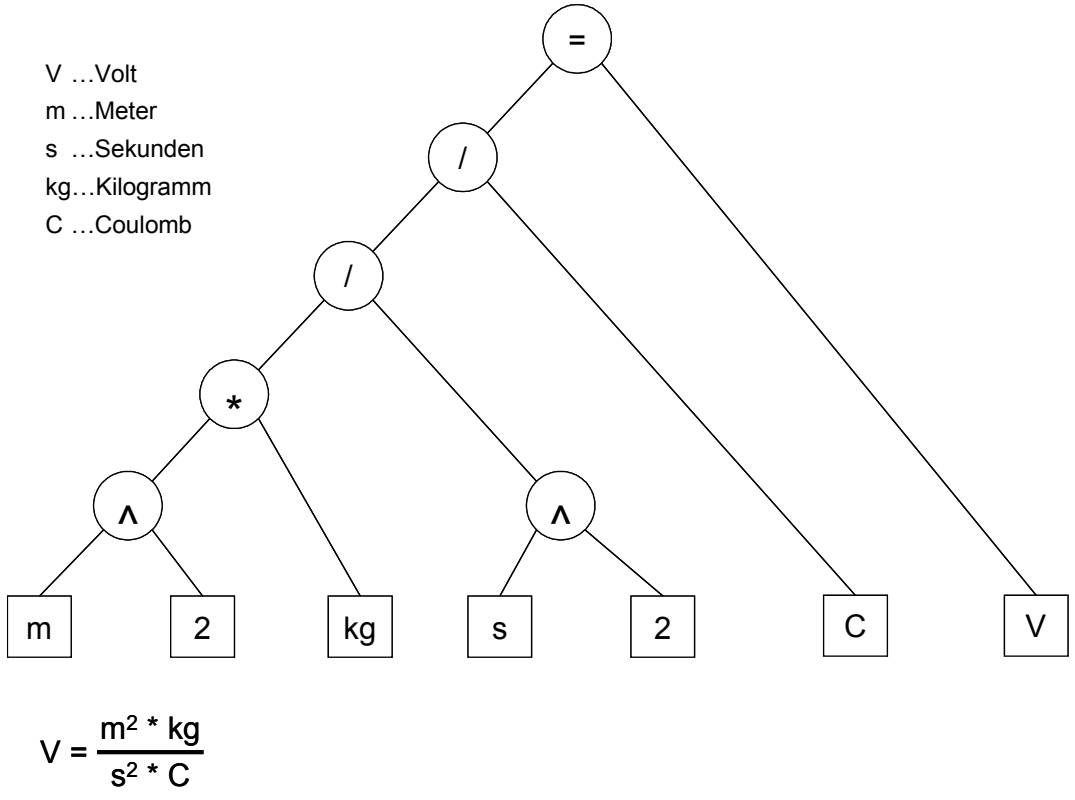


Abbildung 24: Darstellung der Einheit Volt als Binärbaum mit SI-Einheiten

Werden die Einheiten auf ihre Grundform und den Dezimalfaktor heruntergebrochen, ergibt sich der in Abbildung 25 dargestellte Binärbaum mit der Aufspaltung der Einheit kg in k * g.

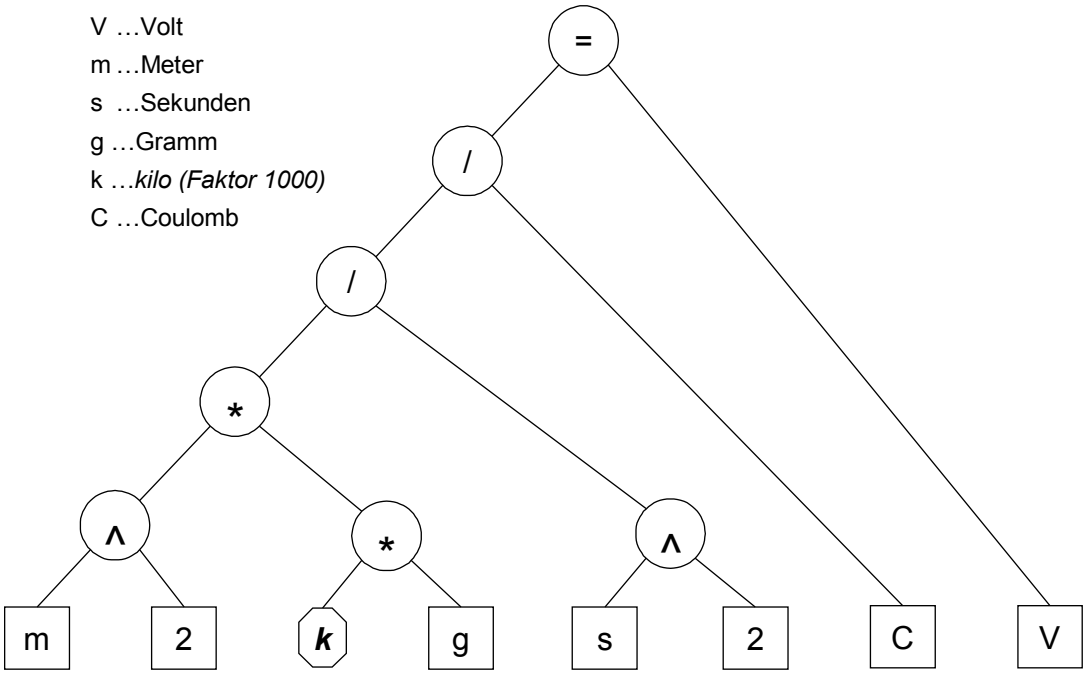


Abbildung 25: Darstellung der Einheit Volt in Grundeinheiten mit Dezimalfaktoren

Sollen Maßeinheiten verschiedener Anforderungen und Eigenschaften miteinander auf Äquivalenz verglichen werden, können verschiedene Grade der Übereinstimmung definiert werden. Eine Übereinstimmung ersten Grades bedeutet, dass die Maßeinheiten identisch sind, d.h. sowohl in der Zusammensetzung ihrer Grundeinheiten als auch in deren Größenordnung übereinstimmen. Eine Übereinstimmung zweiten Grades liegt vor, wenn zwei Maßeinheiten zwar in der Zusammensetzung ihrer Grundeinheiten übereinstimmen, diese aber in verschiedenen Größenordnungen vorliegen, wie z.B. bei den Geschwindigkeitseinheiten km/h und m/s. Der Vergleich der Grade der Übereinstimmung erlaubt eine Einschätzung der Korrektheit der vom System vorgeschlagenen Paarungen von Eigenschaften und Anforderungen und hilft, die Qualität der Systemausgabe in diesem Bereich zu erhöhen.

3.3 Erkennung von Abhängigkeiten zwischen Merkmalen

Die Erkennung, Verfolgung und Auswertung von Abhängigkeiten zwischen Produktmerkmalen ist eine wichtige Unterstützung für alle am Produktentwicklungsprozess beteiligten Personen. Nur dadurch lassen sich die komplexen Zusammenhänge zwischen verschiedenen Produktkomponenten beherrschen und eine ganzheitliche Sicht auf das Produkt herstellen. Die Lösung dieses Sachverhalts ist besonders wichtig für Produkte, die Technologien aus mehre-

ren Disziplinen in sich vereinigen. Typische Vertreter solcher Produkte sind so genannte Mechatronische Produkte[26].

Im vorliegenden Ansatz werden keine lösungsneutralen, also vom eigentlichen Produkt losgelösten, allgemeinen und daher lediglich generell zutreffenden Abhängigkeiten betrachtet, sondern ganz konkret die Abhängigkeiten, die sich individuell innerhalb eines bestimmten Produkts ergeben, behandelt. Das bedeutet, alle Aussagen über Produktmerkmale beziehen sich auf Merkmale, die bestimmten Produktkomponenten zugeordnet sind und nur in dessen Kontext Abhängigkeiten zu anderen, ebenfalls bestimmten Produktkomponenten zugeordneten Merkmalen besitzen. Damit können die Abhängigkeiten zwischen Produktmerkmalen sehr viel genauer abgebildet und ausgewertet werden als mit dem Ansatz zur Abbildung der allgemeingültigen, lösungsneutralen Abhängigkeiten in Anforderungsbibliotheken.

Da die einzelnen Produktmerkmale in Datenmodellen verschiedener Werkzeuge modelliert sein können, ist es möglich, über die Systemgrenzen der unterschiedlichen Produktmodellierungswerkzeuge hinweg Abhängigkeiten auszuwerten. Damit können die Konsequenzen von Änderungen an einer Komponente des Produktmodells unmittelbar erkannt und beurteilt werden. Werden darüber hinaus den einzelnen Produktkomponenten, bzw. deren Eigenschaften, verantwortliche Personen zugeordnet, können diese automatisch durch das System über für sie relevante Änderungen durch andere Stellen informiert werden. In Abbildung 26 ist ein Abhängigkeitsnetz mit zugeordneten Personen schematisch dargestellt.

26 Es existiert bis heute keine allgemein anerkannte Definition des Begriffs *Mechatronisches Produkt*, gleichwohl gibt es, zumindest in Deutschland, ein gemeinsames Verständnis darüber, dass Mechatronik „eine Ingenieurwissenschaft, die die Funktionalität eines technischen Systems durch eine enge Verknüpfung mechanischer, elektronischer und Daten verarbeitender Komponenten erzielt“ ist [Bosc-2003] [NN-2004a]. Diese Definition geht auf die Definition von Harashima, Tomizuka und Fukuda zurück [HaFT-1996], die auch in VDI-Richtlinie 2206 „Entwicklungsmethodik für mechatronische Systeme“ [VDI-2004] referenziert wird.

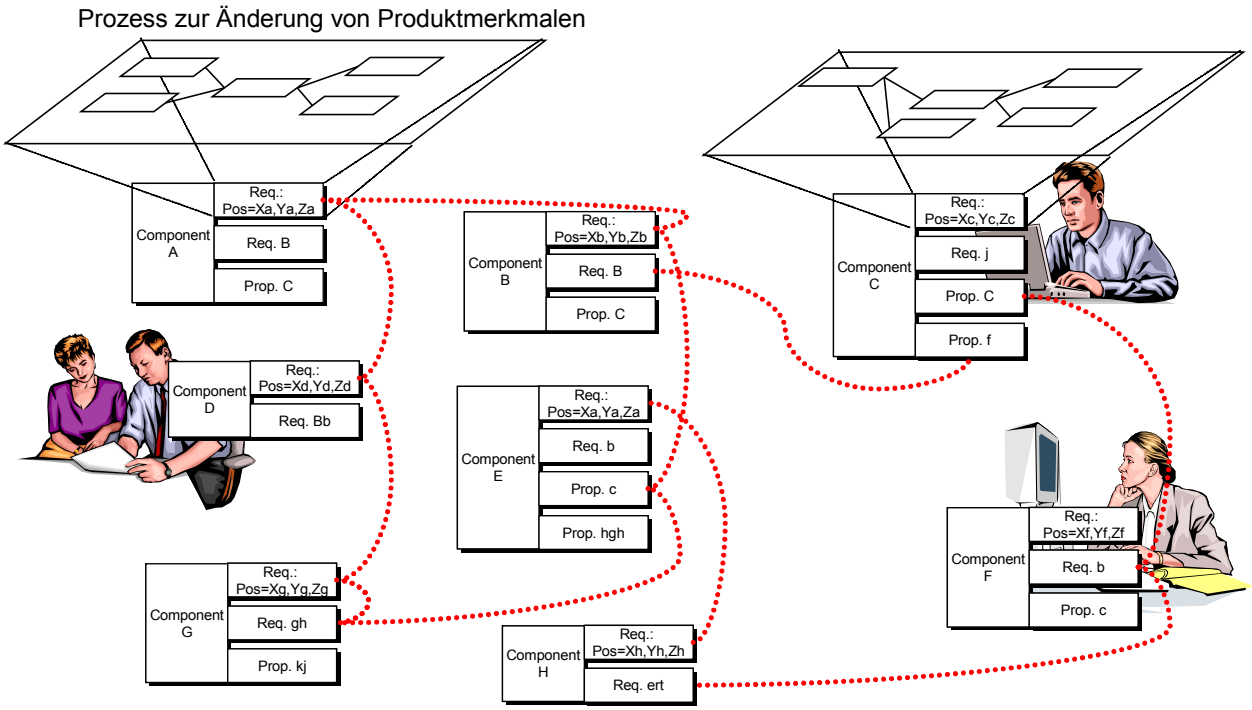


Abbildung 26: Abbildung und Auswertung von Abhängigkeitsnetzen

Abhängigkeitsnetze können darüber hinaus auf bestimmte Kriterien hin untersucht werden:

- **Modularität des Produkts**. Gemäß den in der VDI-Richtlinie 2224 erläuterten Konstruktionsrichtlinien [VDI-1972] sollten Produkte modular aufgebaut sein, d.h. zwischen den einzelnen Komponenten sollen so wenig Wechselwirkungen wie möglich bestehen. Eine Analyse des integrierten Produktmodells hinsichtlich Art und Umfang von Abhängigkeiten zwischen verschiedenen Produktkomponenten in SAEP kann Produktkomponenten ermitteln, die besonders viele Abhängigkeiten zu anderen Produktkomponenten besitzen und eine mangelnde Auslegung des Produkts in diesen Bereichen offenbaren.
- **Erkennung von zyklischen Abhängigkeiten.** Es können zyklische Abhängigkeiten in einer Produktstruktur durch die Rückkopplung von Produkteigenschaften bestehen. In Abbildung 26 ist ein zyklische Abhängigkeit beispielhaft dargestellt, die sich über die Komponenten A, D, G, E, B und A ergibt.

3.3.1 Abbildung von Abhängigkeiten

Abhängigkeiten werden in dem vorliegenden Ansatz zwischen Produktmerkmalen definiert, die einer Produktkomponente zugeordnet sind. Abhängigkeiten können allgemeingültig sein und immer zwischen den Attributen verschiedener Komponenten bestehen, oder aber nur unter bestimmten Bedingungen, die durch das Umfeld vorgegeben sind. So kann beispielsweise eine Abhängigkeit zwischen einer Stromquelle, beispielsweise einem Generator und einem Verbraucher bestehen, die die Ausgangsspannung der Stromquelle mit der Eingangsspannung

des Verbrauchers in Beziehung bringt. Diese Abhängigkeit ist allgemeingültig. Zusätzlich kann ein weiteres Exemplar des gleichen Verbrauchers an einer anderen Stelle im Produkt enthalten sein und dort weitere Abhängigkeiten zu anderen Komponenten in seiner Umgebung besitzen, die aber nur in dieser speziellen Umgebung relevant sind

3.3.2 Arten von Abhängigkeiten

Das Wissen über Abhängigkeiten zwischen Produktmerkmalen kann in verschiedener Art vorliegen. Wenn nur bekannt ist, dass ein Merkmal auf irgendeine Weise ein anderes negativ oder positiv beeinflusst, wird von einer *Abstrakten Abhängigkeit* zwischen den Merkmalen gesprochen. Ist dagegen ein mathematischer Zusammenhang bekannt, der beschreibt, wie sich zwei oder mehrere Produktmerkmale gegenseitig beeinflussen, können sogenannte *Mathematische Abhängigkeiten* formuliert werden.

3.3.2.1 Abstrakte Abhängigkeiten zwischen Produktmerkmalen

In Anlehnung an den Ansatz von GEBAUER [Geba-2001] können Merkmale auf verschiedene Weise miteinander in Beziehung stehen und sich gegenseitig beeinflussen. Im Unterschied zum Ansatz von GEBAUER, in dem nur Abhängigkeiten betrachtet werden, die generell zwischen bestimmten Anforderungen gültig sind, werden hier Abhängigkeiten zwischen Merkmalen betrachtet, die einer Produktkomponente zugeordnet sind, und diese Abhängigkeiten damit nur in diesem Zusammenhang Gültigkeit haben. Abhängigkeiten sind hierbei binäre Beziehungen, die nicht gerichtet sind. Das heißt, die Merkmale beeinflussen sich gegenseitig. Es werden verschiedene Arten von Abstrakten Abhängigkeiten unterschieden:

- *Konkurrierende Abhängigkeit*: Diese Art der Abhängigkeit besagt allgemein, dass zwei Produkteigenschaften nicht zugleich optimiert werden können. Bei *Konkurrierenden Abhängigkeiten* zwischen quantitativen Merkmalen ist entscheidend, dass bei beiden verknüpften Merkmalen die Optimierungsrichtung angegeben ist, da ansonsten keine Aussage über die gegenseitige Beeinflussung getroffen werden kann. Demnach stehen zwei Produktmerkmale in konkurrierender Abhängigkeit, wenn die Verbesserung einer Eigenschaft gemäß der Optimierungsrichtung ihrer zugeordneten Anforderung eine Verschlechterung der anderen Eigenschaft gemäß der Optimierungsrichtung ihrer zugeordneten Anforderung bewirkt. Beispielsweise kann der Kraftstoffverbrauch eines Kraftfahrzeugs nicht so gering wie möglich (Optimierungsrichtung gegen 0) und gleichzeitig die Motorleistung so hoch wie möglich (Optimierungsrichtung gegen unendlich) realisiert werden.
- *Ausschließende Abhängigkeit*: Zwei Produktmerkmale stehen in *Ausschließender Abhängigkeit* zueinander, wenn die Erfüllung einer Anforderung die Erfüllung einer anderen Anforderung unmöglich macht.

- *Unterstützende Abhängigkeit*: Zwei Produktmerkmale stehen in *Unterstützender Abhängigkeit* untereinander, wenn die Optimierung einer Eigenschaft bezüglich ihrer Anforderung zugleich auch die Optimierung einer anderen Eigenschaft bezüglich ihrer Anforderung bewirkt. Bei quantitativen Produktmerkmalen ist dies der Fall, wenn die Verbesserung einer Eigenschaft gemäß der Optimierungsrichtung ihrer zugeordneten Anforderung ebenfalls eine Verbesserung der anderen Eigenschaft gemäß der Optimierungsrichtung ihrer zugeordneten Anforderung bewirkt. Auch hier gilt, dass diese Beziehung nur sinnvoll ist, wenn bei den beiden verknüpften Anforderungen jeweils eine Optimierungsrichtung mit angegeben wird.

3.3.2.2 Mathematisch beschreibbare Abhängigkeiten zwischen Produktmerkmalen (Mathematische Abhängigkeiten)

Mathematische Abhängigkeiten werden durch Gleichungen der Form $f(x_1,...x_n) = y$ realisiert, wobei $x_1,...x_n$, und y sowohl Ausprägungen von quantitativen Produktmerkmalen sein können, als auch beliebige numerische Konstanten. Mathematische Abhängigkeiten verknüpfen Produktmerkmale in beliebig komplexer Form, was sie grundsätzlich von *Abstrakten Abhängigkeiten*, die jeweils nur zwei Produktmerkmale miteinander verknüpfen, unterscheidet.

Mathematische Abhängigkeiten können in verschiedener Ausprägung vorliegen. Sie können in geschlossener algebraischer Form darstellbar sein, über Näherungsfunktionen oder in der Form von Wertetabellen. Die verschiedenen Arten mathematischer Repräsentationen bedingen unterschiedliche, rechnerinterne Repräsentationen. Abhängigkeiten in geschlossener, algebraischer Form sowie Näherungsfunktionen können, ähnlich wie Maßeinheiten und aggregierte Eigenschaften, mit Hilfe von Binärbäumen dargestellt werden. In Abbildung 27 ist als Beispiel eine vereinfachte Gleichung zur Bestimmung der Wärmestrahlung an einem Punkt P in einem Kartesischen Koordinatensystem mit einem punktförmigen Wärmestrahler dargestellt (Anmerkung: die Maßeinheiten wurden für eine bessere Übersichtlichkeit der Grafik nicht mit dargestellt).

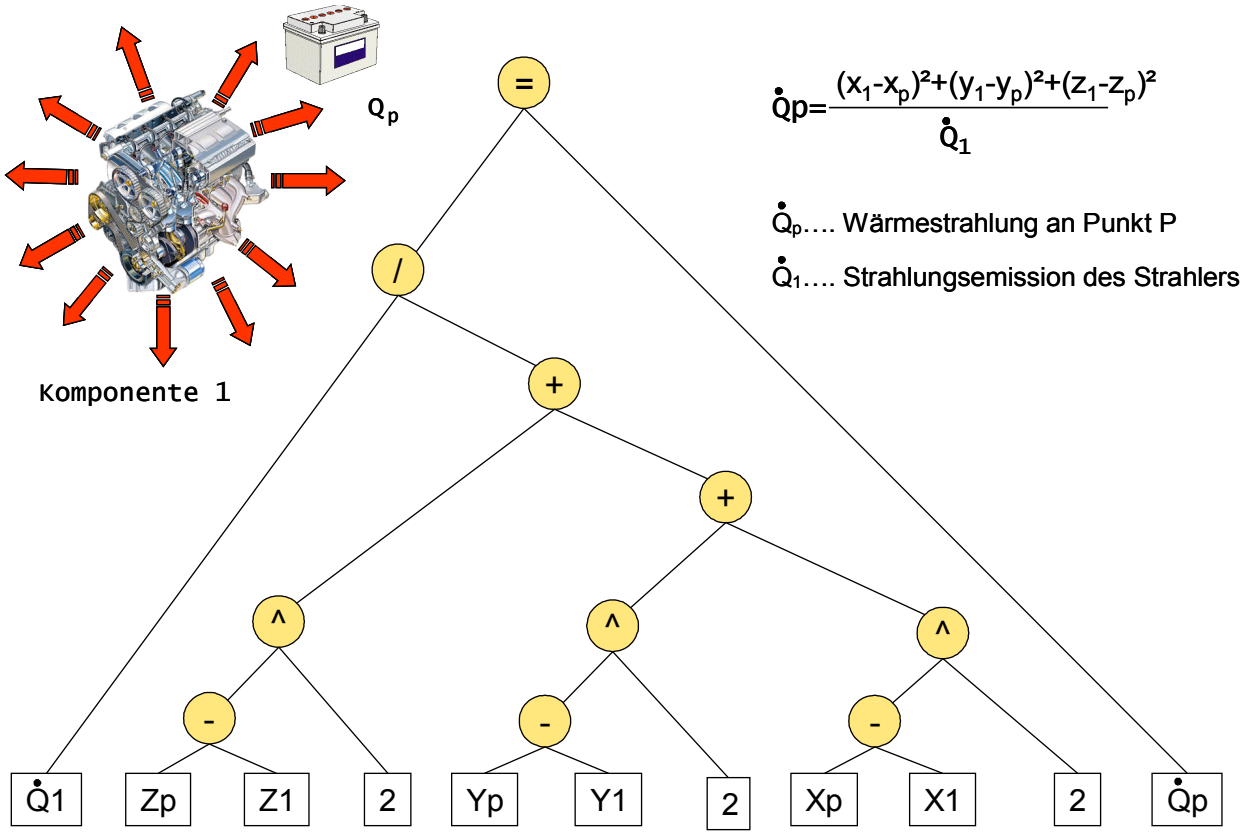


Abbildung 27 : Darstellung des Abstandes zweier paralleler Flächen als Binärbaum

3.3.3 Vorgehensweise zur Ermittlung von Abhängigkeiten zwischen Produktmerkmalen

Bei der Ermittlung von Abhängigkeiten zwischen Produktmerkmalen, die in den Produktmodellierungswerkzeugen definiert werden, kann man unterscheiden in die Ermittlung von Abhängigkeiten zwischen Produktmerkmalen, die im selben System definiert sind und, ergänzend dazu, in die Ermittlung von Abhängigkeiten, die zwischen Produktmerkmalen aus unterschiedlichen Systemen existieren. Der hier vorgestellte Ansatz hat zum Ziel, Abhängigkeiten systemübergreifend zu definieren, zu ermitteln und auswerten zu können.

3.3.3.1 Manueller Ansatz

Da die jeweiligen Modellinhalte der einzelnen Produktmodellierungswerkzeuge in einem integrierten System zugänglich sind (SAEP), besteht die Möglichkeit, über entsprechende Benutzungsschnittstellen, Abhängigkeiten zwischen Produktmerkmalen manuell zu definieren. Das bedeutet, dass der Benutzer ihm bekannte Zusammenhänge zwischen den Merkmalen definiert und diese im System dauerhaft für eine anschließende Auswertung gespeichert werden. Dieser Vorgang kann für komplexe Produkte sehr aufwendig sein. Daher werden Methoden und entsprechende Werkzeuge entwickelt, die den Vorgang der Erkennung

von Abhängigkeiten zwischen Produktmerkmalen und deren Verknüpfung in einem Datenmodell weitergehend unterstützen.

3.3.4 Abbildung von Abhängigkeiten in Schablonen

Der Ansatz der Produktschablonen wird erweitert um die Repräsentation von vordefinierten Abhängigkeiten zwischen den Produktmerkmalen. In den Produktschablonen werden vordefinierte Produktstrukturen abgebildet sowie, dem jeweiligen Abstraktionsgrad entsprechend, den Komponenten der Produktstruktur zugeordnete Produktmerkmale. Zusätzlich kann Wissen über Abhängigkeiten zwischen den Produktmerkmalen in Form von vordefinierten abstrakten oder mathematischen Abhängigkeiten zwischen den Produktmerkmalen in der Schablone abgelegt werden.

Zusammen mit der manuellen Vorgehensweise zur Definition von Abhängigkeiten kann eine zielführende Vorgehensweise definiert werden, die den Prozess zur Ermittlung von Abhängigkeiten entscheidend beschleunigt:

- Sind noch keine Schablonen vorhanden, müssen Produktstrukturen mit dem manuellen Ansatz erstellt werden. Das heißt, die Produktstruktur muss aufgebaut werden, die Strukturkomponenten müssen mit ihren jeweiligen Produktmerkmalen versehen und diese wiederum über die gültigen Abhängigkeiten verknüpft werden. Die so erstellten Produktstrukturen können entweder direkt benutzt werden oder als initiale Schablonen in einen neuen Bestand übernommen werden. Ist beispielsweise noch keine Schablone für die Komponenten eines Antiblockiersystems (ABS) einer Bremsanlage vorhanden, so muss eine geeignet abstrakte Schablone erstellt werden. So kann man als konstante Komponenten eines ABS den Controller, einen Leistungsverstärker, Raddrehzahlsensoren, Ein- und Auslassventile etc. definieren. Abhängigkeiten in einer solchen Konfiguration bestehen beispielsweise immer zwischen dem Strombedarf der Ventile und dem maximalen Schaltstrom des Leistungsverstärkers. Diese Abhängigkeiten können als Bestandteile der Schablone bereits vordefiniert werden.
- Sind Schablonen bereits im System vorhanden, kann der Benutzer eine geeignete Schablone auswählen und diese für die Instanziierung seines zu entwickelnden Produkts benutzen. Mit der Instanziierung aus einer Schablone sind die Komponenten der Produktstruktur bereits mit allen im System bekannten Produktmerkmalen ausgestattet sowie diese mit den entsprechenden Abhängigkeiten untereinander verknüpft. Ist z.B. die Schablone eines ABS vorhanden, so kann diese für die Instanziierung herangezogen werden. Aus den Schablonenobjekten werden konkrete Produktstrukturelemente erzeugt, deren Eigenschaften, soweit noch nicht in der Schablone vordefiniert, durch den Benutzer ausgeprägt werden. Die Abhängigkeit zwischen der Stromaufnahme der Ventile und des Schaltstroms des Leistungsverstärkers ist dann bereits definiert.
- Die instanziierte Produktstruktur wird nun weiter ausgearbeitet, das heißt, neue Produktmerkmale werden gegebenenfalls eingefügt, irrelevante entfernt sowie neu erworbenes Wissen über Abhängigkeiten in das instanziierte Modell eingefügt. Werden bei-

spielsweise die in der Schablone separat ausgeführten Ein- und Auslassventile in der neuen Variante zu Einheiten zusammengefasst, können diese an Stelle der alten Komponenten in die Produktstruktur eingefügt werden.

- Ist das neu erworbene Wissen, das in dem instanziierten Modell eingefügt wurde, für alle Produkte auf der Abstraktionsebene der benutzten Schablone relevant, kann dieses Wissen aus dem instanziierten Produkt in die Schablone zurückgeführt werden und ist für zukünftige Produktentwicklungsprozesse im Gültigkeitsbereich der Schablone verfügbar. Hat sich z.B. die Zusammenfassung der Ein- und Auslassventile zu Einheiten bewährt und soll auch in Zukunft so ausgeführt werden, wird aus der konkreten Produktstruktur eine Schablone erzeugt, die für zukünftige Varianten des ABS herangezogen werden kann. Das neu erworbene Wissen kann aber auch in andere Schablonen überführt werden, indem die Abstraktionsbeziehungen zwischen den Schablonen benutzt werden. Das bedeutet, Schablonen, die in dem Pfad der Abstraktionshierarchie über oder unter der verwendeten Schablone liegen, können mit dem neuen Wissen erweitert werden.

3.3.5 Automatische Erkennung von Mathematischen Abhängigkeiten

Die Erstellung von Schablonen ist, wie die manuelle Vorgehensweise zur Definition von Abhängigkeiten in Produktstrukturen, eine aufwendige Tätigkeit und daher besonders für Varianten- und Anpassungskonstruktionen sinnvoll. Dies zeigen auch Erfahrungen mit parametrischen CAD-Systemen, in denen ähnliche Vorgehensweisen zur Erstellung von Constraintnetzen bekannt sind. Daher ist es erforderlich, die Erkennung von Abhängigkeiten so weit wie möglich zu automatisieren und damit die Potentiale dieser Methodik auch für Neukonstruktionen zu erschließen. Im folgenden wird eine Vorgehensweise vorgeschlagen, mit der die Erkennung von Abhängigkeiten in rechnerinternen Produktmodellen modellübergreifend unterstützt werden kann.

Implizit vorhandene mathematische Abhängigkeiten können durch geeignet hinterlegtes Wissen über globale Zusammenhänge in einem integrierten Produktmodell, das alle produktbeschreibenden Informationen explizit bereitstellt, hergeleitet werden.

Der Ansatz zur Ermittlung von Abhängigkeiten beruht auf der Abbildung von Umgebungsbedingungen in einem Modellraum und wird am Beispiel physikalischer Gesetze im dreidimensionalen, geometrischen Raum erläutert.

Der Ansatz baut auf vier Voraussetzungen auf:

- **Das Wissen über die generelle Möglichkeit gegenseitiger Beeinflussung von Komponenten** wird bereitgestellt durch *Nachbarschaftsbeziehungen* zwischen Produktkomponenten. Nachbarschaftsbeziehungen verknüpfen Komponenten, die in irgendeiner Art in Nachbarschaft zueinander stehen, d.h. sich in einer gemeinsamen Umgebung befinden. Diese Nachbarschaft kann räumlicher Natur sein, wie das implizit z.B. in CAD- bzw. in DMU-Systemen über die Positionen und Lagen der

Komponenten in einem gemeinsamen Raum gegeben ist (siehe Abbildung 28). Nachbarschaftsbeziehungen können aber auch in abstrakterer Form gegeben sein, beispielsweise in System-Engineering-Werkzeugen bei Teilsystemen, die dem gleichen, übergeordneten System zugeordnet sind. Dahinter steht die Annahme, dass Nachbarschaftsbeziehungen die Komponenten identifizieren, die sich potentiell gegenseitig auch über entsprechende Effekte beeinflussen können, die nicht über explizit definierte Abhängigkeiten vorgegeben sind.

- **Das Wissen über die Eigenschaften von Produktkomponenten** beschreibt, ob und wie eine Komponente ihre Umgebung und damit andere Komponenten in ihrer Umgebung beeinflussen kann.
- **Das Wissen über die Bedingungen für die Funktionsfähigkeit einer Komponente** beschreibt, wie Umwelteinflüsse, die durch die Eigenschaften anderer Komponenten bewirkt werden, die Funktionsfähigkeit einer Komponente beeinflussen können. Haben diese Bedingungen Entsprechungen in den Eigenschaften anderer Komponenten in der Umgebung, so sind diese Komponenten potentielle Partner für implizit gegebene Abhängigkeiten.
- **Das Wissen über relevante, physikalische Zusammenhänge im Kontext der betrachteten Komponenten** beschreibt, wie die Eigenschaften einer Komponente andere Komponenten beeinflussen, d.h., wie Eigenschaften auf Bedingungen wirken können.

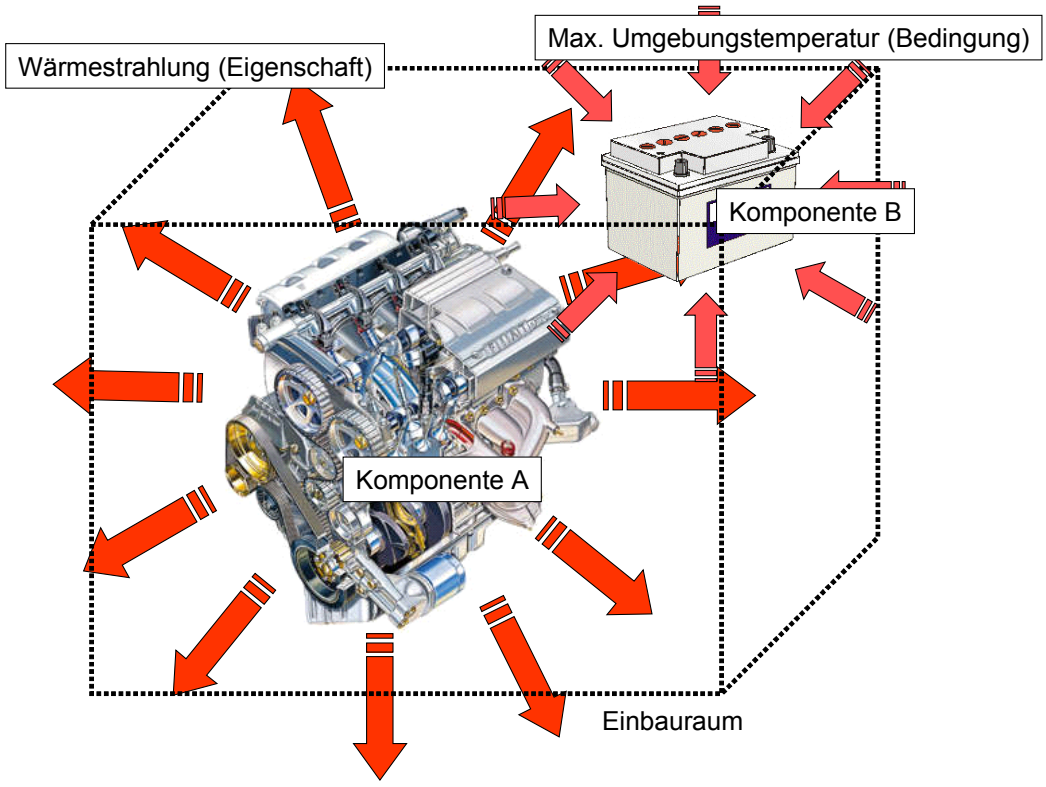


Abbildung 28: Motorblock und Batterie beeinflussen sich gegenseitig in ihrem Einbauraum (Quelle Motorblocksymbol: FIAT)

3.3.5.1 Wirkräume und Kontaktsysteme

Nachbarschaftsbeziehungen sind zusammengefasst in so genannten *Wirkräumen*. Wirkräume stellen damit Systeme von Komponenten dar, die sich potentiell gegenseitig beeinflussen können. Das Wissen über globale physikalische Zusammenhänge, über die sich Produktkomponenten in einer gemeinsamen Umgebung gegenseitig beeinflussen können, ist in so genannten *Kontaktsystemen* gegeben. Kontaktsysteme beschreiben auf abstrakter, mathematischer Ebene, wie sich bestimmte Eigenschaften von Komponenten auf entsprechende Bedingungen der selben oder anderer Komponenten auswirken können. Kontaktsysteme können hierbei lokal begrenzt sein, wie beispielsweise Wärmeleitung über eine gemeinsame Berührungsstelle zweier Komponenten oder aber global wirken, wie z.B. Wärmestrahlung in einem Raum.

Ein Kontaktsystem wird definiert durch einen mathematischen Term, der in Form eines Binärbaums dargestellt wird. Ein Kontaktsystem beschreibt, wie die Eigenschaften der im Wirkraum vorhandenen Komponenten eine globale Eigenschaft des Wirkraums beeinflussen. Die globalen Eigenschaften des Wirkraums beeinflussen die in ihm vorhandenen Komponenten wiederum über deren Bedingungen zur Funktionsfähigkeit. Kontaktsysteme sind Wirkräumen zugeordnet. Damit beschreibt ein Wirkraum ein System aus Komponenten, die sich gegenseitig beeinflussen können, sowie die physikalischen Zusammenhänge, durch die sie sich beeinflussen können (siehe Abbildung 29). Die Komponenten eines Wirkraums sind Modellentitäten des integrierten Modells von SAEP. Das bedeutet, dass Wirkräume, analog zu den verschiedenen Arten von Abhängigkeiten zwischen Produktmerkmalen, Zusammenhänge über die Grenzen von Produktmodellierungswerkzeugen hinweg abbilden können. Der Unterschied zu den Abhängigkeiten zwischen Merkmalen liegt darin, dass die Abhängigkeiten auf Komponentenebene und nicht auf Merkmalsebene abgebildet sind.

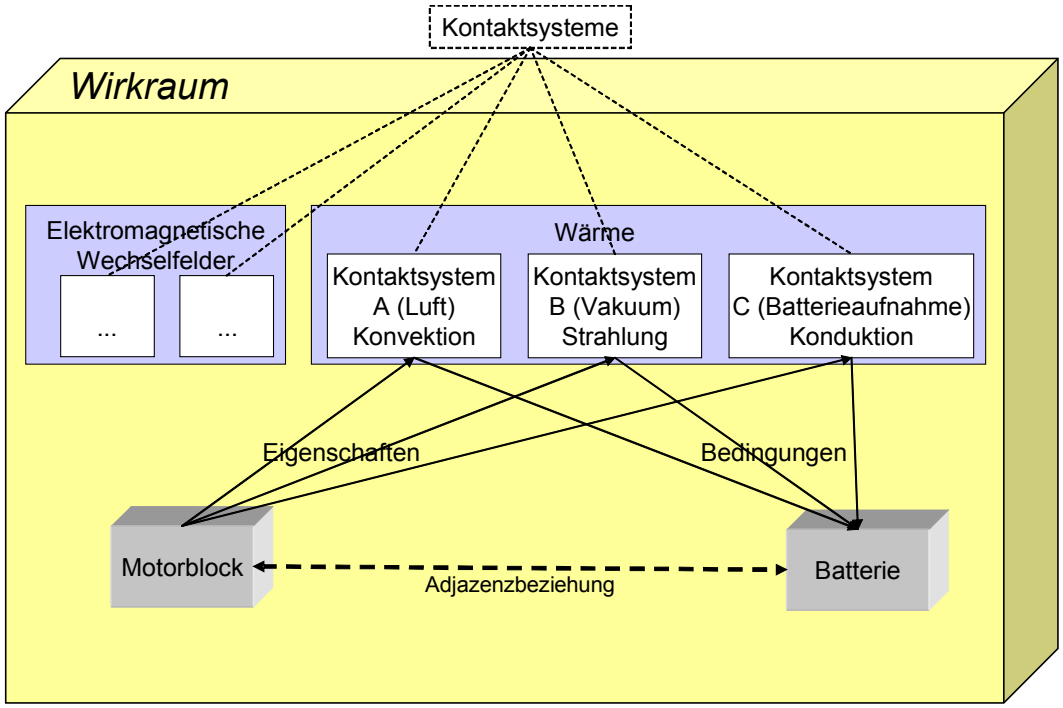
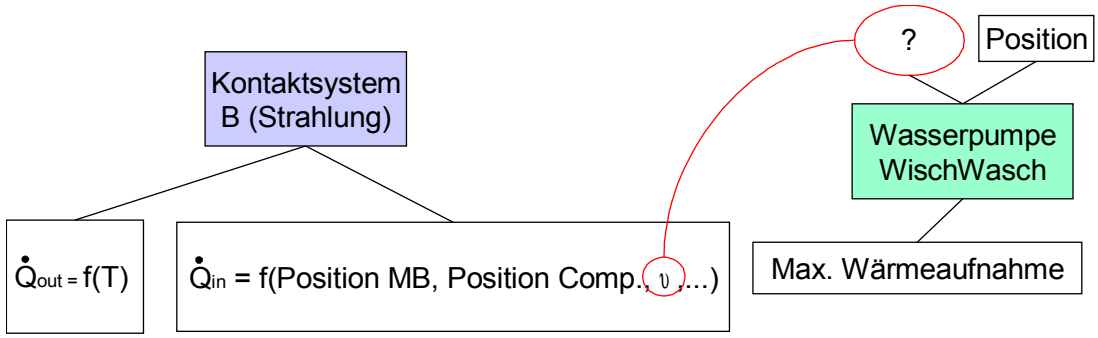


Abbildung 29: Wirkraum mit benachbarten Produktkomponenten und Kontaktsystemen

Die Adjazenzbeziehungen der Wirkräume können rechnerunterstützt aus den eingebundenen Produktmodellierungswerkzeugen geschlossen werden. Beispielsweise können Produktkomponenten, die sich in DMU-Systemen im gleichen geometrischen Raum befinden oder sich berühren, als benachbart angesehen werden und daher mit Adjazenzbeziehungen verknüpft werden. Ebenso verhält es sich mit Teilsystemen, die in System-Engineering-Werkzeugen dem gleichen Obersystem angehören oder direkt mit den externen Schnittstellen des Obersystems verbunden sind.

Die Menge der Merkmale, d.h. der Eigenschaften und Bedingungen von in ausgeprägten Wirkräumen eingebundenen Produktkomponenten können auf ihre Vollständigkeit hin überprüft werden, indem die in den Kontaktsystemen angegebenen Merkmale mit den in den Produktkomponenten definierten Merkmalen verglichen werden. Kann der Platzhalter eines Merkmals in einem Kontaktsystem nicht mit einem passenden Merkmal einer Produktkomponente besetzt werden, ist dies ein Indiz dafür, dass für eine vollständige Produktdefinition notwendige Informationen fehlen. In Abbildung 30 ist als Beispiel ein Kontaktsystem dargestellt, das ein Wärmestrahlungsfeld beschreibt. Um die Erwärmung zu bestimmen, muss der Wärmeübergangskoeffizient υ des bestrahlten Bauteils bekannt sein. Dieser ist in der Kontaktsystemgleichung aufgeführt, fehlt aber in der Definition der Eigenschaften der Baugruppe „Wasserpumpe“.



υ ... Wärmeübergangskoeffizient, $\dot{Q}_{out}$ … ausgehender Wärmestrom, $\dot{Q}_{in}$ … eingehender Wärmestrom

Abbildung 30: Ermittlung von fehlenden Informationen aus ausgeprägten Wirkräumen.

Wirkräume können über Schnittstellen mit anderen Wirkräumen in Verbindung stehen. Über Schnittstellen können prinzipiell beliebige Größen übertragen werden. Schnittstellen sind gerichtet. Das bedeutet, nicht jede Größe kann bidirektional übertragen werden. Ein Wirkraum kann einen anderen Wirkraum nur über die Größen beeinflussen, die in seinen Kontaktsystemen als Eigenschaften abgelegt sind. Äquivalent dazu kann ein Wirkraum nur dann von einem anderen Wirkraum beeinflusst werden, wenn entsprechende Größen als Bedingungen in seinen Kontaktsystemen abgelegt sind. In den Schnittstellen sind die Übertragungsfunktionen für die einzelnen Größen sowohl nach innen als auch nach außen abgelegt. Das heißt, eine Eigenschaft kann nur über eine entsprechende Übertragungsfunktion nach außen wirken. Ebenso kann eine Bedingung nur über eine entsprechende Übertragungsfunktion von außen beeinflusst werden. Abbildung 31 zeigt schematisch, wie Eigenschaften über die Übertragungsfunktionen der Schnittstellen Bedingungen anderer Wirkräume beeinflussen können.

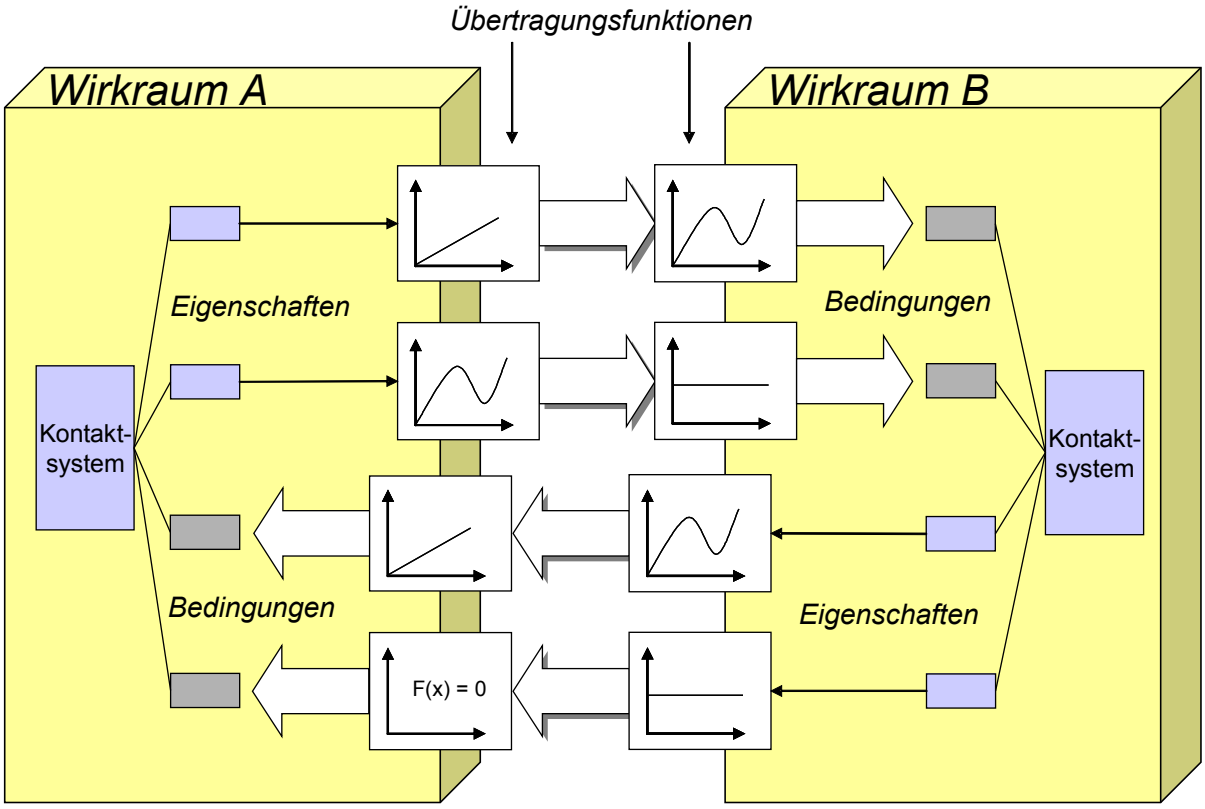


Abbildung 31: Wirkräume beeinflussen sich gegenseitig über Schnittstellen

Ein Beispiel für die Interaktion von Eigenschaften und Bedingungen über Übertragungsfunktionen ist der Wärmeübergang vom im Betriebszustand heißen Wirkraum „Motorraum“ zum benachbarten Wirkraum „Armaturentafel“, in dem Komponenten wie z.B. Airbags enthalten sind. Der Wärmeübergang vom Motorraum zur Armaturentafel erfolgt über eine Schnittstelle, die durch zwei Übertragungsfunktionen gegeben ist, nämlich einer, die die Wärmeabgabe des Motorraums beschreibt und einer, die die Wärmeaufnahme der Armaturentafel beschreibt. Ein weiteres Beispiel ist eine Variation der oben beschriebenen Anordnung von Motorblock und Batterie im selben Wirkraum: Motorblock und Batterie können auch als separate Wirkräume definiert werden. Das bedeutet, dass der Wärmeübergang vom Motorblock auf die Batterie durch entsprechende Übertragungsfunktionen der Wärmeemission (des Motorblocks) und der Wärmeimission (der Batterie) beschrieben wird.

Übertragungsfunktionen definieren, wie ein Eingangswert in einen äquivalenten Ausgangswert überführt wird. Dabei ist die Ausgangsgröße der Quellfunktion Eingangsgröße (Ordinate) der Zielfunktion. Übertragungsfunktionen können aber auch mehrdimensional sein, beispielsweise, wenn die Dauer der Einwirkung der Quellgröße eine Rolle spielt. Abbildung 32 zeigt die Kopplung von Ausgangsfunktion (Quell-Übertragungsfunktion) und Eingangsfunktion (Ziel-Übertragungsfunktion).

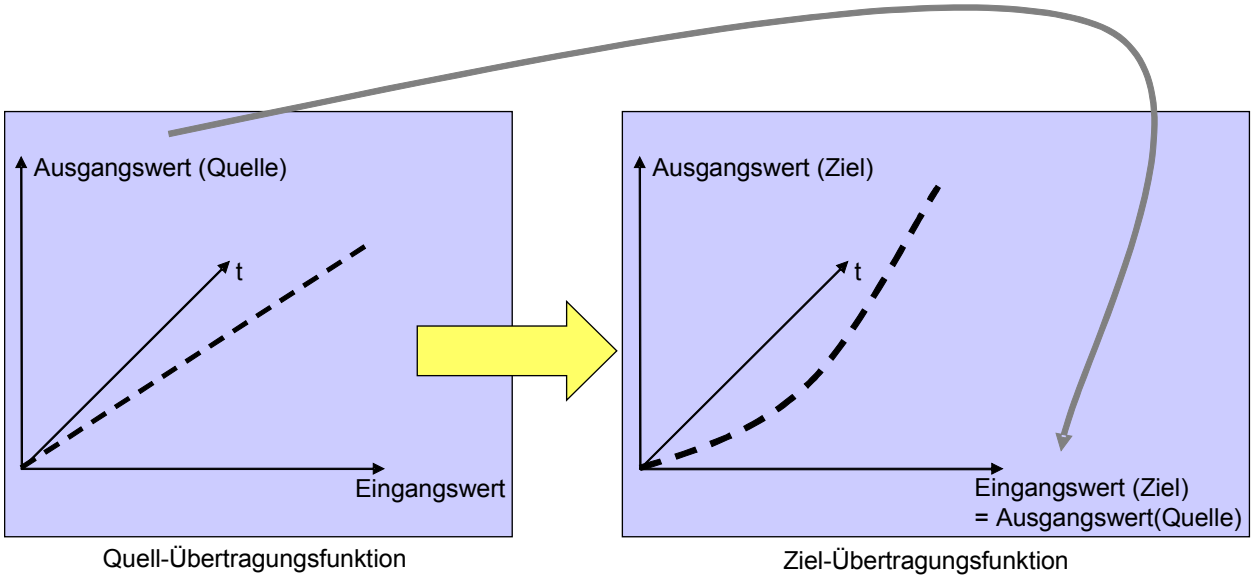


Abbildung 32: Kopplung von Übertragungsfunktionen

3.3.5.2 Abbildung von Wirkräumen in Schablonen

Mit dem Konzept der Schablonen können Wirkräume vordefiniert werden. Dies ist sinnvoll, wenn direkte Abhängigkeiten zwischen den Merkmalen der beteiligten Komponenten nicht definiert werden können oder der Aufwand zu groß ist, direkte Abhängigkeiten fest abzubilden. Dazu werden in den Schablonen entsprechende Wirkräume definiert, in denen die beteiligten Komponenten mit ihren Eigenschaften und Bedingungen über Adjazenzbeziehungen sowie die relevanten vorinitialisierten Kontaktsysteme bereits enthalten sind.

3.3.5.3 Kontaktsystembibliotheken

Kontaktsysteme werden, ähnlich wie Produktschablonen, in allgemeiner Form in Bibliotheken zusammengefasst, in denen sie nach Effekten geordnet abgelegt und verwaltet werden können. In einem allgemeinen Kontaktsystem werden in den Blättern des Binärbaums Platzhalter für Eigenschaften und Bedingungen mit festgelegten Einheiten definiert. Die Platzhalter sind über geeignete mathematische Verknüpfungen in den Knoten miteinander verbunden. Konkrete Kontaktsysteme werden aus allgemeinen Kontaktsystemen instanziiert, indem die Platzhalter mit den entsprechenden Eigenschaften bzw. Bedingungen der modellierten Produktkomponenten besetzt werden. Sie stehen damit einer einfachen Verwendung zur Verfügung und können nach Bedarf in Wirkräume eingebunden werden.

Die Auswahl und Instanziierung von geeigneten Kontaktsystemen mit passenden Eigenschaften und Bedingungen der Produktkomponenten kann ebenfalls durch den Rechner unterstützt werden, indem die Verteilung der Eigenschaften und Bedingungen im betrachteten Wirkraum innerhalb des Produktmodells untersucht wird und daraufhin passende Kontaktsysteme aus der Bibliothek vorgeschlagen werden.

3.4 Fazit

Es wurde ein Konzept entwickelt, das die Anforderungen an eine anforderungsgetriebene, integrierte Produktentwicklung erfüllt und die Defizite existierender Lösungen ausfüllt. Dazu wurden Ansätze für die Anbindung von Produktmodellierungswerkzeugen entwickelt sowie für die Verknüpfung der in ihren Produktdatenmodellen enthaltenen Informationen. Basis dafür ist die Integration ihrer Produktdatenmodelle in ein flexibles, domänenübergreifendes Produktmodell. Des Weiteren wurden Konzepte für die domänenübergreifende Erkennung, Abbildung und Auswertung von Abhängigkeiten zwischen Produktmerkmalen entwickelt. Im folgenden Kapitel wird die Implementierung der vorgestellten Konzepte beschrieben.

4 Implementierung

Die in Kapitel 3 vorgestellten Konzepte wurden auf Basis des Systems zur Anforderungsmodellierung aus dem Teilprojekt F6 „Entwicklung eines Anforderungsmodellierers“ des Transferbereichs (TFB) 16 [TFB-2003] in der Programmiersprache Java implementiert [Sun-2005] [Schi-2002] und mündeten in der Entwicklung des **S**ystems zur **A**nforderungsbasierten **E**valuierung von **P**rodukteigenschaften (*SAEP*).

Die dauerhafte, sitzungsübergreifende Speicherung der im System erzeugten Informationen erfolgt in der vorliegenden Version mit Hilfe der relationalen Datenbank MySQL [MySQ-2005] [ReYK-2002].

Die Komponenten von SAEP lassen sich mehreren Schichten zuordnen (siehe Abbildung 33):

- *Benutzungsoberfläche*: Sie stellt die Komponenten für die Interaktion des Benutzers mit dem System dar. Durch die klare Abgrenzung der Benutzungsoberfläche vom Rest des Systems kann diese unabhängig von anderen Komponenten verändert oder erweitert werden.
- *Anwendungslogik*: auf dieser Ebene ist die eigentliche Funktionalität implementiert.
- *Integrationsschicht*: auf dieser Ebene werden die Modellinhalte einheitlich und transparent[27] für die Anwendungslogik bereitgestellt. Das bedeutet, dass es aus Sicht der Anwendungslogik gleichgültig ist, aus welchen Datenquellen die Objekte stammen, die von ihr dargestellt oder manipuliert werden.
- *Persistenzschicht*[28]: die Komponenten dieser Ebene sind für das Lesen und Schreiben der Modellinhalte verantwortlich. Dies können zum einen externe Modellierungssysteme sein oder im Falle von von SAEP erzeugten Informationen die für die Persistierung benutzten Datenbanken sein.

27 “Transparent” bedeutet, dass der Zugriff auf alle Objekte gleichförmig ist.

28 *Persistenz*: dauerhafte Speicherung von Daten in nicht-flüchtigen Datenspeichern

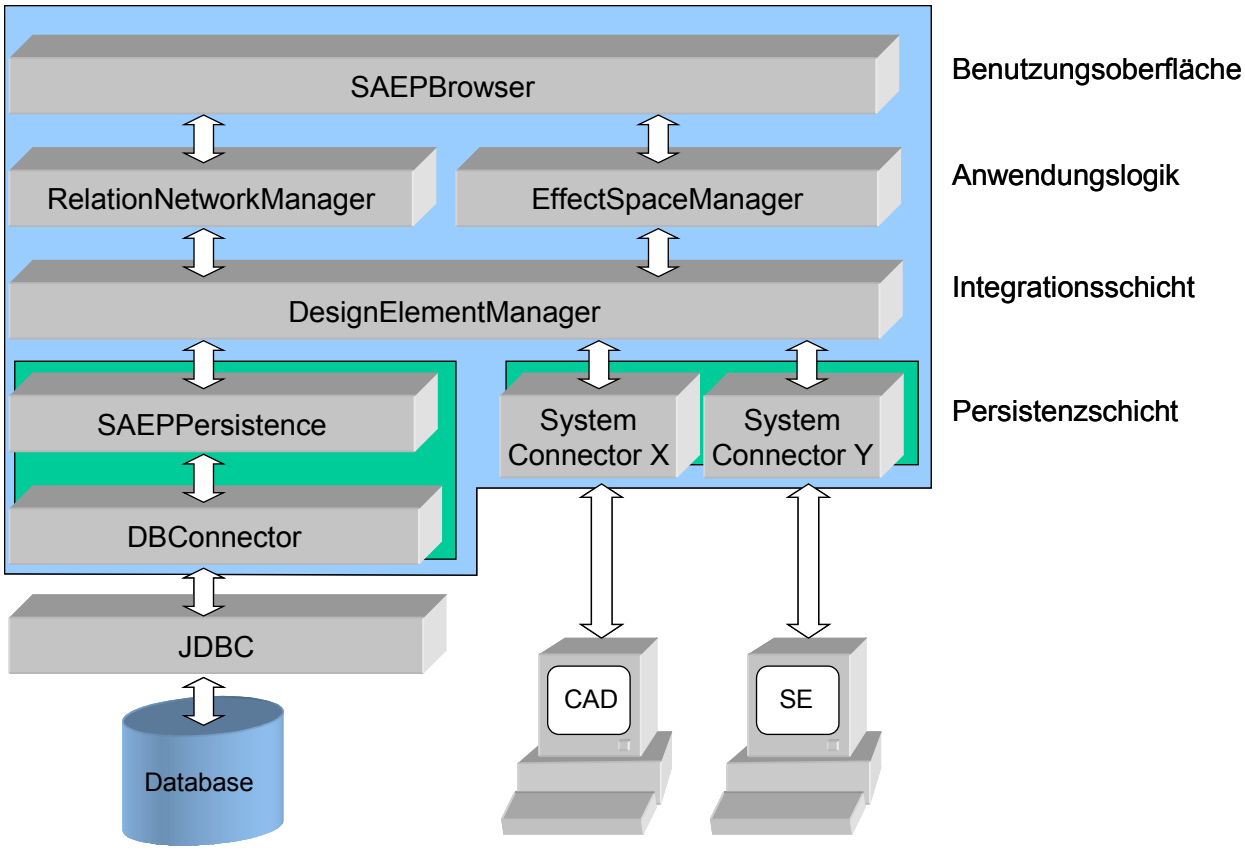


Abbildung 33: Architektur von SAEP

4.1 Datenmodell

Das Datenmodell ist für eine flexible Persistierung ausgelegt. Dazu wurden zwei Interfaces[29] definiert:

- *SAEPObject*: dieses Interface definiert Methoden für den Zugriff auf das eindeutig identifizierende Merkmal eines Objekts (ID) in Form einer Zeichenkette (String). Objekte, die mittels der SAEP-eigenen Persistenzschicht gespeichert werden sollen, müssen dieses Interface implementieren.
- *ExternalObject*: dieses Interface ist von *SAEPObject* abgeleitet und definiert zusätzliche Methoden für den Zugriff auf die ID eines externen Objekts (*externalID*) in seinem Ursprungssystem sowie für den Zugriff auf die ID des angebundenen Ursprungssystems an sich (*serviceID*). Die Klassen von Objekten, die als Repräsentanten von externen Objekten auftreten sollen, müssen dieses Interface implementieren.

---

[29] Ein Java-*Interface* definiert, welche Methoden ein Objekt zur Verfügung stellen (=implementieren) muss. Implementiert eine Klasse ein bestimmtes Interface, können Objekte auch als Objekte dieses Interfaces deklariert werden. Interfaces stellen gewissermaßen ein zugesichertes Verhalten eines Objekts dar. Die Einhaltung des Interfaces wird vom Compiler kontrolliert. Dabei wird lediglich das Vorhandensein der entsprechenden Methoden überprüft, nicht aber deren korrekte Funktion.

4.1.1 Abbildung von Produktmerkmalen

Zentrales Element des Datenmodells ist die Klasse *DesignElement* (siehe Abbildung 37). Objekte dieser Klasse fungieren als Bindeglieder zwischen Modellentitäten. Gleichzeitig bildet sie die Oberklasse für alle abgeleiteten Klassen, deren Objekte merkmalbehaftete Modellentitäten repräsentieren, d.h. Objekte, die zumindest im weitesten Sinne Elemente der Produktstruktur darstellen. Merkmale können sowohl ausgeprägte Eigenschaften als auch Anforderungen sein. Diese beiden Arten von Merkmalen (*Attribute*) haben prinzipiell den gleichen Aufbau und unterscheiden sich lediglich in ihrer Bedeutung als Ist-Eigenschaft (*Property*) bzw. Soll-Eigenschaft (*Requirement*). Merkmale (und damit Eigenschaften als Anforderungen) werden in Merkmale mit qualitativer bzw. quantitativer Ausprägung unterschieden. Qualitativ ausgeprägte Merkmale werden durch einen Text beschrieben (*description*), während quantitativ ausgeprägte Merkmale durch einen Wert (*Value*), bzw. einen Wertebereich beschrieben werden.

Value-Objekte benutzen für die Darstellung von Werten Objekte der Klasse *Quantity* aus dem externen Paket[30] *JSciences* (*org.sciences.physics.quantity* [Daut-2005], [Daut-2001]). Physikalische Größen (etwa Längen, Flächen, Massen, etc.) können damit invariant gegenüber Maßeinheiten (*Unit*) verwaltet und deren Wert in jeder gewünschten, kompatiblen Maßeinheit abgefragt werden. JScience enthält Module, die eine automatische Einheitenumrechnung durchführen können. Die Klasse *Unit* stammt ebenfalls aus dem externen Paket JSciences (*org.jsciences.physics.units*).

Wertebereiche haben eine untere Grenze (*lower*) und eine obere Grenze (*upper*), die entweder zum Intervall dazu gehört, (geschlossenes Intervall, das boolesche Attribute *include* ist *TRUE*) oder nicht (offenes Interval, *include* ist *FALSE*). Ist die Merkmalsausprägung kein Wertebereich, so ist nur das *Value*-Objekt *opt* mit einem Wert belegt. Im Fall von Anforderungen kann bei Wertebereichen zusätzlich eine Optimierungsrichtung angegeben werden, die im Falle einer Optimierungsanforderung angibt, in welche Richtung des Intervalls die auszuprägende Ist-Eigenschaft optimiert werden soll. Dies wird durch die Booleschen Variablen *lowerIsOptimum* bzw. *upperIsOptimum* ausgedrückt[31]. Ein Merkmal trägt immer eine Merkmalsbezeichnung (*AttributeName*). Ein ausgeprägtes Merkmal trägt zusätzlich eine Merkmalsausprägung (*AttributeValue*). Dies erlaubt es, eindeutige Bezeichnungen für Merkmale zu vergeben und so semantische Eindeutigkeit zu gewährleisten.

Die Eigenschaften eines Produkts sind oft auf einer sehr hohen Detaillierungsstufe beschrieben, wie z.B. Punktkoordinaten oder Kantenlängen in B-Rep-Geometriemodellen. Die mit Anforderungen beschriebenen Soll-Eigenschaften eines Produkts werden in der Regel mit einem niedrigeren Detaillierungsgrad angegeben, wie z.B. die Bauteilgesamtlänge. Das bedeutet, dass es möglich sein muss, aus den detailliert definierten Produkteigenschaften ab-

30 Der Paketmechanismus (engl. „package mechanism“) von Java sorgt für eine Gruppierung von zusammengehörigen Klassen in hierarchisch aufgebaute Namensräumen (packages), um Namenskonflikte zu vermeiden [Schi-2002].

31 Hierbei muss die Anwendungslogik sicherstellen, dass nicht beide Werte auf TRUE gesetzt sind.

straktere Eigenschaften zusammenzustellen[32] (Kantenlängen addieren sich beispielsweise zur Bauteilgesamtlänge). Abbildung 34 zeigt die Klassenstruktur zur Abbildung von Merkmalen.

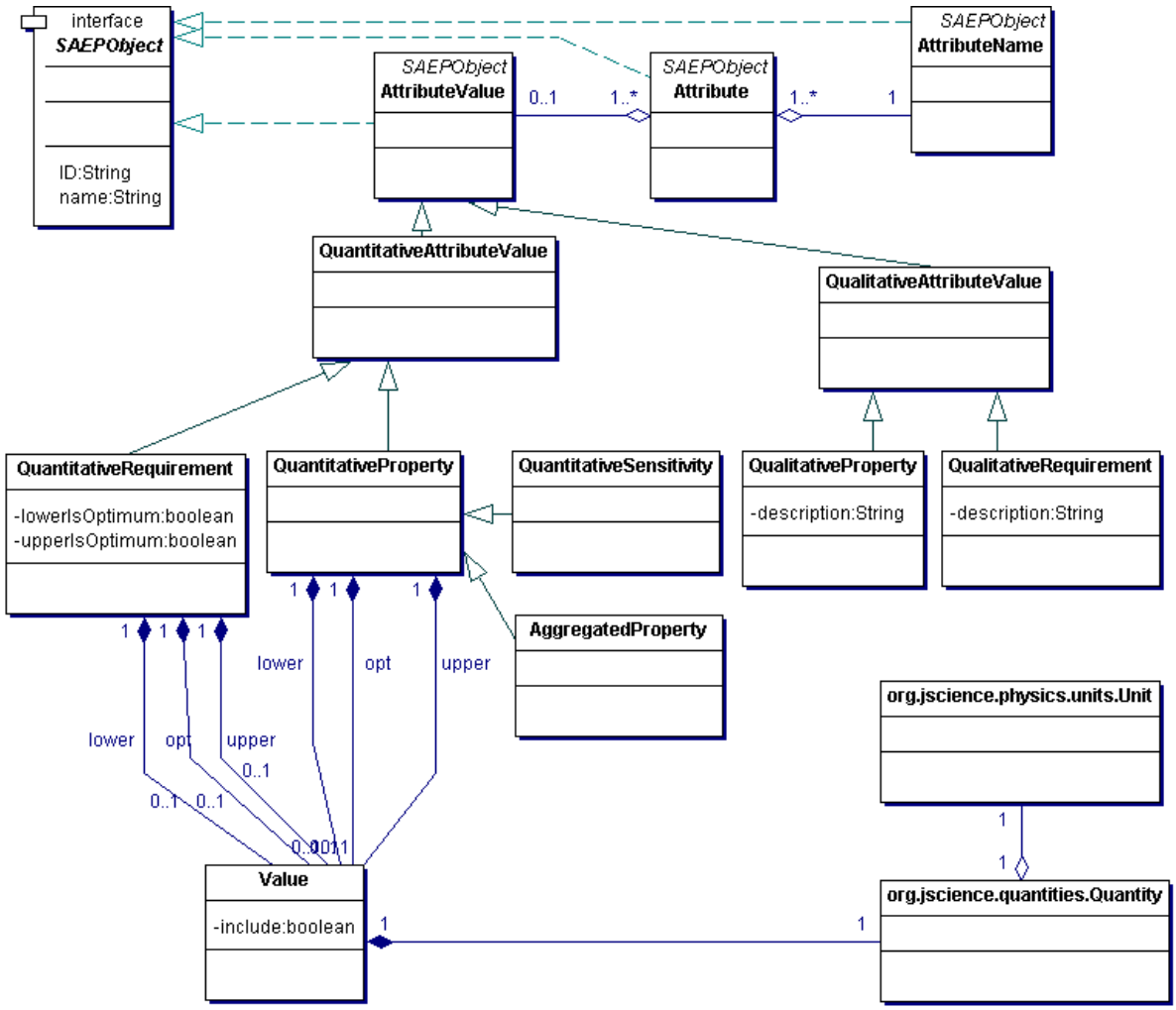


Abbildung 34: Klassenstruktur zur Abbildung von Merkmalen

Eine weitere Art von Merkmalen stellen *Sensibilitäten* (*QuantitativeSensitivity*) dar. Sensibilitäten sind Eigenschaften, die während des Produktbetriebs variabel sind und von außen beeinflusst werden können. Das bedeutet, das Vorhandensein einer Sensibilität bei einer Modellentität drückt aus, dass die Komponente über diese Eigenschaft beeinflusst werden kann. In Verbindung mit einer zugeordneten Anforderung, die die zulässigen Werte für eine Beeinflussung vorgibt, wird eine Bedingung definiert, die für die Funktionsfähigkeit dieser Modellentität erfüllt sein muss. Prinzipiell sind Sensibilitäten nichts anderes als im Einbauzusammenhang variable Eigenschaften von Modellentitäten. Von daher könnten sie auch entsprechend modelliert werden. Die Unterscheidung zwischen festgelegten Eigenschaften und von Sensibilitäten, über die die Komponente von außen beeinflusst werden kann, ist aber für den Aufbau von Kontaktsystemen wichtig und erlaubt es außerdem, Kontaktsysteme

[32] Dazu sind Objekte der Klasse *AggregatedProperty* vorgesehen, die im weiteren Verlauf dieses Kapitels vorgestellt wird.

komponentenweise zu definieren, mit deren Hilfe die Beeinflussungen von außen auf variable Eigenschaften der Komponente abgebildet werden kann.

Entsprechend der oben beschriebenen und in Abbildung 34 dargestellten Klassenstruktur für die Abbildung von in SAEP erzeugten merkmalsbezogenen Informationen, gibt es eine Klassenstruktur zur Abbildung extern erzeugter Informationen. Dazu wurden die Basisklassen *ExtAttribute*, *ExtAttributeValue*, *ExtAttributeName* und *ExtValue* von entsprechenden Klassen abgeleitet und eine äquivalente Unterstruktur angelegt. Abbildung 35 zeigt die Klassenstruktur zur Abbildung von Merkmalen extern erzeugter Objekte.

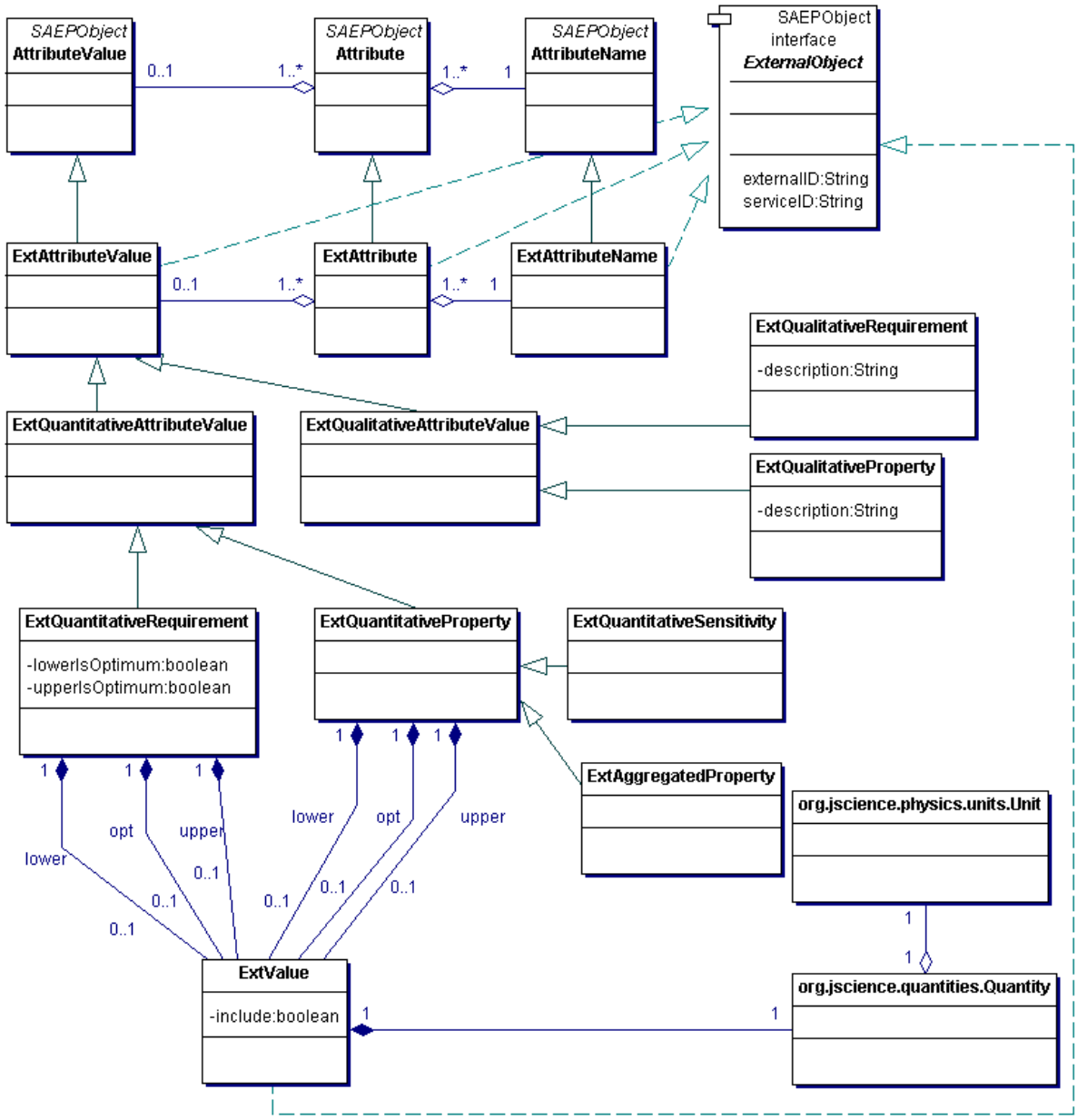


Abbildung 35: Klassenstruktur zur Repräsentation von Merkmalen externer Objekte

4.1.2 Abbildung von Produktmerkmalsbibliotheken

Das Konzept der Produktmerkmalsbibliotheken wird mittels der Klasse *AttributeLibrary* verwirklicht. Der Ansatz nach GEBAUER [Geba-2001][Rzeh-1998] wurde insoweit weiterentwickelt, dass in den Produktmerkmalsbibliotheken nicht nur Anforderungen verwaltet werden, sondern allgemein Produktmerkmale, die durch den Benutzer in Anforderungen oder Eigenschaften unterschieden werden können (siehe auch 3.2.1). Des Weiteren können Sichten auf die hierarchische Struktur von Anforderungssammlungen definiert werden. Da die Beziehungen zwischen den Produktmerkmalen lediglich zur intuitiven Navigation in Produktmerkmalssammlungen durch den Benutzer dienen soll, kann auf eine exakte Definition der Semantik der Beziehungen verzichtet werden. Das bedeutet, dass die Beziehungen teilweise eher die Bedeutung einer Aggregation haben können (z.B. „Länge“ ist ein Merkmal unter anderen zur Definition des „Volumens“) und teilweise eher die Bedeutung einer Konkretisierung haben (z.B. „spezifisches Gewicht“ ist eine Konkretisierung des Merkmals „Gewicht“). Die Strukturierung von Produktmerkmalssammlungen nach ergonomischen Gesichtspunkten stand zudem nicht im Mittelpunkt der vorliegenden Arbeit.

Die Klasse *AttributeLibrary* stellt die oberste Aggregationsstufe der Struktur dar. Sie ist abgeleitet von der Klasse *ViewRepository* zur Verwaltung von verschiedenen Sichten auf hierarchische Strukturen von Objekten. In Fall der Merkmalsbibliotheken lassen sich dadurch Merkmale unterschiedlich Gruppieren. Die hierarchische Struktur wird, wie bei der Abbildung von Produktstrukturen, über *ParentChildRelation*-Objekte aufgebaut, die in Objekten der Klasse *View* abgelegt sind.

AttributeLibrary-Objekte können von einem anderen *AttributeLibrary*-Objekt abgeleitet werden. Eine Referenz auf die Ursprungsbibliothek wird als *parentLibrary*-Attribut in der abgeleiteten Bibliothek gespeichert. Abbildung 36 zeigt die Klassenstruktur zur Abbildung von Produktmerkmalsbibliotheken.

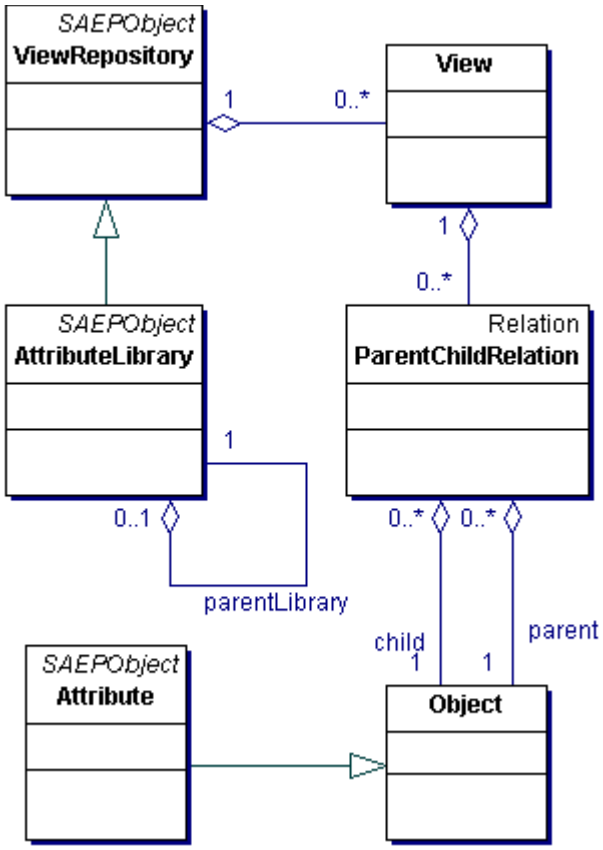


Abbildung 36: Klassenstruktur zur Abbildung von Produktmerkmalsbibliotheken

4.1.3 Abbildung und Verknüpfung von Produktstrukturelementen

DesignElement-Objekte bilden die Oberklasse für alle aggregationsfähigen, merkmalbehafteten Produktstrukturelemente. Sie stellen, wie alle von SAEP persistent zu haltenden Objekte, Methoden für den Zugriff auf das von SAEP's Persistenzschicht vergebene Identifikationsmerkmal (ID) bereit. Die Aggregation von Modellentitäten (Produktstruktur), die durch *DesignElement*-Objekte repräsentiert werden, wird mit Hilfe spezieller Verknüpfungsobjekte realisiert, die durch die Klasse *ParentChildRelation* definiert sind. *ParentChildRelation*-Objekte verknüpfen Objekte der Klasse *Object*, die die Oberklasse aller in Java verwendeten Klassen darstellt. Daher sind sie sehr flexibel einsetzbar. In einigen Fällen ist eine semantisch eindeutigere Verknüpfung notwendig. Dann werden Objekte von Klassen benutzt, die die Klasse *ParentChildRelation* geeignet spezialisieren.

ParentChildRelation-Objekte (und damit ebenso Objekte aus spezialisierenden Klassen) sind ihrerseits in Containerobjekten der Klasse *View* enthalten. Auf diese Weise können verschiedene Sichten auf ein Produkt abgebildet und verwaltet werden. *View*-Objekte sind wiederum in Containerobjekten der Klasse *ViewRepository* enthalten. Ein *ViewRepository*-Objekt stellt damit alle möglichen Sichten auf ein Produkt (*Product*) zusammen. In Objekten der Klasse *Product* sind alle Informationen, die ein bestimmtes Produkt betreffen, zusammengefasst. Abbildung 37 zeigt die Abbildung von Produktstrukturen in Sichten. Von *Product* ausgehend

können alle Datenmodellentitäten durch Navigation entlang der datenmodellinternen Aggregationsbeziehungen erreicht werden[33].

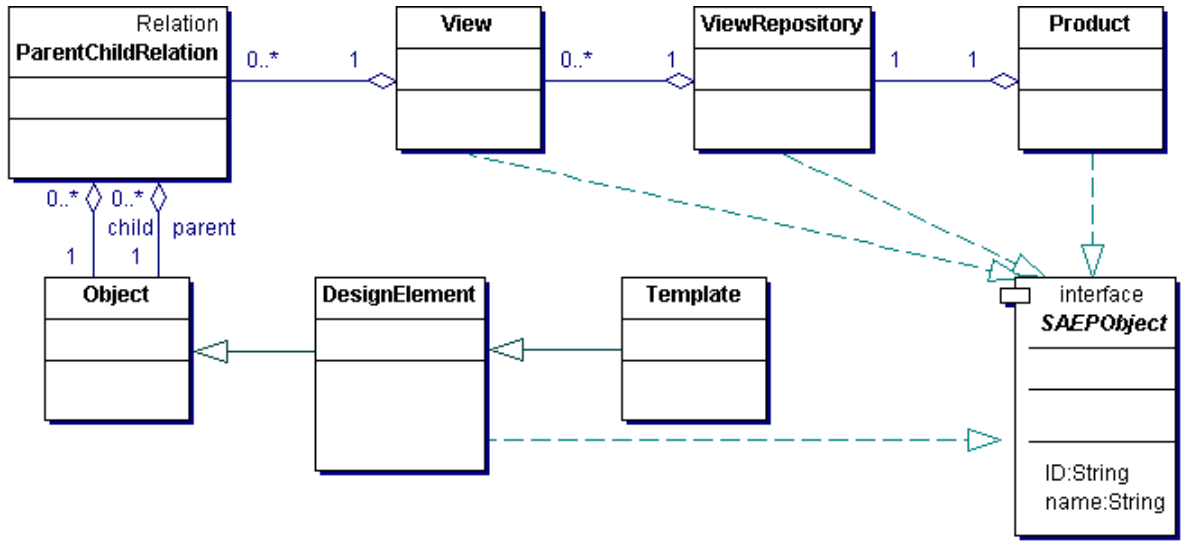


Abbildung 37: Abbildung von Produktstrukturen

Eine Spezialisierung der Klasse *DesignElement* ist die Klasse *Template*, mit deren Hilfe das Konzept der Schablonen umgesetzt wurde (siehe Seite 55 und Abschnitt 3.3.4). Objekte dieser Klasse dienen als Vorlage für zu entwickelnde Produktkomponenten. Diese Klasse dient nur zur Unterscheidung zwischen konkreten Produktkomponenten und Schablonen d.h., Objekte dieser Klasse erben alle Eigenschaften ihrer Oberklasse ohne eigene, klassenspezifische Erweiterungen hinzuzufügen. Ebenso können sie mit allen Beziehungsobjekten verknüpft werden, mit denen auch *DesignElement*-Objekte verknüpft werden können. Soll aus einem *Template*-Objekt eine Produktkomponente abgeleitet werden, so muss das *Template*-Objekt geklont[34] werden sowie von ihm ausgehend alle *ParentChildRelation*-Objekte. Die von ihnen verknüpften Objekte werden dagegen standardmäßig als Referenzen kopiert, damit Änderungen am Originalobjekt automatisch propagiert werden können. Ausnahmen bilden dabei *Template*-Objekte. Sie werden ebenfalls geklont.

Für die Verknüpfung von Objekten aus externen Systemen mit dem SAEP-Datenmodell sind Objekte der Klasse *ConnectedDesignElement* vorgesehen, die direkt von der Klasse *DesignElement* abgeleitet ist. Sie sind Repräsentanten der externen Objekte und implementieren das Interface *ExternalObject*, d.h. sie stellen Methoden für den Zugriff auf die ID des von ihnen referenzierten Objekts (*externalID*) bereit sowie Methoden für den Zugriff auf die eindeutige ID ihres Ursprungssystems (*serviceID*). Von *ConnectedDesignElement* wiederum abgeleitet

---

33 Hier sind Aggregationsbeziehungen im Zusammenhang mit dem Paradigma der Objektorientierung gemeint, das heißt, dass Datenobjekte Attribute anderer Datenobjekte sind (Aggregation).

34 *Klonen* bedeutet in diesem Zusammenhang, dass rekursiv von allen Attributen eines Objekts Kopien („Klone“) erzeugt und ihrerseits als Attribute des Objekts eingefügt werden.

sind in der vorliegenden Implementierung die abstrakten[35] Klassen *CADElement* für die Anbindung von CAD-Modellen und *SystemEngineeringElement* für die Anbindung von System Engineering Modellen. Abbildung 38 zeigt die Klassenstruktur für die Abbildung von Komponenten der Produktstruktur.

Die Unterklassen von *CADElement* sind an die übliche Struktur von B-Rep-Modellen angelehnt (siehe Abschnitt 2.3.1). Das bedeutet, es findet ebenfalls eine Trennung zwischen topologischen Elementen (*TopologyElement*) und geometrischen Elementen (*GeometryElement*) statt. Die topologischen Elemente eines Geometriemodells sind Baugruppe (*Assembly*), Einzelteil (*Part*), Körper (*Body*), Flächenverband (*Shell*) Fläche (*Face*), Kante (*Edge*) und Eckpunkt (*Vertex*). Einer Fläche kann eine Oberfläche (*Surface*), einer Kante eine Kurve (*Curve*) und einem Eckpunkt ein geometrischer Punkt (*Point*) zugeordnet werden, um die sichtbare Gestalt zu beschreiben. Die Aggregationsbaumstruktur wird wie bei anderen *DesignElement*-Objekten mit Hilfe von *ParentChildRelation*-Objekten hergestellt.

Systeme aus externen System Engineering Modellen werden als Objekte der Klasse *System* abgebildet. Durch die Verknüpfung mit *ParentChildRelation*-Objekten können Systemhierarchien zusammengestellt werden. Systemkomponenten, die nicht weiter in Einzelsysteme zerlegt werden sollen oder können, werden als Objekte der Klasse *Component* abgebildet. Systeme und Einzelkomponenten können *Eingangs-* und *Ausgangsports* besitzen, über die beliebige Größen übertragen werden können. Die Art der am Port anliegenden Größe wird durch ein Objekt der Klasse *Unit* repräsentiert. Eventuelle Übertragungsfunktionen von Eingans- zu Ausgangsports innerhalb von Komponenten stehen nicht im Mittelpunkt der vorliegenden Implementierung von SAEP. Verbindungen zwischen Ports zweier verschiedener Systeme bzw. Komponenten werden durch Verknüpfungsobjekte der Klasse *Port2PortRelation* beschrieben (siehe Abbildung 40).

35 *Abstrakte Klasse* sind in Java Klassen, in denen mindestens eine Methode nur deklariert aber nicht zusätzlich implementiert ist. Von abstrakten Klassen können keine Objekte erzeugt werden.

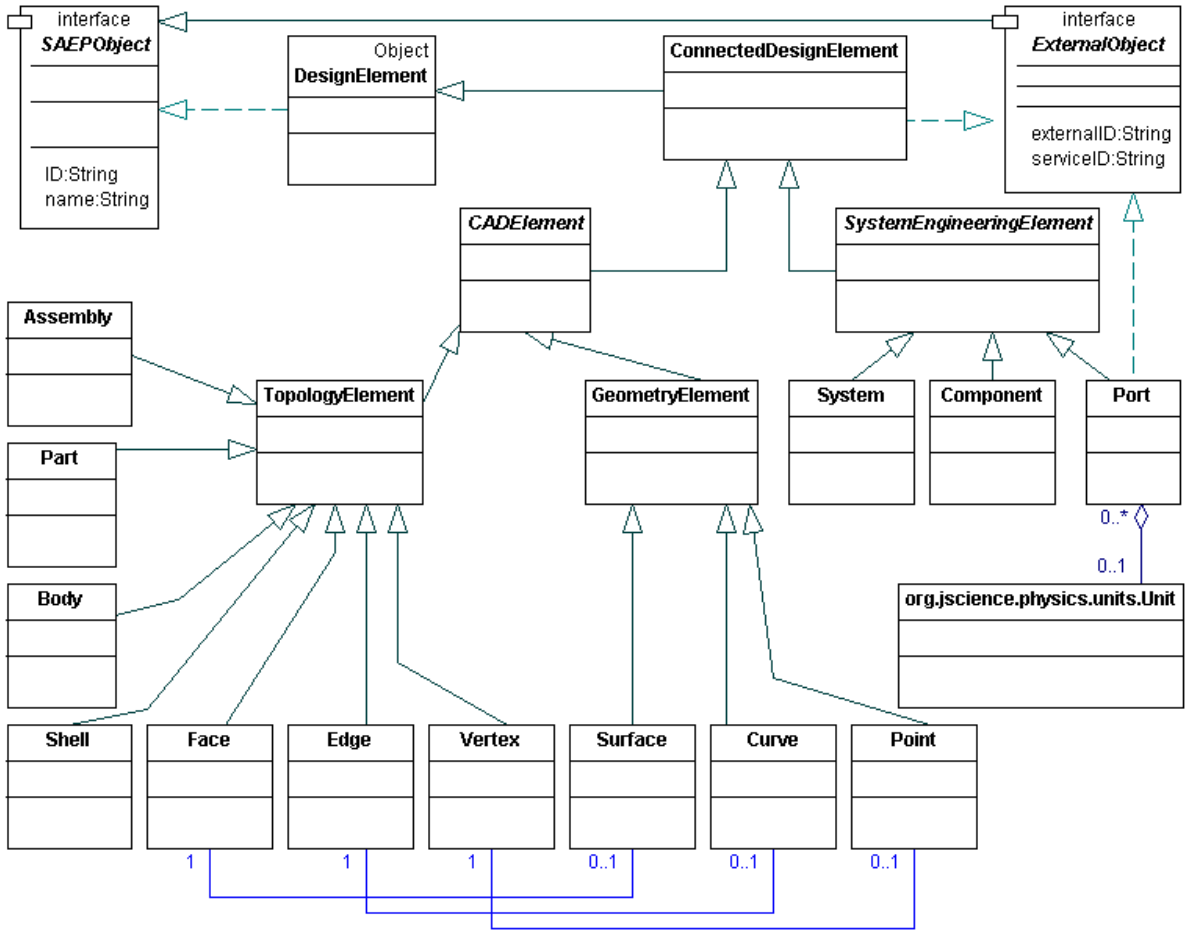


Abbildung 38: Klassenstruktur zur Abbildung von Modellentitäten aus externen Werkzeugen

Sichten auf die Produktstruktur werden wie bei allgemeinen *DesignElement*-Objekten über Objekte der Klasse *ParentChildRelation* aufgebaut und in *View*-Objekten zusammengefasst. Die Beziehungen von allgemeinen *DesignElement*-Objekten zu *ConnectedDesignElement*-Objekten, die aus externen Werkzeugen stammen, werden mit Hilfe von Objekten der Klasse *DEConnectedDERelation* (DesignElementConnectedDesignElementRelation) abgebildet, wie in Abbildung 39 dargestellt. Diese Klasse ist eine Spezialisierung der Klasse *ParentChildRelation*. Objekte dieser Klasse werden zusammen mit Objekten der Klasse *ParentChildRelation* in *View*-Objekten zusammengefasst.

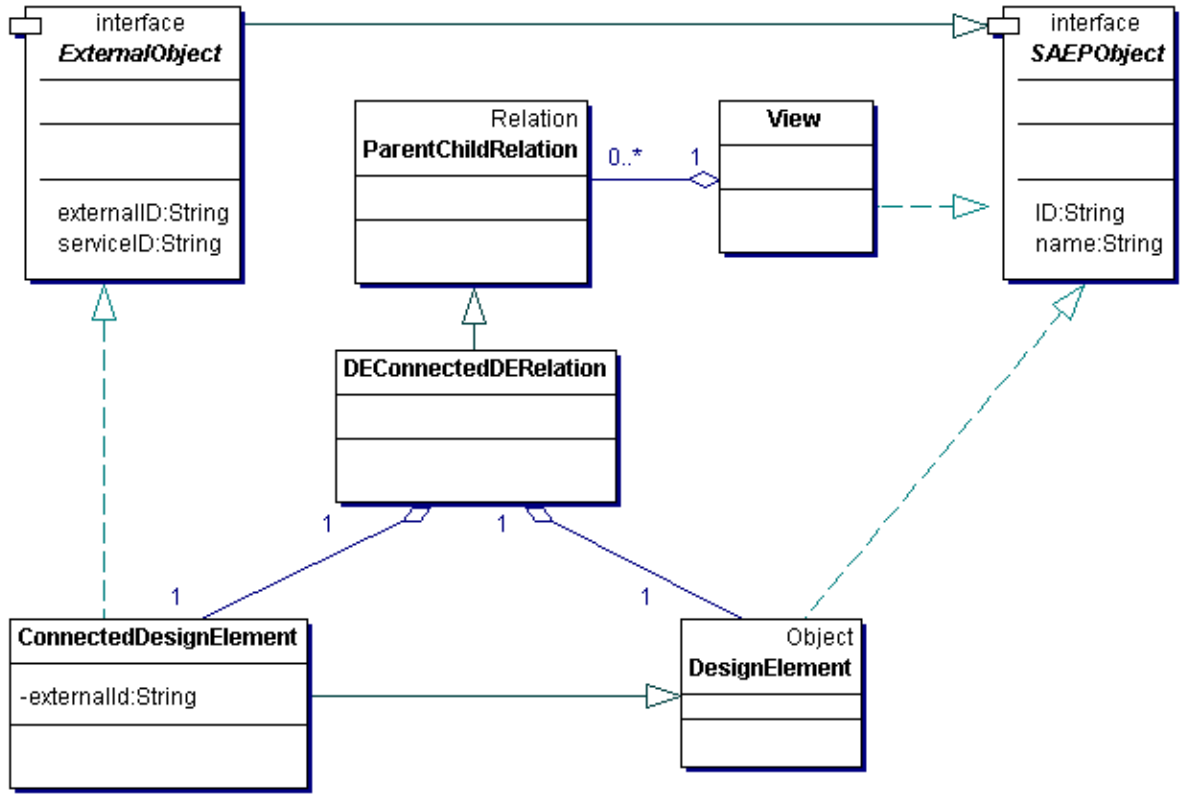


Abbildung 39: Die Verknüpfung von DesignElement-Objekten mit externen Modellentitäten

Eine weitere Möglichkeit zur Abbildung von Beziehungen zwischen Elementen der Produktstruktur bilden Objekte der Klasse *AdjacencyRelation*. Mit ihrer Hilfe werden Nachbarschaftsbeziehungen zwischen Modellentitäten ausgedrückt, d.h., dass sich zwei Elemente potentiell gegenseitig beeinflussen können. Nachbarschaftsbeziehungen können sich aus geometrischen Gegebenheiten ergeben, etwa wenn sich zwei Elemente direkt berühren oder in räumlicher Nähe zueinander stehen. Sie können aber auch allein durch eine direkte oder indirekte Verbindung zwischen ihnen entstehen, beispielsweise wenn zwei Elemente durch eine elektrische Leitung zur Energie- oder Informationsübertragung miteinander verbunden sind. Dieser Zusammenhang wird durch die Klasse *Port2PortRelation* unterstrichen, deren Objekte Ports in System Engineering Modellen miteinander verbinden. Sie ist abgeleitet von der Klasse *AdjacencyRelation*. Abbildung 40 zeigt eine Übersicht über die Klassen zur Verknüpfung von Produktstrukturelementen.

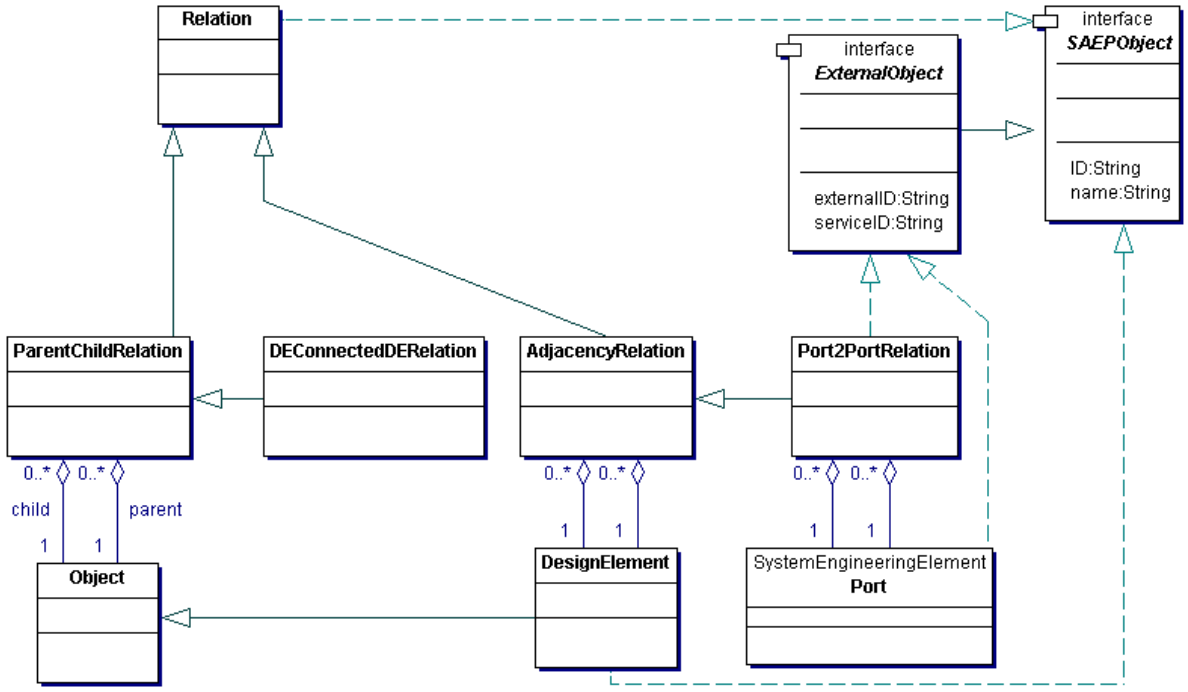


Abbildung 40: Klassen zur Verknüpfung von Produktstrukturelementen

4.1.4 Verknüpfung von Produktmerkmalen und Produktstrukturelementen

Ein und dasselbe Merkmal[36] kann mehreren Modellentitäten gleichzeitig zugeordnet sein. Dadurch wird Datenredundanz vermieden und die Pflege des Modellinhalts bei Änderungen von Merkmalsausprägungen erleichtert. Diese Vorgehensweise hat zunächst jedoch einen Nachteil: Abhängigkeiten, die direkt zwischen Merkmalen modelliert sind, würden automatisch bei allen Modellentitäten auftreten, denen die abhängigen Merkmale zugeordnet sind (da es sich jeweils um dasselbe Merkmalsobjekt handelt). Daher werden Merkmale nicht direkt Produktkomponenten zugeordnet sondern indirekt über spezielle Verknüpfungsobjekte der Klasse *DesignElementAttributeRelation*, die jeweils ein Merkmal mit einer Produktkomponente verknüpfen. Diese Verknüpfungsobjekte werden einmalig erzeugt und ermöglichen daher die eindeutige Identifikation der *Zuordnung* eines Merkmals zu einer Produktkomponente. Um nun Abhängigkeiten zwischen Merkmalen abhängig vom Kontext ihrer Zuordnung zu bestimmten Modellentitäten modellieren zu können, werden die Abhängigkeiten zwischen den Merkmalen nicht direkt modelliert, sondern indirekt über Abhängigkeiten zwischen speziellen Verknüpfungsobjekten ausgedrückt.

In Abbildung 41 ist die Oberklasse *AbstractRelation* aller Klassen dargestellt, die mathematisch nicht beschreibbare Abhängigkeiten repräsentieren. Sie implementiert das Interface *BinaryRelation*, das von allen Beziehungsklassen, die bei einer Abhängigkeitsanalyse berücksichtigt werden sollen, implementiert werden muss. Alle anderen Klassen, die Abhängigkeiten

---

36 genauer: Objekte, die Merkmale repräsentieren.

zwischen Merkmalen abbilden, verknüpfen ebenfalls Objekte der Klasse *DesignElementAttributeRelation* und nicht direkt Objekte der Klasse Attribute.

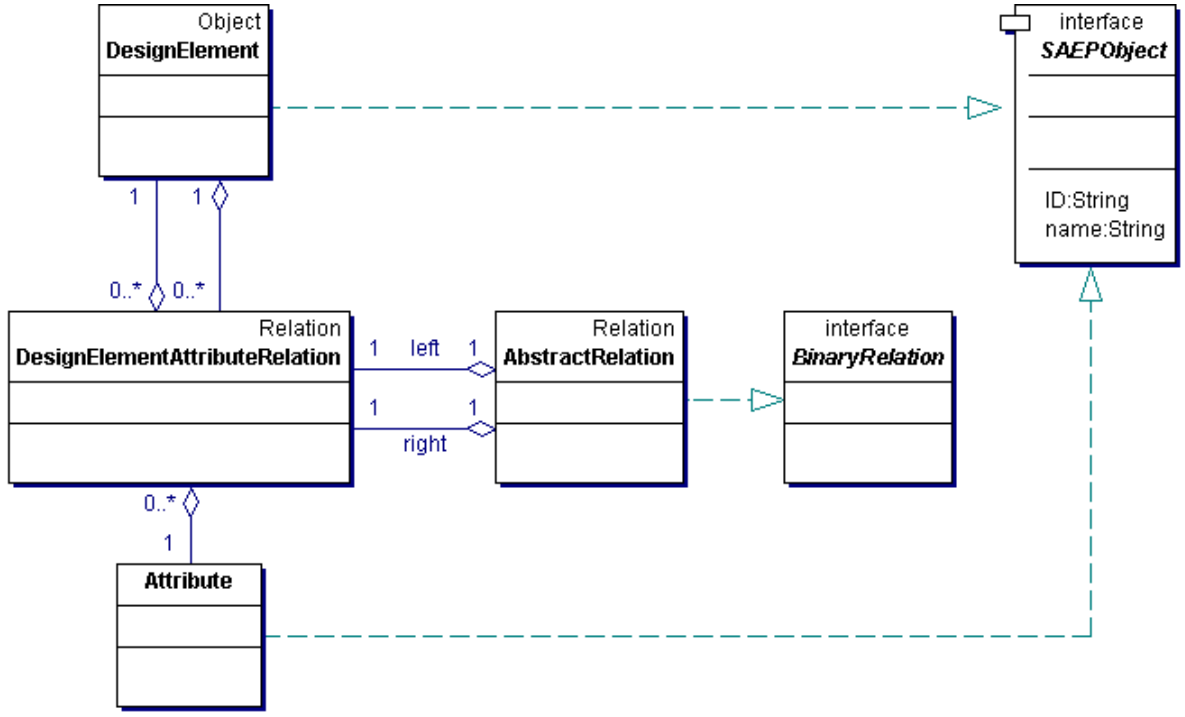


Abbildung 41: Verknüpfung von Attribute- und DesignElement-Objekten

4.1.5 Abbildung von Abhängigkeiten zwischen Merkmalen

Abhängigkeiten zwischen Merkmalen von Modellentitäten werden mittels dedizierter Verknüpfungsobjekte abgebildet. Die Oberklasse aller Verknüpfungsobjekte und auch der Merkmalsabhängigkeiten ist die Klasse *Relation*. In ihr sind allgemein notwendige Attribute definiert, wie Bezeichnung, ID, Kommentar etc. Wie in Abschnitt 4.1.1 erläutert, werden Abhängigkeiten zwischen Merkmalen nicht direkt modelliert, sondern indirekt über eine Verbindung der Verknüpfungsobjekte, die die Zugehörigkeit eines Merkmal-Objekts zu einem *DesignElement*-Objekt festlegen (siehe oben, Abbildung 41). Alle Beziehungen, die innerhalb eines Produkts zwischen Merkmalen bestehen, sind in einem Objekt der Klasse *RelationContext* enthalten. Ein *RelationContext*-Objekt ist daher genau einem *Product*-Objekt zugeordnet.

4.1.5.1 Nicht-mathematisch beschreibbare Abhängigkeiten

Nicht-mathematisch beschreibbare Beziehungen zwischen Anforderungen werden durch Objekte der Klasse *AbstractRelation* ausgedrückt. Diese Beziehungen lassen sich weiter unterteilen in *Unterstützende Beziehungen* (*SupportingRelation*), *Konkurrierende Beziehungen* (*CompetingRelation*) und *Ausschließende Beziehungen* (*ExcludingRelation*). Diese Unterklassen von *AbstractRelation* erben alle Attribute und fügen selbst keine weiteren hinzu, d.h.

sie dienen lediglich zur Klassifikation von Beziehungen zwischen Merkmalen. Abbildung 42 zeigt die Klassenstruktur der möglichen, nicht-mathematischen Beziehungen zwischen Merkmalen.

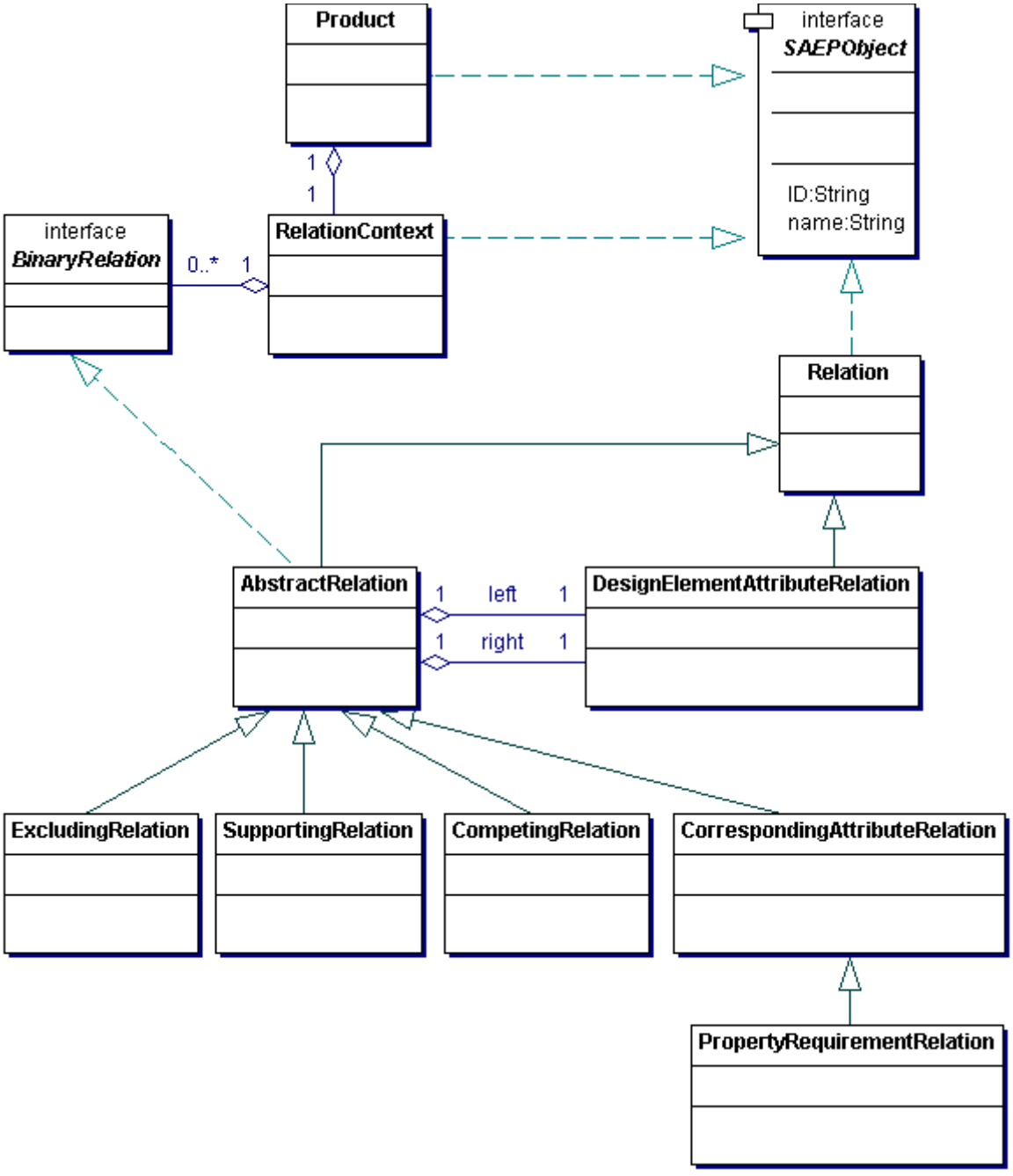


Abbildung 42: Klassen zur Beschreibung von nicht-mathematischen Beziehungen zwischen Merkmalen

Objekte der Klasse *CorrespondingAttributeRelation* verknüpfen korrespondierende Merkmale, d.h. Merkmale, die dieselbe Bedeutung haben, aber aus unterschiedlichen Autorensyste-

men[37] stammen. Objekte der Klasse *PropertyRequirementRelation*, die von *CorrespondingAttributeRelation* abgeleitet ist, verknüpfen Eigenschaften mit Anforderungen. Durch Auflösung dieser Beziehungen werden die eigentlichen mathematischen Zusammenhänge für den Vergleich von Eigenschaft und zugehöriger Anforderung hergestellt. Wird die zugeordnete Anforderung durch einen Wertebereich repräsentiert, so ergeben sich zwei Ungleichungen mit deren Hilfe geprüft wird, ob die Eigenschaft oberhalb des unteren Grenzwertes und unterhalb des oberen Grenzwertes liegt. Wird die Anforderung durch einen Einzelwert repräsentiert, ergibt sich eine einzelne Gleichung.

4.1.5.2 Mathematisch beschreibbare Abhängigkeiten zwischen Merkmalen

Mathematisch beschreibbare Abhängigkeiten zwischen Merkmalen lassen sich in der Regel nur über komplexere Konstrukte darstellen. Lediglich im einfachsten Fall, wenn zwei Merkmale direkt über einen Gleichheits- bzw. Ungleichheitsoperator verknüpft sind, lässt sich dies ebenfalls über eine einfache binäre Beziehung ausdrücken. In den anderen Fällen, wenn Terme aus Merkmalen zu Gleichungen zusammengesetzt werden, können sich beliebig komplexe Binärbäume ergeben (siehe Abbildung 23 und Abschnitt 3.3.2, Seite 64). Abbildung 43 zeigt die Klassen zur Abbildung mathematisch beschreibbarer Abhängigkeiten zwischen Merkmalen. Grundklasse zur Repräsentation von Knoten in Binärbäumen zur Abbildung von Ausdrücken ist die Klasse *Expression*. Sie ist eine abstrakte Klasse, das bedeutet, dass von ihr nicht direkt Objekte instanziiert werden können. *Expression*-Objekte können entweder Unäre Operatoren (*UnaryOperator*, z.B. „sin“, „cos“) Binäroperatoren (*BinaryOperator*, z.B. „+“, „-“) oder mathematische Werte (*MathValue*) sein, die aus (Produkt-)Merkmalen gewonnen werden. Binäroperatoren verknüpfen immer zwei Ausdrücke (*left*, *right*). Die Klasse *BinaryOperator* implementiert das Interface *BinaryRelation* und damit stehen auch Objekte dieser Klasse für Abhängigkeitsanalysen zur Verfügung.

Vergleichsoperatoren (*CompareOperator*, z.B. „=“, „<“, „>“) sind eine Unterklasse von Binäroperatoren. Sie stellen immer die Wurzelknoten von Gleichungen dar. Mathematische Werte (*MathValue*) sind mit einem *DesignElementAttributeRelation*-Objekt assoziiert, aus dessen verknüpften Merkmal der eigentliche Wert für die Auswertung des Ausdrucks, bzw. der Gleichung gewonnen wird. Um die Funktionalität zur Auswertung von mathematischen Abhängigkeiten flexibel nutzen zu können, akzeptieren *MathValue*-Objekte als Attribute Objekte der allgemeinen Klasse zur Abbildung von Objekten, nämlich *Object*.

---

37 Autorensysteme sind die Systeme, in denen die ursprünglichen Informationen erzeugt und verändert werden. Beispiele dafür sind in diesem Zusammenhang CAD-Systeme, SE-Werkzeuge etc.

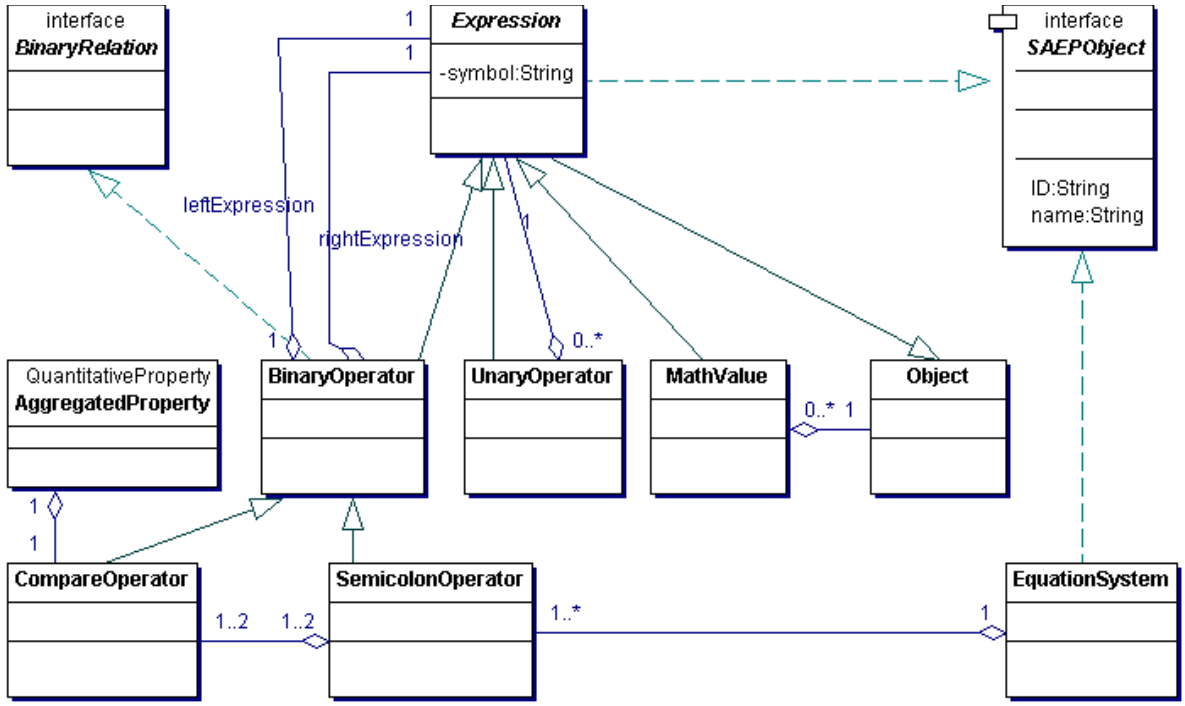


Abbildung 43: Klassenstruktur zur Abbildung mathematisch beschreibbarer Abhängigkeiten

Gleichungssysteme (*EquationSystem*) werden mit Hilfe von Objekten der Klasse *SemicolonOperator* zusammengestellt. Objekte dieser Klasse verknüpfen in der Regel zwei Objekte der Klasse *CompareOperator*, die ihrerseits die Wurzelknoten von Gleichungen darstellen. Wenn das Gleichungssystem nur aus einer Gleichung besteht, beinhalten *SemicolonOperator*-Objekte nur ein *CompareOperator*-Objekt.

Die Klasse *AggregatedProperty* ist für die Repräsentation von aggregierten Eigenschaften einer Produktmodellentität verantwortlich. Dazu ist sie mit einem Objekt der Klasse *CompareOperator* verknüpft. Die rechte Seite des Objekts verkörpert dabei die aggregierte Eigenschaft in Form eines *MathValue*-Objekts, während auf der linken Seite ein beliebig komplexer Teilbaum für die Verknüpfung der zu aggregierenden Eigenschaften sein kann[38].

4.1.6 Abbildung von Wirkräumen und Kontaktsystemen

Wirkräume werden durch Objekte der Klasse *EffectSpace* repräsentiert. Wirkräume sind DesignElement-Objekten zugeordnet. Das bedeutet, dass wenn das DesignElement. Wirkräumen sind *DesignElement*-Objekte zugeordnet sowie die in einem Wirkraum wirkenden Kontaktsysteme. Kontaktsysteme werden durch Objekte von Implementierungen des Java-Interface *ContactSystem* repräsentiert. Implementierungen von *ContactSystem* verknüpfen und evaluieren die Eigenschaften bzw. Bedingungen der *DesignElement*-Objekte, die einem Wirkraum zugeordnet sind. Durch die Verwendung eines Interface können auch externe Werkzeuge, bei-

38 Die Unterscheidung in linke und rechte Seite (*leftExpression*, *rightExpression*) wird per Konvention in der entsprechenden Systemkomponente getroffen, ist aber nicht explizit im Datenmodell vorgegeben.

spielsweise für exakte numerische Berechnungen für die Evaluierung des Kontaktsystems benutzt werden. In der vorliegenden Implementierung ist die Klasse *SAEPContactSystemImpl* eine Implementierung des Interfaces und übernimmt die Auswertung von abhängigen Eigenschaften und Bedingungen. Objekte von Klassen die das Interface *ContactSystem* implementieren sind in *ContactSystemLibrary*-Objekten zusammengefasst. Sie dienen zur Verwaltung von Kontaktsystemen. Abbildung 44 zeigt die Klassenstruktur zur Abbildung von Wirkräumen.

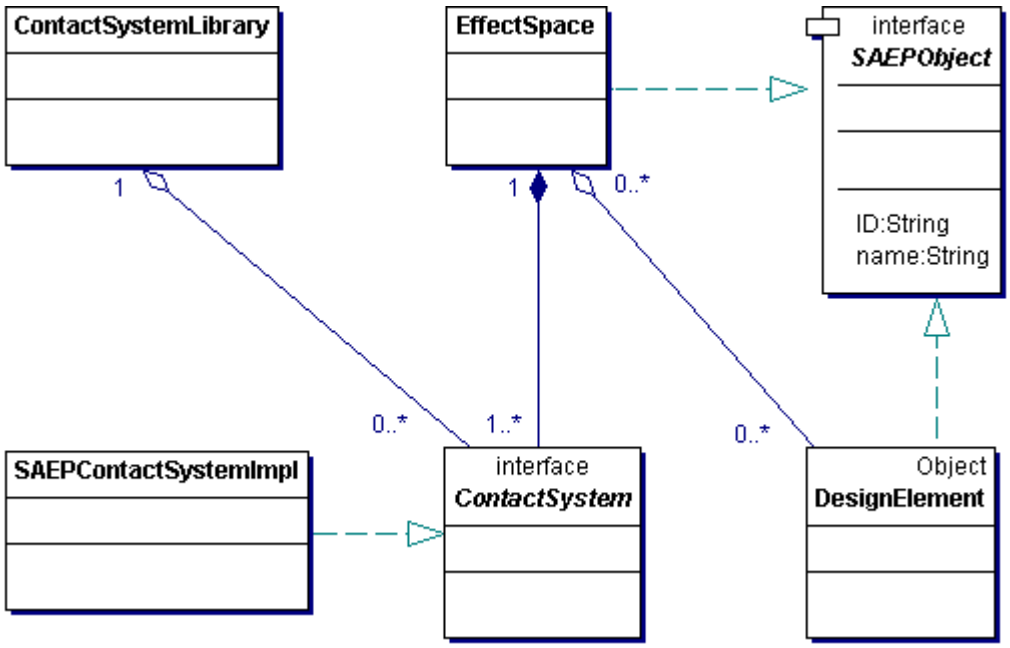


Abbildung 44: Klassenstruktur zur Abbildung von Wirkräumen

Die Klasse *SAEPContactSystemImpl* (siehe Abbildung 45) benutzt zur Abbildung und Evaluierung von mathematischen Abhängigkeiten ein Objekt der Klasse *EquationSystem*. Mit Hilfe der Klasse *CSPort* werden die relevanten Merkmale der in den Wirkräumen enthaltenen *DesignElement*-Objekte in das Gleichungssystem übernommen und an den entsprechenden Stellen der ihm zugrunde liegenden Binärbäume eingefügt.

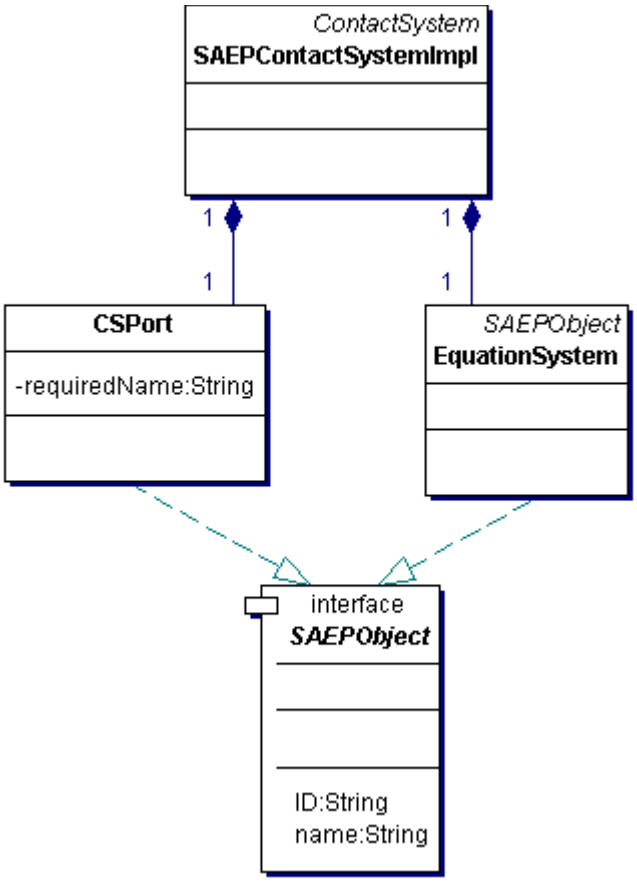


Abbildung 45: Klassenstruktur zur Abbildung von Kontaktsystemen

Durch die Anbindung von identischen Merkmalen in verschiedenen Kontaktsystemen ergeben sich Abhängigkeiten zwischen mehreren Kontaktsystemen und damit ein Supergleichungssystem, das beliebig viele Merkmale aus beliebig vielen *DesignElement*-Objekten verknüpft. Dies funktioniert allerdings nur mit Kontaktsystemen, die durch *SAEPContactSystemImpl*-Objekte beschrieben werden, da explizite Abhängigkeiten aus externen Evaluierungswerkzeugen zur Auswertung von Abhängigkeiten in der vorliegenden Implementierung nicht ermittelt werden. Dessen unbeschadet ermittelt SAEP, welche Merkmale generell in Kontaktsystemen eingebunden sind und kann dem Benutzer entsprechende Hinweise geben.

4.2 Basisklassen für die Darstellung nichtstrenger Hierarchien

Strenge Hierarchien sind hierarchische Strukturen, in denen jeder Knoten maximal einen direkten übergeordneten Knoten (Elternknoten) haben darf. Im Gegensatz dazu erlauben nichtstrenge Hierarchien, dass ein Knoten mehrere Elternknoten haben darf. Der Implementierung liegen einige Basisklassen zugrunde, die grundlegende Funktionalität für den Aufbau von strengen und nichtstrengen, hierarchischen Strukturen bereitstellen, die in vielen Bereichen

benötigt wird. Basisklasse ist *AbstractTreeModel*, die das Java-Swing-Interface[39] *TreeModel* implementiert. Die durch diese Klassen dargestellten Hierarchien können dank der Implementierung des *TreeModel*-Interfaces direkt mit Hilfe spezieller, im Java-SDK mitgelieferter Klassen visualisiert werden. *AbstractTreeModel* stellt Funktionalität für den Aufbau von Hierarchien bereit. Von ihr abgeleitete Klassen müssen lediglich die Methode *setChildren(SAEPNode parentNode)* zur Ermittlung der Kindknoten eines Knotens implementieren, die abhängig von den Konstrukten des betrachteten Datenmodellaspekts ist. Die Objekte des Datenmodells, die in der Hierarchie eingeordnet sind, werden in Objekten der Klasse *SAEPNode*, die die Knoten des Hierarchiebaums repräsentieren, als sogenanntes *User-Object* gekapselt und können so direkt angesprochen werden. Abbildung 46 zeigt die Klassen zur Abbildung nicht-strenger Hierarchien.

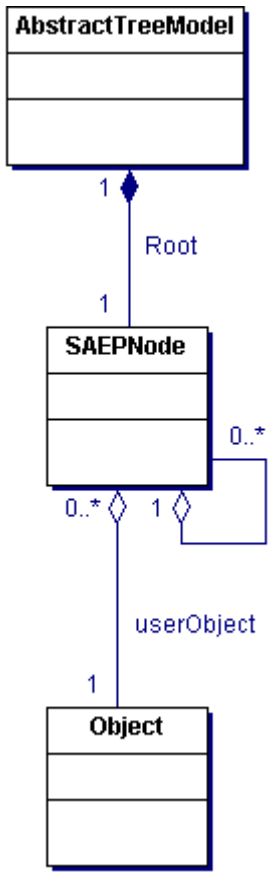


Abbildung 46: Klassenstruktur zur Abbildung nicht-strenger Hierarchien

Hierbei ist zu beachten, dass die Struktur, die durch die Objekte der Klasse *SAEPNode* aufgebaut wird, sehr wohl streng hierarchisch ist. Dadurch aber, dass Objekte mehreren *SAEPNode*-Objekten (und somit mehreren „Elternknoten“) zugeordnet werden können ergibt sich dennoch eine Abbildung einer nicht-strengen Hierarchie.

---

39 Java Swing ist ein umfassende Sammlung von Klassen zur Programmierung graphischer Benutzerschnittstellen (GUI, Graphical User Interface), auch Benutzungsoberfläche genannt [Schi-2002].

4.3 Persistenzschicht

Die Persistenzschicht kapselt die Komponenten, die für eine dauerhafte Speicherung der Daten und den Zugriff auf sie zuständig sind. Dazu gehört zum einen der Zugriff auf die in SAEP selbst erzeugten Daten und zum anderen der Zugriff auf Daten aus externen Autorensystemen wie CAD-Systeme oder SE-Werkzeuge.

4.3.1 Persistierung der in SAEP erzeugten Informationen

Im Rahmen der Implementierung wurde ein flexibles, objektorientiertes Datenbankmanagementsystem realisiert. Es basiert auf einem relationalen Datenbankschema, in das mit Hilfe einer Mappingkomponente beliebig komplexe Objektstrukturen abgebildet werden können. Dazu müssen die abzubildenden Objekte lediglich einigen, wenigen Konventionen entsprechen:

- Es muss ein leerer Konstruktor, das heißt ein Konstruktor ohne Parameter für jede zu persistierende Klasse definiert sein. Das bedeutet, dass ein Objekt für seine Erzeugung zunächst keine weiteren Informationen benötigen darf.
- Objektattribute müssen mit einem kleinen Buchstaben beginnen (z.B. „*objectName*“, „*id*“, „*myAttribute*“).
- Der Zugriff auf die Attribute muss über Lese- und Schreibmethoden erfolgen, die der Namenskonvention *get<AttributeName>* für lesenden Zugriff, bzw. *set<AttributeName>* für schreibenden Zugriff folgen, z.B. „*getObjectName()*“, „*setObjectName()*“, „*getId()*“, „*setId()*“, „*getMyAttribute()*“ und „*setMyAttribute()*“.

Sind diese Voraussetzungen erfüllt, kann das System durch die Java-eigenen Mechanismen der Introspektion[40] zur Laufzeit alle Attribute eines Objekts ermitteln und rekursiv auch komplexe Objektstrukturen persistent machen. Umgekehrt können wiederum beliebig komplexe Objektstrukturen aus dem Inhalt der Datenbank erzeugt werden. Die Flexibilität des Ansatzes erlaubt es, das Datenmodell von SAEP mit Hilfe marktgängiger Java-Modellierungswerkzeuge, wie z.B. *Rational Rose*[41] oder *Together* [42] zu erweitern. Werden die oben genannten Bedingungen von den neu erzeugten Klassen erfüllt, können von ihnen erzeugte Objekte direkt und ohne Veränderungen an der Implementierung persistiert werden. Dies kann sogar zur Laufzeit erfolgen, da es Java erlaubt, neu erzeugte Klassen mit Hilfe des Java Class Loaders, der ein Bestandteil der Java Laufzeitumgebung ist, in die Laufzeitumgebung zu laden[43].

---

40 Introspektion bedeutet, dass Objekte zur Laufzeit eines Programms auf Attribute, Methoden, Superklassen etc. hin untersucht werden können.

41 http://www-306.ibm.com/software/rational/

42 http://www.borland.com/together/

43 Siehe auch „http://java.sun.com/j2se/1.4.2/docs/api/java/lang/ClassLoader.html“

Für die Persistierung wird die Tatsache zunutze gemacht, dass komplexe Objektstrukturen hierarchisch strukturiert sind. Das bedeutet, dass man ausgehend von einem bestimmten Objekt alle enthaltenen Objekte erreichen kann und von diesen ausgehend deren enthaltene Objekte und so weiter. Abbruchkriterium ist entweder das Fehlen weiterer Objekte im betrachteten Objekt oder es ist ein Objekt referenziert, das bereits „besucht“ also analysiert wurde. In diesem Fall wird ebenfalls eine Referenz eingefügt und der Vorgang für dieses Objekt abgeschlossen. In Abbildung 47 ist als Beispiel ein Ausschnitt aus dem Aggregationsbaum eines *View*-Objekts dargestellt. Man erkennt ausgehend vom Objekt *View1* die verschiedenen Aggregationsebenen 0,1, 2 und 3. Das Objekt *PCR1* verknüpft das *DesignElement*-Objekt *DesignElement1* und das *View*-Objekt *View1*. Hierbei tritt eine Rekursion auf, da *PCR1* bereits in *View1* enthalten ist. Der Persistierungsalgorithmus beendet an dieser Stelle den Ast, da er das *View*-Objekt bereits besucht hat und fügt lediglich eine Referenz ein. Das gleiche gilt für das zweite Vorkommen des *DesignElement*-Objekts *DesignElement1* im Objekt *PCR2*.

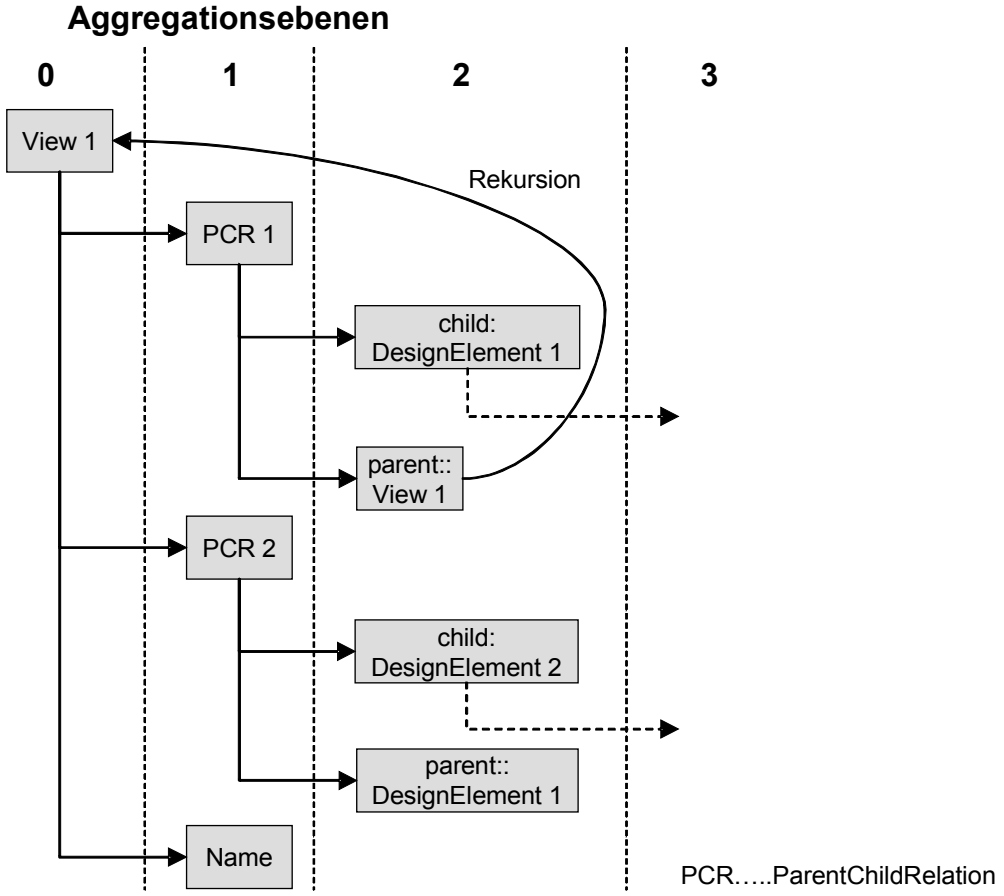


Abbildung 47: Beispiel eines Aggreationsbaums

Soll ein Objekt gespeichert werden, wird der Aggregationsbaum Stufe für Stufe abgewandert und die auf der aktuellen Ebene vorhandenen Objekte in die Datenbank geschrieben. Dabei wird für jedes Objekt eine eindeutige ID (*id*), seine Bezeichnung (*name*, Attributname im enthaltenden Objekt), die ID des enthaltenden Objekts (*parent_id*), optional ein Wert (*value*), der

belegt wird, wenn es sich um ein atomares Attribut handelt (String, Integer, Double etc.) sowie der voll-qualifizierte[44] Klassenname (*type*) des Objekts abgespeichert.

Der Vorgang des Ladens einer Objektstruktur geschieht zweistufig: Zunächst wird als Zwischenergebnis ein Aggregationsbaum (*PersistenceTreeModel*) aus Helferobjekten (*PersistenceNode*) aufgebaut, die die zunächst noch unabhängigen Objekte beinhalten, sowie zusätzlich Informationen über Attributnamen (*attributeName*) und die Datenbank-interne ID des Objekts tragen (*objectId*). Im zweiten Schritt wird der Baum aus den Helferobjekten durchlaufen und die entsprechenden Verknüpfungen in den Objekten erstellt. Abbildung 48 zeigt die Klassenstruktur zur Abbildung von Objektstrukturen.

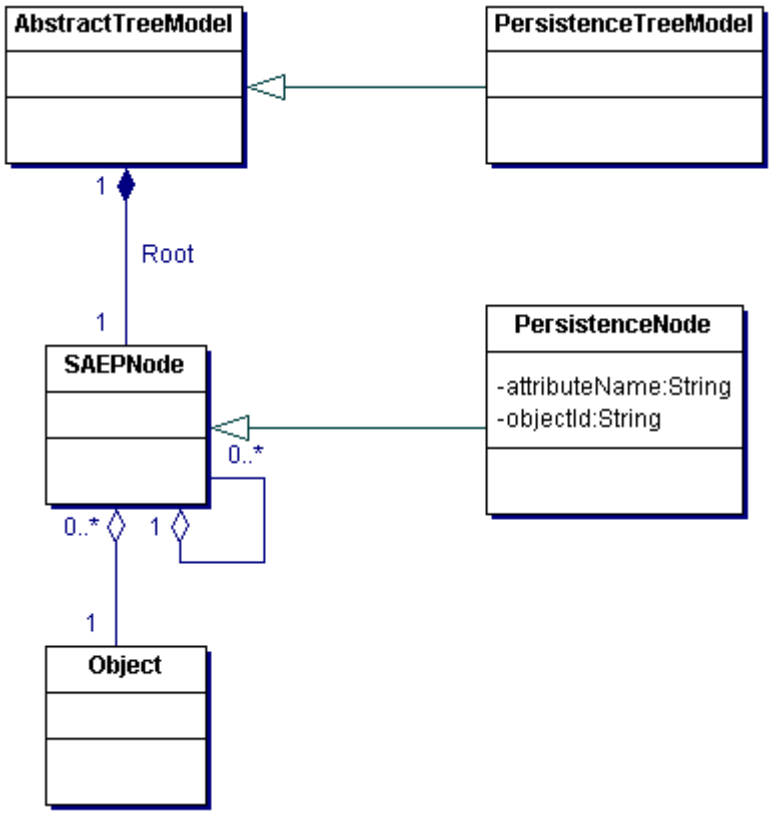


Abbildung 48: Klassenstruktur zur Abbildung von Aggregationsbäumen von Objektstrukturen

Soll ein Objekt geladen werden, wird also zunächst nach dem Objekt in der Datenbank gesucht und seine ID ermittelt. Danach wird der voll-qualifizierte Klassenname ausgelesen und ein Objekt der entsprechenden Klasse erzeugt. Anschließend wird es in ein Objekt der Klasse *PersistenceNode* verpackt (ge-*wrapped*), in das auch der Attributname sowie die Datenbank-interne ID des Objekts eingetragen wird. Danach wird nach allen Objekten gesucht, die die ID des betrachteten Objekts als *parent_id* besitzen und diese nach der oben beschriebenen Vorgehensweise erzeugt und ihrerseits in *PersistenceNode*-Objekte verpackt. Die neuen *PersistenceNode*-Objekte werden als Kinder des *PersistenceNode*-Objekts eigetragen, dass das enthaltende Objekt beinhaltet. Diese Vorgang wird solange ausgeführt bis keine Daten-

44 Voll-qualifiziert bedeutet Klassenname inklusive Package-Angabe (z.B. saep.model.attribute.Attribute)

sätze mehr gefunden werden, die als *parent_id* die ID eines betrachteten Objekts besitzen. Abbildung 49 zeigt schematisch die zweistufige Vorgehensweise

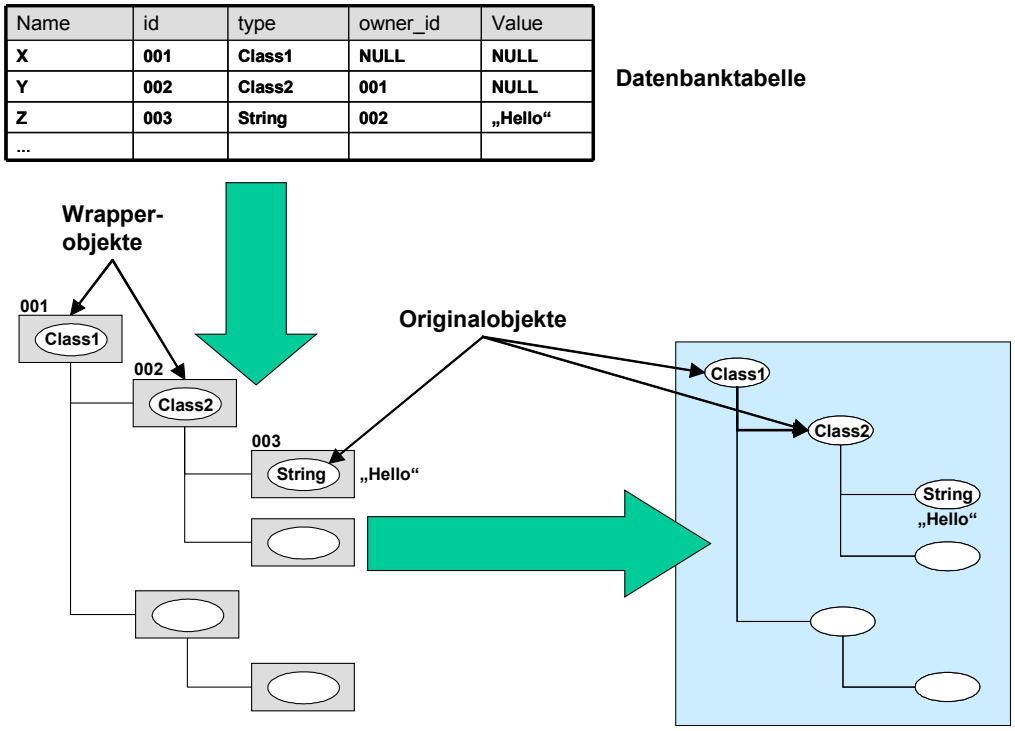


Abbildung 49: 2-stufige Vorgehensweise bei der Instanziierung von Objektstrukturen aus Datenbankrepräsentationen

Durch die Verwendung von entsprechenden Interfaces, die die notwendigen Methoden für den Datenzugriff definieren, können verschiedenste Ansätze zur Speicherung von Daten realisiert werden. Im Rahmen der Implementierung von SAEP wurde neben der Anbindung einer relationalen Datenbank auch eine auf XML[45]-Dateien basierende Datenspeicherung prototypenhaft implementiert. Abbildung 50 zeigt schematisch den Aufbau der Persistenzschicht von SAEP.

45 XML: eXtensible Markup Language: Flexibles Datenaustauschformat.

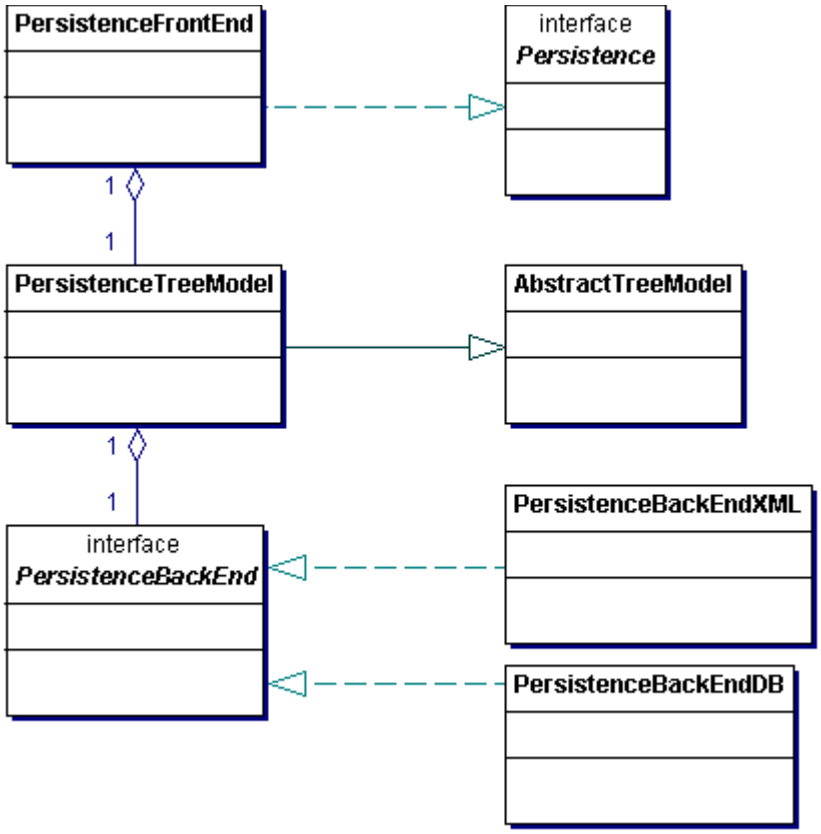


Abbildung 50: Aufbau der Persistenzschicht

Die Klasse *PersistenceFrontEnd* stellt die oberste Schnittstelle zur persistenten Speicherung von Objekten dar. Die Klasse implementiert das Interface *Persistence*, was einen einheitlichen Zugriff für darüber liegende Komponenten auf entsprechende Funktionalität zur Verfügung stellt. In der vorliegenden Implementierung zur Speicherung von Objekten sind in der Klasse *PersistenceTreeModel* die erforderlichen Methoden für die Introspektion der abzuspeichernden, bzw. zu aktualisierenden Objekte bei schreibendem Zugriff sowie zur Zusammensetzung komplexer Objekte bei lesendem Zugriff implementiert. Die einzelnen Objekte, die in *PersistenceTreeModel* zusammengesetzt werden, werden in Implementierungen des Interfaces *PersistenceBackEnd* aus Daten aus den zugrunde liegenden Datenquellen (z.B. relationale Datenbank, XML-Datei) zusammengesetzt.

4.3.2 Zugriff auf externe Autorensysteme

Der Zugriff auf externe Autorensysteme erfolgt ebenfalls über standardisierte Schnittstellen. Dazu wird vorausgesetzt, dass die externen Produktmodelle ebenfalls einen Aggregationsansatz verfolgen, d.h. dass es Modellentitäten gibt, die aus weiteren Entitäten zusammengesetzt (aggregiert) sein können und Merkmale besitzen, die abgefragt werden können. Dabei ist es prinzipiell gleichgültig, ob die Daten aus den externen Systemen in Dateien vorliegen oder ob sie online von einem laufenden System abgefragt werden. Dieses Verhalten wird in den zuständigen Klassen gekapselt und ist für darüber liegende Komponenten transparent. Des Wei-

teren müssen die Modellelemente über eindeutige Bezeichnungen identifizierbar sein (IDs). Diese sollten persistent im angebundenen System vergeben sein. Im Rahmen der vorliegenden Implementierung wurde der Zugriff auf CAD-Systeme mittels der von der OMG standardisierten *CADServices*-Schnittstelle realisiert. Als SE-Werkzeug wurde das System RODON der Firma R.O.S.E Informatik über eine Dateischnittstelle angebunden.

4.3.2.1 Die CADServices-Schnittstelle

Die neutrale CAD-Schnittstelle *CAD-Services* wird von der Manufacturing Technology & Industrial Systems Task Force (ManTIS) [Mant-2005] der Object Management Group (OMG) definiert [OMG-2005a]. Ziel ist es, eine standardisierte Schnittstelle für einen gleichförmigen Zugriff auf B-Rep-basierte CAD-Systeme zu definieren (siehe Abbildung 51). CADServices baut auf dem CORBA-Standard auf, mit dessen Hilfe CAD- und andere Systeme zur rechnerunterstützten Konstruktion in einer verteilten Umgebung online miteinander verbunden werden können [OMG-2005b].

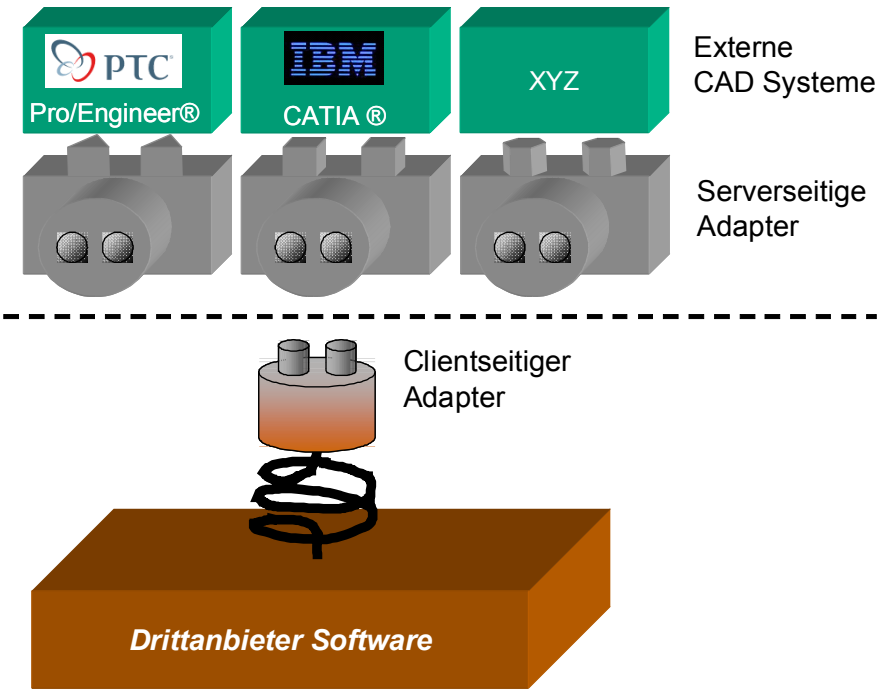


Abbildung 51: Schematische Darstellung der CADServices als neutrale CAD-Schnittstelle

Die Schnittstelle stellt vor allem geometrische und topologische Informationen und deren Aggregation in Features aus CAD-Modellen für andere Systeme bereit. Durch die Definition von abstrakten Objektstrukturen und Methoden in der CADServices Schnittstelle wird für den Anwender[46] der Umgang mit systemspezifischen Datenstrukturen des angebundenen CAD-

46 Ein Anwender der CADServices-Schnittstelle ist in diesem Zusammenhang ein Programmierer, der von außen auf CAD-System-interne Daten über die Schnittstelle zugreifen möchte.

Systems weitestgehend vermieden. Die Abfragen verwenden, soweit möglich, die natürliche Implementierung der jeweiligen Funktionen im geometrischen Kern des CAD-Systems, d.h. die Methoden der Schnittstelle greifen auf das CAD-Modell durch die API[47] des CAD-Systems zu [OMG-2005c]. Das CAD-System übernimmt die Rolle eines Servers, der von anderen Systemen als Clients abgefragt wird, bzw. Kommandos von ihnen entgegennimmt.

Die Funktionalität der CAD Services ist in sieben weitestgehend voneinander unabhängigen Module bzw. Pakete aufgeteilt. Jedes Modul definiert Teilmengen der CADServices-Schnittstelle, die zu einem übergeordneten Funktionsbereich gehören, z.B. Geometrie, Topologie, Features, etc.

Das Modul *CadConnection* definiert die Methoden für das Verbinden mit einem CAD Services Server. Die Methoden von CadConnection gewähren den Zugriff auf die Klasse *Model* des Pakets *CadMain*, die auch eine Unterstützung für Baugruppen bietet. Das Modul *CadFoundation* definiert die allgemeine, übergeordnete Klasse *Entity* und dazugehörige Datenstrukturen. Diese Klasse abstrahiert alle geometrischen und topologischen Modell-Elemente sowie Form-Features. Das Paket *CadGeometry* definiert Methoden und Klassen für die geometrischen Elemente *Oberfläche* und *Kurve*, sowie die Datenstrukturen für die Tesselierung[48] der geometrischen Elemente. Das Modul *CadBrep* definiert alle Klassen, die für die topologische Repräsentation des Volumenmodells notwendig sind. Methoden und Klassen für parametrische Form-Feature und der dazugehörigen Parameter sind im Modul *CadFeature* enthalten. Allgemeine Datenstrukturen und Klassen, die aus allen Modulen referenziert werden, sowie Methoden und Klassen für die Fehlerbehandlung sind im Umfang des Modul *CadUtility* definiert.

CADServices ermöglicht einen bidirektionalen Zugriff auf das CAD-System. Es kann somit nicht nur lesend auf CAD-Modelle zugegriffen werden, sondern interaktiv mit dem System gearbeitet werden. So können neue Modelle auf einem CAD-Server erstellt und mit Hilfe von definierten Methoden geometrisch-topologische Elemente in diesen Modellen erzeugt werden. Bestehende Modelle können verändert, Feature-Parameter variiert und anschließend die Modelle regeneriert, dargestellt oder gespeichert werden.

Im Rahmen der vorliegenden Implementierung wird die CADServices-Schnittstelle für das CAD-System Pro/ENGINEER verwendet, die von der Firma Incentrix GmbH [Ince-2005] entwickelt wird.

---

47 API, Application Programming Interface. Allgemein: Programmierschnittstelle von Software

48 *Tesselierung*: Zerlegung von komplexen Flächen in primitive Flächen, meistens Dreiecke (Triangulierung)

4.4 Integrationsschicht

Objekte, die in SAEP bearbeitet werden, können nach der Herkunft der in ihnen enthaltenen Informationen unterschieden werden. Zum einen werden Informationen in SAEP selbst erzeugt, manipuliert und verwaltet (SAEP ist das Autorensystem dieser Informationen), zum anderen werden Informationen in externen Autorensystemen erzeugt und mit Hilfe geeigneter, SAEP-eigener Objekte repräsentiert und dargestellt. Es findet also eine Abbildung der externen Informationen in Elemente des SAEP-eigenen Datenmodells statt.

Zentrale Komponente der Integrationsschicht ist der *Design Element Manager*. Er stellt die Abstraktionsschicht für den transparenten Zugriff auf alle Produktstrukturkomponenten dar, die durch Objekte der Klasse *DesignElement* (und der von ihr abgeleiteten Klassen) repräsentiert werden, sowie auf alle mit diesen verknüpften Objekten. Voraussetzung dafür ist, dass die Objekte aus den angebundenen Systemen persistente Identifikationsattribute (IDs) haben, damit sitzungsübergreifend auf diese zugegriffen werden kann. Der *Design Element Manager* bedient sich für den Zugriff auf externe Systeme der Komponente *PersistenceRegistry*, die den Zugriff auf externe Systeme regelt. Die in SAEP erzeugten Informationen werden mit Hilfe von Klassen persistent gemacht, die das Interface *Persistence* implementieren. Auf extern erzeugte Informationen wird mittels Objekten zugegriffen, deren Klassen das Interface *SystemConnector* implementieren. Die Interfaces erwarten Implementierungen des Interfaces *Initialiser*, die für die Initialisierung der Konnektoren, bzw. der Persistenzschicht zuständig sind. Der *Design Element Manager* bedient sich zudem einer Helferklasse mit der Bezeichnung *EntityMapper*. Sie ist für die Erkennung von gleichartigen Modellentitäten beliebiger Klase zuständig. Abbildung 52 zeigt die Komponenten des *Design Element Managers*.

Darüber hinaus stellt der *Design Element Manager* die Funktionalität für die Verwaltung, Instanziierung und Manipulation von Objekten der Klasse *DesignElement* und sämtlichen von ihr abgeleiteten Klassen bereit.

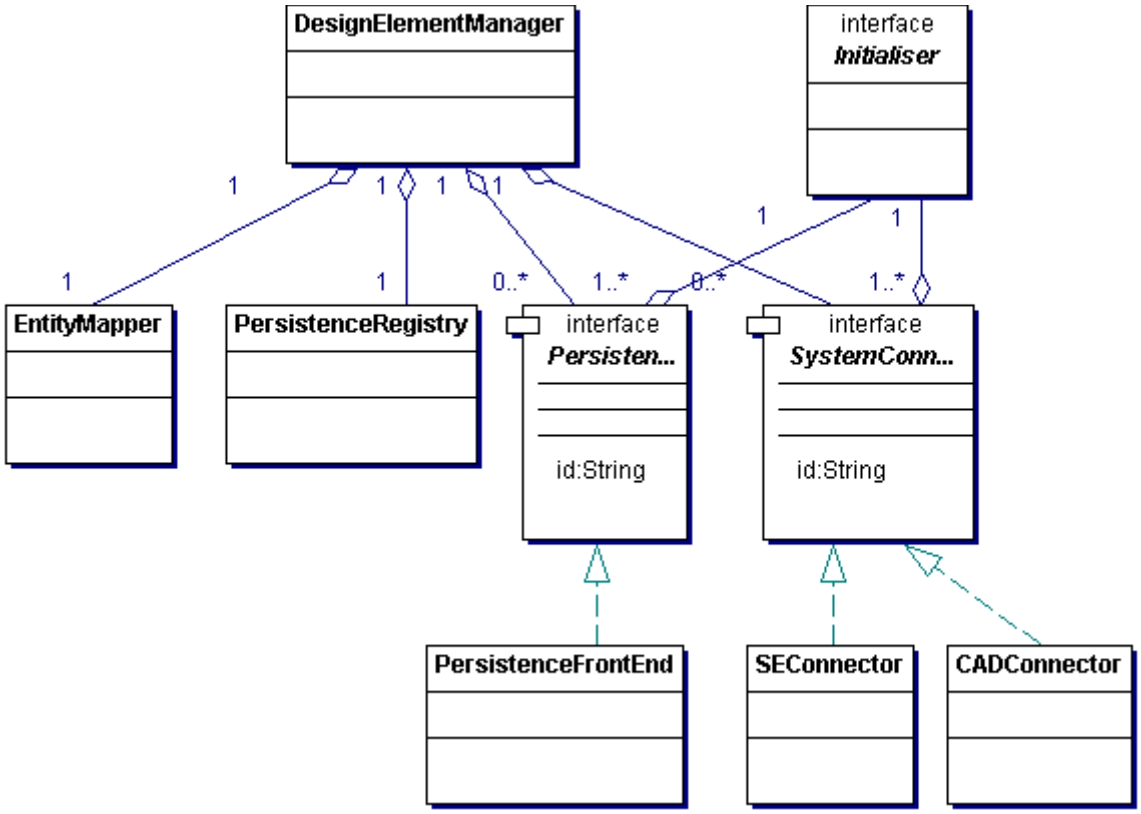


Abbildung 52: Klassenstruktur des Design Element Managers

4.4.1 Initialisierung der Integrationsschicht

Für die Initialisierung der Integrationsschicht müssen als erstes die Quellsystemkonnektoren (Objekte von Klassen, die das Interface *SystemConnector* implementieren) und die Persistenzschicht (Objekte von Klassen, die das Interface *Persistence* implementieren) erzeugt und initialisiert werden. Dazu wird von dem für jede Konnektorklasse und jede Persistenzklasse implementierten *Initialiser*-Objekt ein zentrales XML-Skript ausgelesen, aus dem der Konnektor, bzw. das Persistenzobjekt initialisiert wird[49]. Für dateibasierte Konnektoren sind in dem Skript beispielsweise das Wurzelverzeichnis für auszulesende Dateien vermerkt sowie deren Dateiendungen, während für einen CORBA-basierten Konnektor IP-Adresse und Portnummer des Nameservers sowie der Dienstname des betreffenden Konnektors abgelegt ist. Des Weiteren ist für jede Datenquelle eine eindeutige ID vermerkt, die für die Zuordnung des korrekten Konnektors für ein vorgegebenes Objekt benötigt wird. Die Initialisierung der Persistenzschicht erfolgt analog (siehe auch 4.2). Hier muss die IP-Nummer, bzw. der DNS-Name des Datenbankservers, die Portnummer unter der der Server auf Anfragen wartet, der Datenbankname sowie Benutzername und Kennwort[50] angegeben werden.

---

[49] Durch die Architektur ist nicht vorgegeben, in welcher Form als *Initialiser* deklarierte Objekte ihre Informationen abspeichern und auslesen. In der vorliegenden Implementierung wurde dies mit XML-Skripten realisiert.

[50] In der vorliegenden Implementierung wurde auf eine Benutzerauthentifikation verzichtet. Diese kann aber leicht nachträglich durch geeignete Mechanismen, z.B. über ein entsprechendes Element der Benutzungsoberfläche implementiert werden.

Die *PersistenceRegistry* verknüpft die Konnektoren-IDs mit Referenzen zu den Konnektoren zu den externen Systemen, in denen die Objekte persistent gehalten werden. Das bedeutet, dass für jedes Element ein Eintrag mit einer eindeutigen Kennung des Konnektors zum jeweiligen externen System vorhanden ist. Diese ID ist auch in Objekten aller Klassen abgelegt, die das Interface *ExternalObject* implementieren (*serviceID*). Damit kann bei Bedarf direkt auf den richtigen Konnektor für das Laden oder Speichern von extern gehaltenen Objekten zugegriffen werden.

4.4.2 Laden und Speichern von Modellen

Zu Beginn einer Sitzung muss das integrierte Datenmodell initialisiert werden, d.h. alle Referenzen auf IDs werden aufgelöst und die entsprechenden Objekte aus den angebundenen Systemen erzeugt und miteinander verknüpft. Dieser Vorgang muss in der vorliegenden Implementierung bei jeder Änderung an Objekten in den externen Systemen wiederholt werden, um die Konsistenz und Korrektheit des Modells zu gewährleisten. Die Objekte aus den externen Systemen werden mit Hilfe von Objekten der SAEP-eigenen Klasse *ConnectedDesignElement* repräsentiert (siehe auch 4.1.1, S. 81), die in der Persistenzschicht von SAEP sitzungsübergreifend gespeichert werden. *ConnectedDesignElement*-Objekte dienen primär zur Verwaltung unterschiedlicher Identifikationssysteme. Zum einen haben diese Objekte durch die Implementierung des Interfaces *SAEPObject* eine eindeutige ID innerhalb der SAEP-Persistenzschicht und zum anderen referenzieren sie zusätzlich eine im Ursprungssystem des repräsentierten Objekts eindeutige ID (*externalID*) durch die Implementierung des Interfaces *ExternalObject*.

Nach Auswahl eines *DesignElement*-Objekts, das geladen werden soll, werden die entsprechenden *ParentChildRelation*-Objekte ermittelt und so sukzessive der Aggregationsbaum aufgebaut, wie in Abschnitt 4.3.1 beschrieben. Für jedes Produktstrukturelement werden die zugehörigen, in der Persistenzschicht abgelegten, aggregierten Objekte geladen. Die in diesem Stadium sind die erzeugten *ConnectedDesignElement*-Objekte sind mit den in der Datenbank hinterlegten Daten der letzten Aktualisierung aus den externen Systemen belegt. Das gleiche gilt für die *ExternalAttribute*-Objekte und die dazugehörigen aggregierten Objekte.

Ist die Baumstruktur aufgebaut, wird für jedes erzeugte Repräsentantenobjekt in der *PersistenceRegistry* mittels der in ihnen hinterlegten *serviceID* die Referenz auf den entsprechenden Quellsystemkonnektor ermittelt und aus diesem mittels der *externalID* die aktuellen Daten für das Objekt abgefragt. In der *PersistenceRegistry* sind dazu den SAEP-IDs der Objekte Kennungen (ID) in Form von Zeichenketten zur Ermittlung der korrekten Quellsysteme zugeordnet.

Gespeichert werden in der vorliegenden Implementierung lediglich die Informationen, die in der Persistenzschicht abgelegt werden. Es findet keine Modifikation von Modellentitäten in externen Autorensystemen statt. Es wird, mit Ausnahme atomarer Werte, von denen keine weiteren Verknüpfungen ausgehen, grundsätzlich der gesamte Modellinhalt gespeichert, da noch keine Mechanismen zur Aktualisierung von Teilaspekten des Modells implementiert

sind. Die Implementierung dieser Mechanismen erwies sich als sehr aufwendig, da die Konsistenzsicherung umfangreicher, aggregierter Modelle sehr komplex ist.

4.4.3 Verknüpfung korrespondierender Merkmale

Es werden zwei verschiedene Möglichkeiten korrespondierender Merkmale in dieser Arbeit betrachtet: zum einen kann es Produkteigenschaften gleicher Bedeutung aber unterschiedlicher Herkunft geben, was das Autorensystem betrifft. Dies ist zum Beispiel der Fall, wenn in einem CAD-System und in einem separaten DMU oder Simulationswerkzeug das gleiche Bauteil, bzw. die gleiche Baugruppe betrachtet wird. Beides sind geometrische Modelle und beschreiben zumindest teilweise den gleichen Aspekt, das heißt, es gibt eine Schnittmenge ihrer Modellinhalte. Zum anderen müssen im Zuge einer anforderungsgetriebenen Produktmodellierung die Produktanforderungen mit den entsprechenden Produkteigenschaften verknüpft werden, um die gefundene Konstruktions- oder Entwicklungslösung auf Konformität mit den Anforderungen zu überprüfen.

Die Verknüpfung korrespondierender Merkmale birgt verschiedene Herausforderungen, wie in Abschnitt 3.2.6 beschrieben. Es müssen korrespondierende Produktkomponenten sowie deren zugeordnete Merkmale miteinander verknüpft werden.

4.4.3.1 Manueller Ansatz zur Verknüpfung von Modellentitäten

Da die Modellinhalte externer Systeme sowie SAEP-eigene in einem integrierten Modell abgebildet sind, ist die programmtechnische Umsetzung mit Hilfe entsprechender Benutzungsoberflächen einfach.

Korrespondierende Produktstrukturelemente, die durch Objekte der Klasse *DesignElement* oder einer ihrer Unterklassen repräsentiert werden, können durch den Benutzer bestimmt und mittels Objekten der Klasse *DEConectedDERelation* miteinander verknüpft werden.

Durch die Beziehungsklassen zur Verknüpfung von Merkmalen können diese miteinander verknüpft werden (siehe auch 4.1.5, Seite 86). Die Verknüpfung zweier korrespondierender Produkteigenschaften wird durch Objekte der Klasse *CorrespondingAttributeRelation* dargestellt, während die Verknüpfung einer Produktnforderung mit einer Produkteigenschaft über Objekte der Klasse *RequirementPropertyRelation* realisiert wird.

Die modellierten Verknüpfungen können bei Bedarf auch als Schablonen abgespeichert werden, um sie für spätere Verknüpfungsprozesse wieder zu verwenden.

4.4.3.2 Schablonenbasierter Ansatz zur Verknüpfung von Merkmalen

Durch den Ansatz der Schablonen (*Templates*), die vordefinierte Produktstrukturen bereitstellen, die nur noch ausgeprägt werden müssen, kann der Vorgang der Verknüpfung von Modellentitäten wesentlich beschleunigt werden. Der Ansatz ist prinzipbedingt vor allem bei sich wiederholenden Strukturen nutzbringend. Dazu werden in den Schablonen existierende Referenzen (Objekte der Klasse *ConnectedDesignElement*) eingefügt und mit beliebigen anderen Elementen der Produktstruktur verknüpft. Wie beim manuellen Ansatz werden korrespondierende Elemente mit einem Objekt der Klasse *DEConnectedDERelation* mit einem *DesignElement*-Objekt verknüpft. Soll ein neues Produkt (oder eine Produktkomponente) angelegt werden, so muss lediglich die entsprechende Schablone ausgewählt und instanziiert werden. Produktmerkmale sind ebenfalls bereits in den Schablonen vordefiniert und miteinander verknüpft.

4.4.3.3 Automatisierter Ansatz zur Verknüpfung von Merkmalen

Der automatisierte Ansatz setzt auf einen Vergleich der Bezeichnungen, sowohl bei der Erkennung korrespondierender Produktstrukturelemente als auch bei der Erkennung korrespondierender Merkmale. Der Vergleich beruht auf Regulären Ausdrücken, das heißt, syntaktisch ähnliche Elemente und Merkmale können so identifiziert werden. Bei der Verknüpfung von Merkmalen kann optional auch ein Vergleich über Maßeinheiten erfolgen. Dabei kann optional die Größenordnung der Einheiten berücksichtigt werden.

Die Funktionalität für die Erkennung korrespondierender Elemente und Merkmale wird durch die Klasse *EntityMapper* bereitgestellt. Für die Mustersuche mittels Regulärer Ausdrücke wird das Paket *java.util.regex* benutzt, dass bei neueren Version der Java-Entwicklungsumgebung (SDK) mitgeliefert wird[51]. Der Vergleich von Maßeinheiten erfolgt mit Hilfe des Pakets JScience (siehe auch Abschnitt 4.1.1).

Alle erzeugten Verknüpfungen können mit Hilfe der Persistenzschicht dauerhaft gespeichert werden.

4.5 Anwendungslogik

Die Anwendungslogik besteht im wesentlichen aus einer Komponente zur Verwaltung und Auswertung von Gleichungssystemen, einer Komponente zur Verwaltung und Auswertung von Wirkräumen, einer Komponente zur Verwaltung und Auswertung von Merkmalsbeziehungen sowie einer Komponente zur Verwaltung von Produktmerkmalsbibliotheken und zur Zuordnung von Produktmerkmalen zu Produktstrukturelementen.

---

51 http://java.sun.com/developer/technicalArticles/releases/1.4regex/

Die Komponenten sind einer Komponente zur Kontrolle des Programmablaufs untergeordnet. Diese Komponente wird durch ein Objekt der Klasse *SAEPController* realisiert. Der *SAEP-Controller* nimmt über die Benutzungsschnittstelle die Benutzereingaben entgegen, stellt die Anfragen an die Komponenten der einzelnen Funktionsbereiche zusammen und gibt deren Ausgabe an die Benutzungsoberfläche zurück. Die Benutzungsoberfläche ist ebenfalls in einer aggregierten Komponente zusammengefasst, die von einem Objekt der Klasse *SAEP-GUI* realisiert wird. Abbildung 53 zeigt vereinfacht die Architektur von SAEP

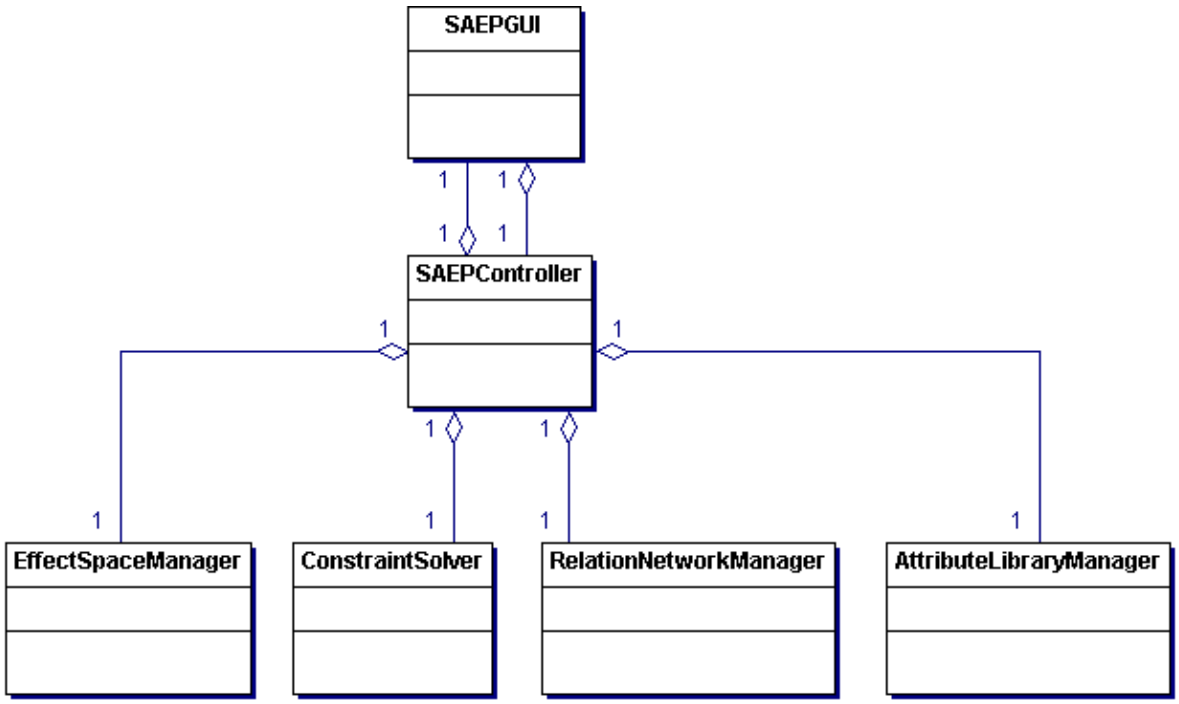


Abbildung 53: Die Architektur von SAEP

4.5.1 Auswertung von mathematischen Gleichungssystemen

Für die mathematische Auswertung von Gleichungssystemen, die durch Objekte der Klasse *EquationSystem* repräsentiert werden, ist die Klasse *ConstraintSolver* zuständig. Mit ihrer Hilfe werden Gleichungssysteme verwaltet und ausgewertet. In der vorliegenden Implementierung wird zur Auswertung von mathematischen Gleichungssystemen das Constraintsolverpaket IASolver[52] von Timothy J. Hickey benutzt, das mit Intervallen arbeitet [HiQE-2001]. Die Funktionalität für den Zugriff auf die IASolver-Infrastruktur ist in der von *ConstraintSolver* abgeleiteten Klasse *IAConstraintSolver* bereitgestellt. Sie ist unter anderem zuständig für die Umsetzung der SAEP-eigenen Repräsentation von Binärbäumen in die IA-Solver-Struktur. Zu diesem Zweck dient die Klasse *IAMathRelationModel*, die von MathRelationModel abgeleitet ist. Durch die Trennung von SAEP-eigener, allgemeiner Funktionalität

52 http://cvs.sourceforge.net/viewcvs.py/jscheme/jscheme/contrib/ia/

zur Verwaltung von Binärbäumen von der eigentlichen Auswertungsfunktionalität können mit dem geringstmöglichen Aufwand auch andere Constraintsolverpakete benutzt werden.

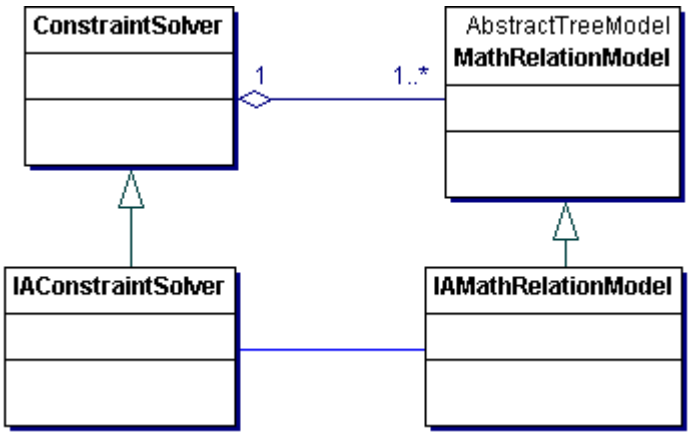


Abbildung 54: Klassenstruktur des Constraintsolvers

Oberklasse aller Ausdrücke in IASolver ist *Exp*. Objekte der Klasse *Var* repräsentieren Variablen. Objekte der Klasse *Num* repräsentieren Konstanten. *BinaryOp*-Objekte repräsentieren binäre Operatoren, wie „+“, „-“ etc. Sie verknüpfen zwei *Exp*-Objekte. Die von *Exp* abgeleiteten Klassen sind aber nicht nur reine Repräsentationen von Gleichungen sondern beinhalten auch Funktionalität für die Lösung von Gleichungen, bzw. Gleichungssystemen. Abbildung 55 zeigt eine vereinfachte Darstellung des Datenmodells von IASolver.

Var- und *Num*-Objekte besitzen jeweils ein Objekt der Klasse *RealInterval*, das wiederum zwei *double*-Werte als Attribute für die Definition der unteren und oberen Grenze des Intervalls besitzt.

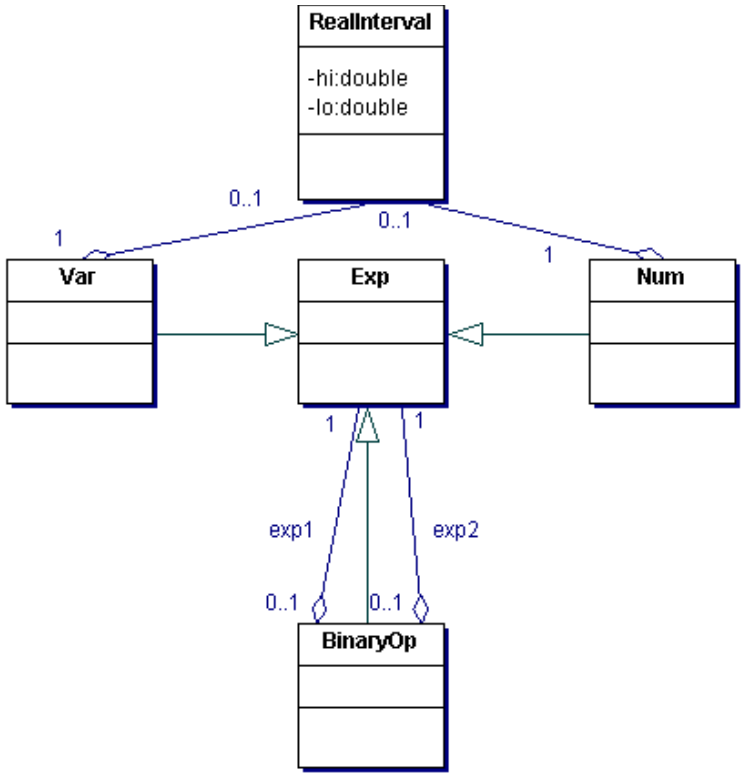


Abbildung 55: Vereinfachte Klassenstruktur von IASolver

Die Objekte zur Repräsentation von mathematischen Gleichungssystemen (*Expression, MathValue* etc. siehe auch 4.1.4, Seite 88) werden nicht direkt manipuliert, sondern in Objekten der Klasse *MathNode* gekapselt. Die Auswertung von Gleichungssystemen erfolgt mit Hilfe der Klasse *MathRelationModel*, die von der Klasse *AbstractTreeModel* abgeleitet ist. *MathRelationModel* ermöglicht es, beliebige, auf Binärbäumen basierende Constraintsolver auf Basis der SAEP-eigenen Binärbaumstruktur zu initialisieren. Dazu muss lediglich der Binärbaum durchwandert und in jedem Knoten den SAEP-Ausdrücken entsprechende Ausdrücke im angebundenen Constraintsolvermodell erzeugt werden. Da der angebundene Constraintsolver IASolver mit Intervallen arbeitet, muss für die Repräsentation von exakten Werten eine Grenze für den Abstand zweier Intervallgrenzen angegeben werden, ab der das Intervall als Einzelwert interpretiert werden soll (*threshold*). Abbildung 56 zeigt die vereinfachte Klassenstruktur zur Kapselung von Binärbäumen zur Darstellung von Ausdrücken und Gleichungen.

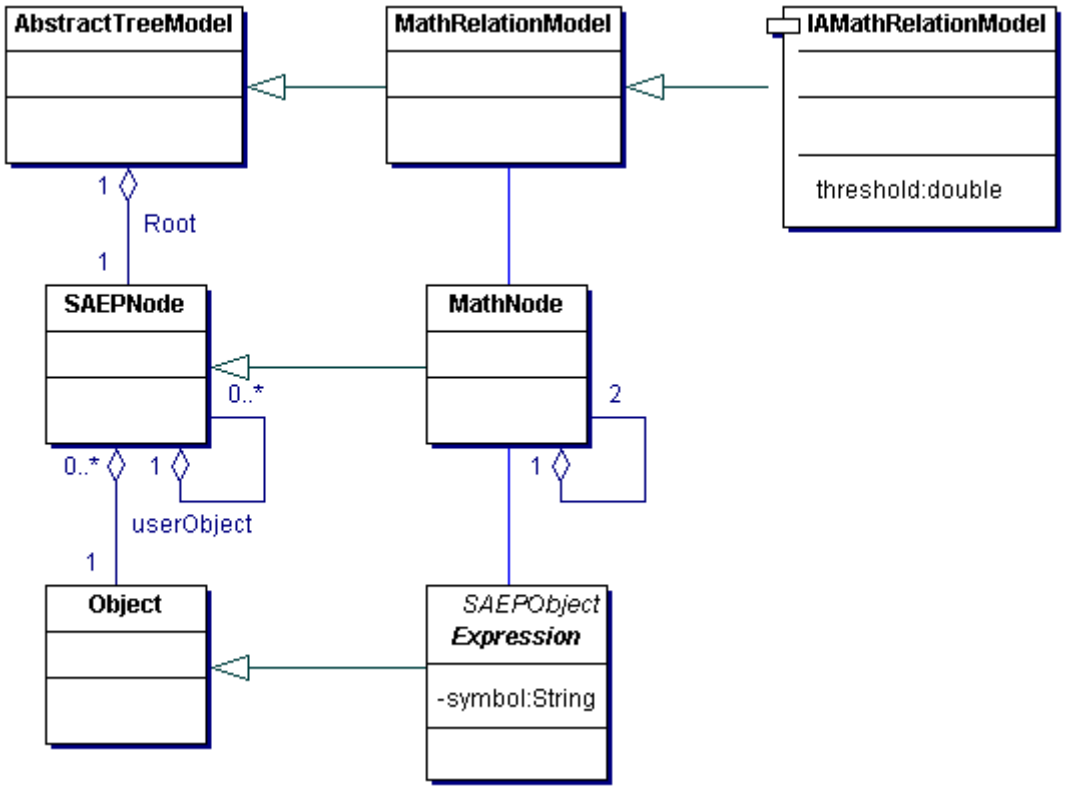


Abbildung 56: Vereinfachte Klassenstruktur zur Kapselung von SAEP-Expression-Bäumen

Expression-Objekte werden in Objekten der Klasse *MathNode* gekapselt, die von der Klasse *SAEPNode* abgeleitet ist. Die verwendbaren Operatoren in SAEP sind vorgegeben und haben jeweils eine Entsprechung im Operatorenvorrat von IASolver, daher können sie direkt aufeinander abgebildet werden. Merkmale müssen dagegen vor einer Übergabe an IASolver angepasst werden. Zum einen müssen eventuell voneinander abweichende Größenordnungen von Einheiten aneinander angepasst werden. Dazu werden sie in SI-Einheiten umgewandelt und die Parameter mit den sich daraus ergebenden Faktoren multipliziert. Die notwendige Funktionalität ist im Paket JScience vorhanden, das in SAEP für die Repräsentation von Werten benutzt wird. Zum anderen muss bei einem Merkmal, das durch einen Einzelwert repräsentiert ist, der Einzelwert jeweils als untere und als obere Grenze eines IASolver-Werteintervalls eingetragen werden.

Jedes *MathRelationModel*-Object repräsentiert eine Gleichung mit einem Vergleichsoperator als Wurzelelement. Die einzelnen *MathRelationModel*-Objekte werden mittels der Klasse *ConstraintSolver* zu einem Gleichungssystem zusammengeschaltet, indem aus den einzelnen MathRelationModell-Binärbäumen entsprechende IASolver-Bäume konstruiert und mit dem Semikolon-Operator von IASolver verknüpft werden.

Ergebnis der Auswertung von Gleichungssystemen sind entweder ausgeprägte Merkmale, oder, wenn das Gleichungssystem überbestimmt ist, eine entsprechende Fehlermeldung des IAConstraintSolvers.

4.5.2 Wirkraummanager (*EffectSpaceManager*)

Das Konzept der Kontaktsysteme ist durch Konstrukte zur dynamischen Anpassung von Gleichungen realisiert. Das bedeutet, ein Kontaktsystem enthält einerseits einen Binärbaum zur Repräsentation der charakteristischen Rumpfgleichung des Kontaktsystems und andererseits enthält es Wissen darüber, wie die Rumpfgleichung bei Hinzufügen weiterer Produktstrukturelemente zum übergeordneten Wirkraum mit deren Merkmalen erweitert, bzw. bei Entfernung von Komponenten reduziert werden muss.

Der Wirkraummanager (in Abbildung 33 als *EffectSpaceManager* bezeichnet) ist für die Verwaltung, Auswertung und Manipulation von Wirkräumen zuständig. Er greift über die Integrationsschicht auf Elemente des oben beschriebenen Datenmodells zu. Die Aufgaben des Wirkraummanagers umfassen im einzelnen:

- Initialisierung des Wirkraummodells
- Auswertung der in den Wirkräumen und Kontaktsystemen abgebildeten Zusammenhänge
- Verwaltung von Wirkräumen und Kontaktsystemen
- Erzeugung und Änderung von Wirkräumen und Kontaktsystemen

Der Wirkraummanager bedient sich zweier Klassen zur Realisierung seiner Funktionalität. Der *Design Element Manager* dient zur Verwaltung und zum Zugriff auf die Elemente der Produktstruktur, die durch Objekte der Klasse *DesignElement* aufgebaut wird, sowie zum Zugriff auf deren Merkmale. Für die Verwaltung, Pflege und Auswertung von Kontaktsystemen ist der *ContactSystemManager* zuständig. Beide Objekte greifen auf eine Implementierung des Interfaces *Persistence* zurück, um die in ihnen verwalteten Informationen dauerhaft zu speichern. Der *Design Element Manager* ist Bestandteil der Integrationsschicht, die das der Implementierung zugrunde liegende Datenmodell verwaltet und einen transparenten Zugriff auf alle Datenmodellelemente bereitstellt. Der Aufbau des Wirkraummanagers ist in Abbildung 57 dargestellt.

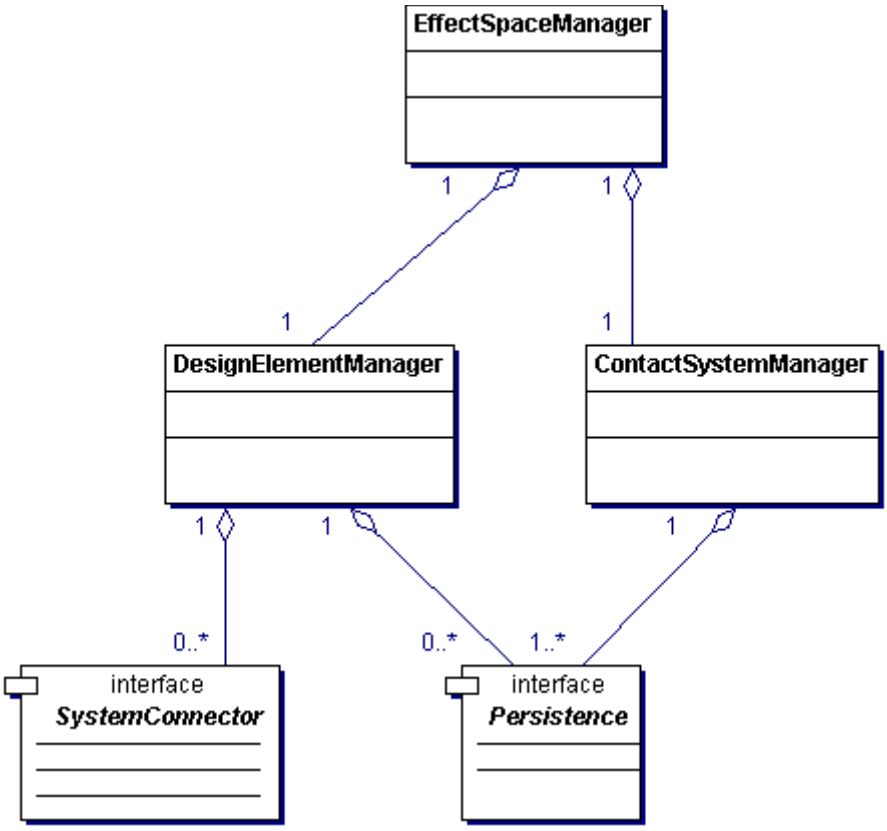


Abbildung 57: Komponenten des Wirkraummanagers

4.5.2.1 Initialisierung des Wirkraummodells

Der Wirkraum-Manager geht bei der Initialisierung von Wirkräumen zweistufig vor: zunächst müssen die relevanten Produktkomponenten ermittelt und den entsprechenden Wirkräumen zugeordnet werden. Dazu werden die Adjazenzbeziehungen ausgewertet, die vom *Design Element Manager* geliefert werden. Sind Komponenten mit mehreren anderen Komponenten über Adjazenzbeziehungen verknüpft, so sind diese als Schnittstellen zwischen verschiedenen Wirkräumen zu behandeln und gegebenenfalls entsprechende Übertragungsfunktionen zu ermitteln, die durch komponentengebundene Kontaktsysteme[53] definiert sind. Wirkräume können zudem zu jedem Zeitpunkt und in jeder Abstraktionsstufe des Konstruktionsprozesses angelegt bzw. modifiziert werden. Aggregate, wie z.B. Baugruppen, SE-Systeme oder abstrakte Aggregate können bereits zu Beginn in Wirkräume eingefügt werden. Werden neue Elemente in diese Aggregate eingefügt, werden sie automatisch zu den sie enthaltenden Wirkräumen hinzugefügt.

Sollen neue Adjazenzbeziehungen erstellt werden, so dienen in der vorliegenden Implementierung *ParentChildRelation*-Objekte als Grundlage, die aggregierte Komponenten (z.B. Baugruppen) definieren. Diese sind von der Sicht (*View*) auf die Produktstruktur abhängig. Sind Unterbaugruppen in einer Baugruppe vorhanden, so werden deren Komponenten rekur-

53 Komponentengebundene Kontaktsysteme verknüpfen die Merkmale einer einzigen Komponente miteinander.

siv in den Wirkraum eingefügt. Durch Interaktion mit dem Benutzer können die nicht-relevanten Komponenten aussortiert werden.

Sind die Wirkräume mit Komponenten versehen, müssen in einem zweiten Schritt die für die Auswertung notwendigen Eigenschaften und Sensibilitäten der Komponenten ermittelt und den Kontaktsystemen zugeordnet werden. Diese Aufgabe übernimmt der *Contact System Manager*. Dieser überprüft zunächst, ob Merkmale mit den von den Kontaktsystemen erwarteten Maßeinheiten Komponenten zugeordnet sind. Dies geschieht in der vorliegenden Implementierung zum einen mit Hilfe einer Analyse der Maßeinheiten, mit denen die einzelnen Merkmale beschrieben sind und zum anderen über einen Vergleich der Merkmalsbezeichnungen mittels Regulärer Ausdrücke. Das Ergebnis dieser Auswertung kann für jede Komponente und jedes Merkmal unterschiedlich ausfallen:

- Eindeutige Zuordnung eines Merkmals: Dies ist entweder der Fall, wenn nur ein einziges Merkmal mit der erwarteten Maßeinheit einer Komponente zugeordnet ist, oder wenn ein einziges Merkmal mit der erwarteten Bezeichnung und der erwarteten Maßeinheit der Komponente zugeordnet ist. In diesem Fall muss der Benutzer lediglich die Zuordnung bestätigen.
- Mehrdeutige Zuordnung von Merkmalen: Mehrere Merkmale mit gleicher oder äquivalenter Maßeinheit sind derselben Komponente zugeordnet. Der Benutzer muss in diesem Fall das korrekte Merkmal selbst auswählen.
- Fehlende Merkmale: Es ist kein Merkmal mit der erwarteten Maßeinheit der Komponente zugeordnet. Der Benutzer muss das fehlende Merkmal spezifizieren.

Die Objekte, durch die eine Gleichung definiert ist (*Expression*, *MathValue* etc.) werden in Objekten der Klasse *CSMathNode* gekapselt, die die Klasse *MathNode* spezialisiert. *CSMathNode*-Objekte tragen die zusätzlichen Boolschen Attribute *isInputDockingNode* und *isOutputDockingNode*. Ist der Wert von *isDockingInputDockingNode* auf *TRUE* gesetzt, bedeutet dies, dass bei einer Erweiterung der Kontaktsystemgleichung (z.B. wenn das Merkmal eines neuen Produktstrukturelements hinzugefügt wird) unterhalb dieses Knotens der neue Knoten mit dem zu verknüpfenden Merkmal eingefügt werden soll. Des Weiteren tragen sie ein Attribut, das definiert, wie die Bezeichnung eines Merkmals sein muss, damit es an dieser Stelle eingefügt werden kann und ein *Unit*-Objekt, das gegebenenfalls die erforderliche Einheit an dieser Stelle festlegt. Abbildung 58 zeigt ein einfaches Beispiel für das Einfügen eines neuen Merkmals in eine bestehende Gleichung. Abbildung 59 zeigt die Klassen zur Abbildung von Kontaktsystemen.

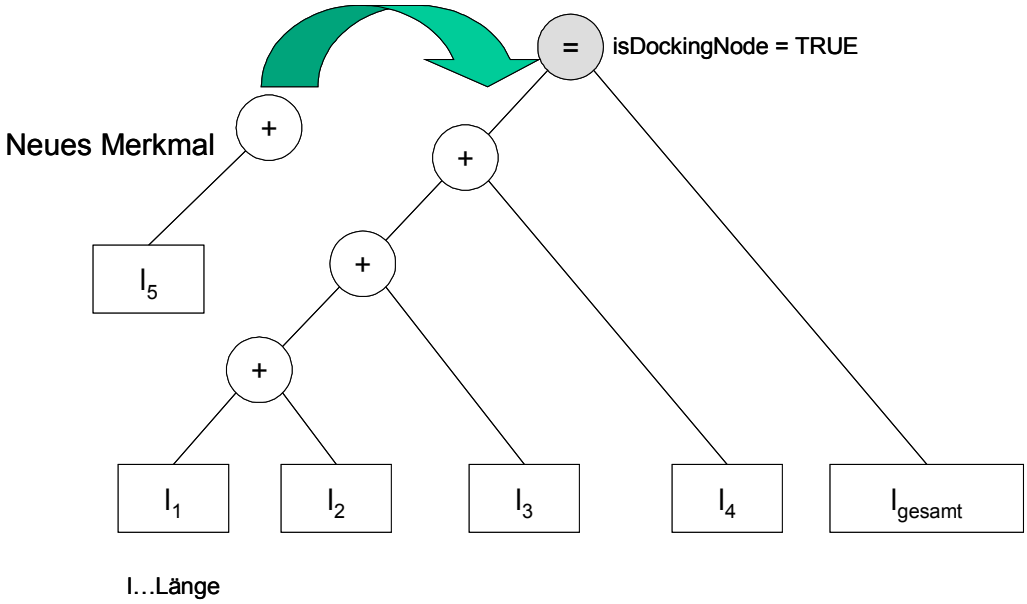


Abbildung 58: Hinzufügen eines neuen Merkmals in eine bestehende Gleichung

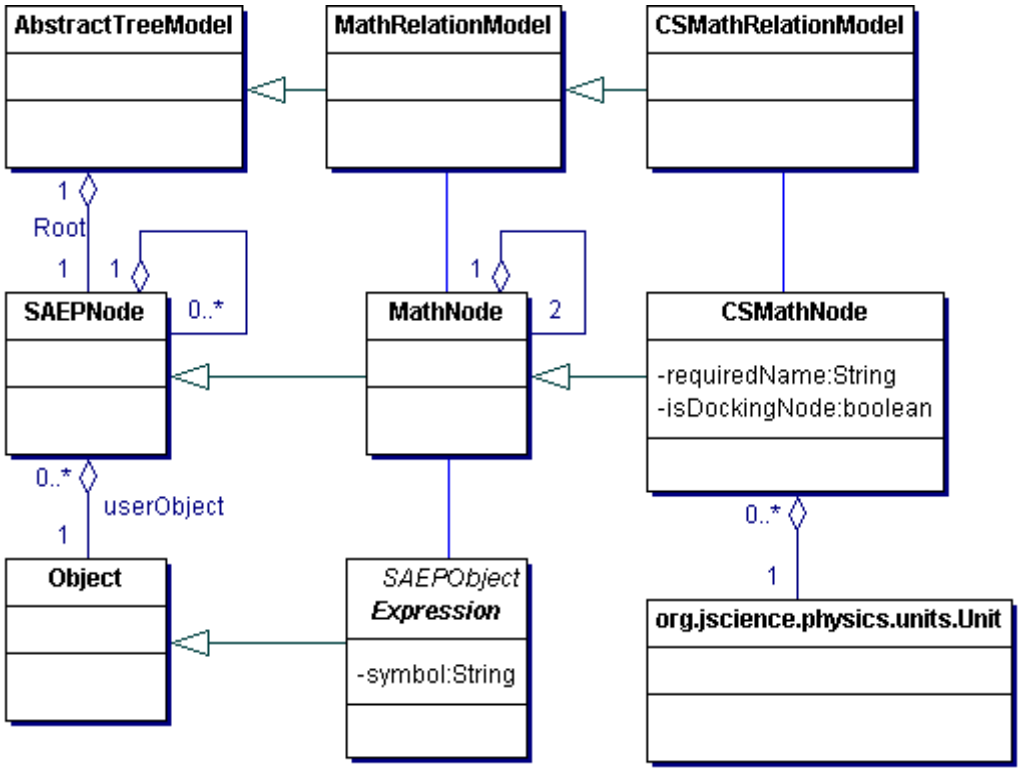


Abbildung 59: Klassenstruktur zur Abbildung und Auswertung von Kontaktsystemen

Der *Contact System Manager* bedient sich Objekten der Klasse *CSPort*, die jedem Kontaktsystem zugeordnet sind, um die richtigen Merkmale an der richtigen Stelle im Binärbaum einzufügen. Sie dienen quasi als Schnittstelle zwischen den im Wirkraum zugeordneten Komponenten und den Kontaktsystemen und definieren, welche Voraussetzungen die Merkmale einer Komponente erfüllen müssen, damit sie in das Kontaktsystem aufgenommen werden und wie, d.h. mit welchem Operator und an welcher Stelle sie in die Gleichung einge-

fügt werden sollen. Sie müssen dazu mit den in den *CSMathNode*-Objekten vorgegebenen Bezeichnungen und, falls spezifiziert, mit den Einheiten übereinstimmen. Die Art der Merkmale wird durch eine optionale Zeichenkette (*requiredName*) und einem *Unit*-Objekt definiert. Die Zeichenkette dient zur Überprüfung, ob die Bezeichnung eines Merkmals mit der in der Zeichenkette geforderten Bezeichnung übereinstimmt. Dazu müssen sie nicht exakt übereinstimmen, sondern durch die Angabe eines Regulären Ausdrucks in der Zeichenkette können auch ähnliche Bezeichnungen berücksichtigt werden. Durch das *Unit*-Objekt kann eine geforderte Maßeinheit vorgegeben werden. Durch die im JScience-Paket vorliegende Funktionalität zur Umrechnung von Einheiten ist es möglich, auch die der vorgegebenen Einheit äquivalente Einheiten zu berücksichtigen. Der erforderliche Operator ist in einem *Expression*-Objekt definiert, das bei jeder Erweiterung der Gleichung geklont wird und dessen Klon, gekapselt in einem *CSMathNode*-Objekt, in den Binärbaum eingefügt wird. Abbildung 60 zeigt den Aufbau von *CSPort*-Objekten.

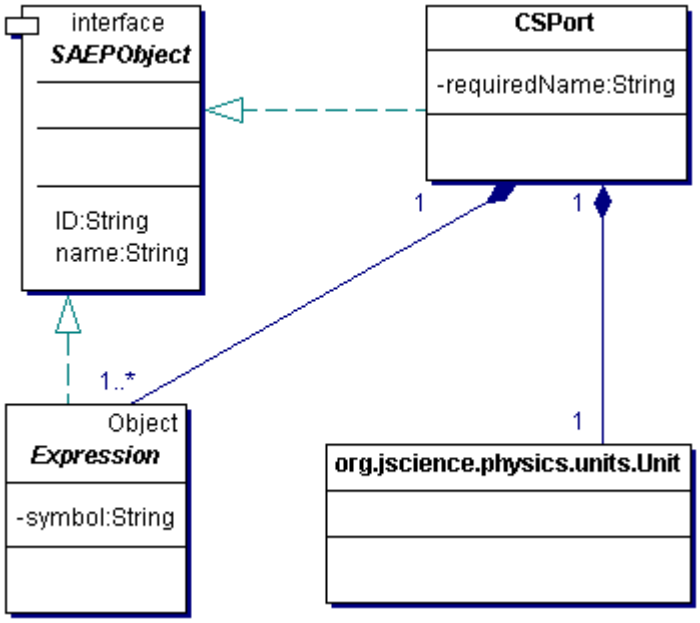


Abbildung 60: Die Klasse CSPort

Die Gleichung eines Kontaktsystems beschreibt mathematisch, wie die Eigenschaften der dem Wirkraum zugeordneten Produktstrukturkomponenten die durch das Kontaktsystem repräsentierte globale Eigenschaft beeinflussen. Das bedeutet, dass auf der „linken“ Seite einer Kontaktsystemgleichung die Eigenschaften der Produktstrukturkomponenten einfließen und dass auf der rechten Seite der Kontaktsystemgleichung die resultierende globale Eigenschaft abgelesen werden kann. Diese stellt die Ausgangsgröße eines Kontaktsystems dar. Diese globale Eigenschaft muss nun lediglich mit den Sensibilitäten der Produktstrukturkomponenten durch jeweils eine weitere Gleichung verknüpft werden. Das Ergebnis der Initialisierung ist daher ein ausgeprägtes Gleichungssystem, das mittels des in SAEP verwendeten Constraintsolvers ausgewertet werden kann (siehe auch 4.5.1).

4.5.2.2 Auswertung von Kontaktsystemen

Kontaktsysteme können auf verschiedene Arten ausgewertet werden, je nach Anwendungsfall:

- Darstellung der Abhängigkeiten als solche: Für die Ermittlung von Abhängigkeiten genügt es zunächst zu wissen, welche Merkmale mit welchen anderen Merkmalen zusammenhängen. Falls ein Kontaktsystem mit Hilfe eines Objekt der Klasse *SAEPContactSystemImpl* beschrieben ist, können die Abhängigkeiten direkt aus den *BinaryRelation*-Objekten abgeleitet werden, bzw. sie werden durch eben diese repräsentiert. Falls sie durch eine andere Klasse, die ein externes System kapselt (eine Implementierung des Interface *ContactSystem*), beschrieben werden, muss die entsprechende Methode des Kontaktsystems die Abhängigkeiten wiedergeben. Diese Abhängigkeiten werden als Objekte von Klassen repräsentiert, die das Interface *BinaryRelation* implementieren, können also auch mathematische Abhängigkeiten darstellen (*Expression*, *BinaryOperator* etc.). Falls das externe System auf numerischen Näherungsverfahren beruht und keine exakten Lösungen liefert, werden sie durch Objekte der Klasse *AbstractRelation* beschrieben und geben daher keine mathematisch auswertbaren Abhängigkeiten wieder.
- Bei Änderungen an Merkmalen werden die Auswirkungen auf andere Merkmale über die Auswertung der Gleichungssysteme berechnet, bzw. über die verwendete Implementierung des *ContactSystem*-Interfaces.

4.5.2.3 Verwaltung von Kontaktsystemen

Kontaktsysteme sind Wirkräumen zugeordnet und werden zusammen mit diesen in der Persistenzschicht abgelegt und modifiziert. Zusätzlich sind Kontaktsysteme in einer Kontaktsystembibliothek (*ContactSystemLibrary*) zusammengefasst, aus der sie vom Benutzer ausgewählt werden können. Kontaktsysteme können nach den Einheiten ihrer Ausgangsgrößen oder ihrer Bezeichnung ausgewählt werden.

Für die Neuerstellung eines Kontaktsystems muss zunächst der Binärbaum für die Rumpfgleichung des Kontaktsystems erstellt werden. Wie in Abschnitt 4.5.2. auf Seite 81 erläutert, bestehen die Binärbäume aus Objekten der Klasse *CSMathNode*, die über *Expresison*-Objekte miteinander verknüpft sind. Nach Erstellung der Rumpfgleichung müssen die Knoten angegeben werden, an denen die Gleichung des Kontaktsystems erweitert wird. Dazu müssen ihre Attribute *isDockingNode* auf den Wert *TRUE* gesetzt, sowie gegebenenfalls die erforderlichen Einheiten und Bezeichnungen definiert werden. Um den Vorgang für die Erstellung von Kontaktsystemen zu unterstützen, können Merkmale aus einer Merkmalsbibliothek ausgewählt werden. Diese werden nicht direkt referenziert, da in der Auswertungslogik nur an Produktkomponenten gekoppelte Merkmale ausgewertet werden, sondern ihre *AttributeName*-Objekte als Vorlage für die Bezeichnungsbedingungen benutzt, sowie ihre *Unit*-Objekte für die Definition der erforderlichen Einheit herangezogen.

Aus diesen Knoten werden die entsprechenden *CSPort*-Objekte erzeugt, die für die Anbindung der Produktmerkmale an die entsprechenden Stellen im Binärbaum sorgen.

4.5.3 Beziehungsnetzwerkmanager (*Relation Network Manager*)

Der *Relation Network Manager* ist zuständig für die Navigation in Beziehungsnetzwerken, die zwischen den Merkmalen der Produktstruktur bestehen, sowie für den Unterhalt und die Pflege der Netzwerke. Er basiert auf den Basisklassen zur Darstellung nicht-streng hierarchischer Strukturen. Zentrale Klasse für die Repräsentation von Netzwerken ist *SimpleRelationModel*, die Funktionalität für die Darstellung von hierarchischen Produktstrukturen bereitstellt und zusätzliche Methoden für die Verwaltung von und die Navigation über *BinaryRelation*-Beziehungsobjekten, die einzelne Merkmale miteinander verbinden. Da die Beziehungsobjekte die Merkmale nicht direkt verbinden, sondern indirekt über die Verknüpfung von *DesignElementAttributeRelation*-Objekten (siehe Abschnitt 4.1.1, Seite 86), können Beziehungen über verschiedene Produktstrukturkomponenten hinweg verwaltet und ausgewertet werden. Die Klasse *RelationNetworkModel* erweitert *SimpleRelationModel* um Methoden zur Ermittlung und Pflege von komplexen Beziehungsnetzwerken, in denen Beziehungen zwischen Beziehungen bestehen (Superbeziehungen), wie dies beispielsweise bei Binärbäumen zur Abbildung von Gleichungen der Fall ist (siehe Abbildung 61). Das bedeutet, es erkennt auch Abhängigkeiten zwischen Merkmalen, die nicht direkt über ein Beziehungsobjekt miteinander verknüpft sind sondern über Objekte, die Superbeziehungen repräsentieren. Dabei kann die Schachtelung der Beziehungen beliebig tief sein.

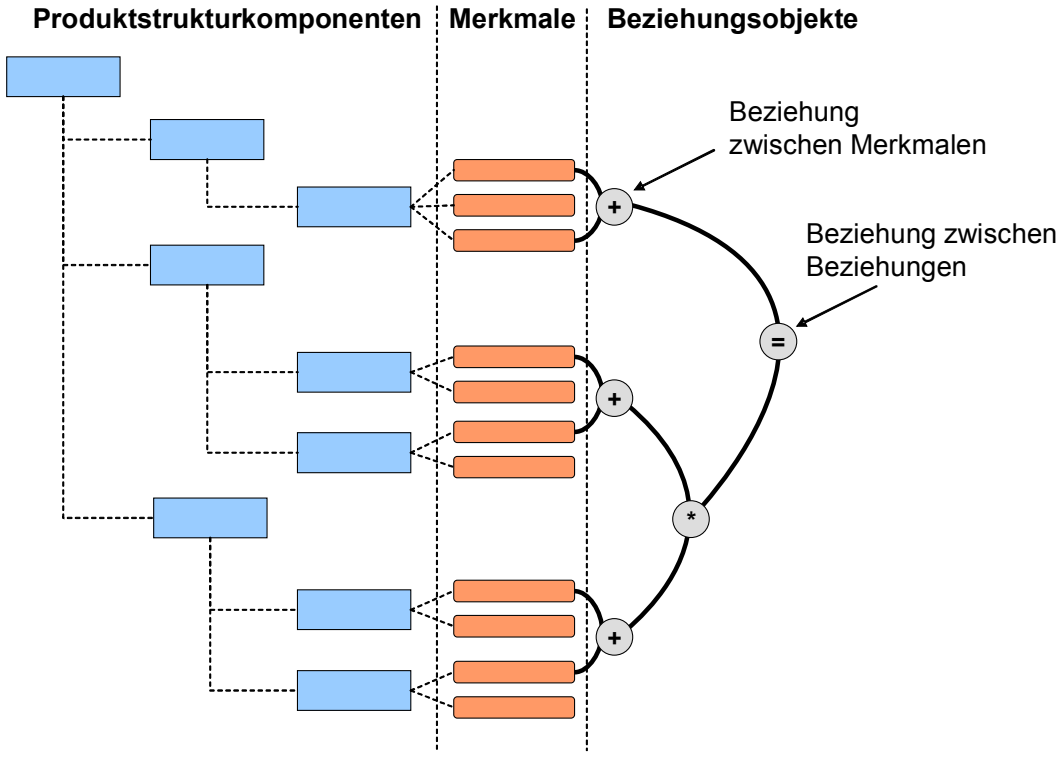


Abbildung 61: Superbeziehungen und Beziehungen zwischen Merkmalen

SimpleRelationModel (und damit auch *RelationNetworkModel*) sind von der Klasse *ViewTreeModel* abgeleitet. Diese Klasse ist wiederum direkt von der Klasse *AbstractRelationModel* abgeleitet und implementiert die dort abstrakt deklarierte Methode *setChildren(SAEPNode parenNode)* (siehe auch Abschnitt 4.2). Dazu wertet sie die *ParentChildRelation*-Objekte aus einem bestimmten *View*-Objekt aus und generiert daraus eine entsprechende, hierarchische Ordnung von *SAEPNode*-Objekten, die die in den *ParentChildRelation*-Objekten verknüpften Objekte (in diesem Falle Objekte der Klasse *DesignElement*) kapseln.

Die Sichten auf ein Produkt sind in einem Objekt der Klasse *ViewRepository* hinterlegt, das seinerseits einem Produkt (*Product*) zugeordnet ist. Objekte der Klasse *Product* besitzen zusätzlich ein Attribute der Klasse *RelationContext*, in dem sämtliche in einem Produkt bekannten Beziehungen zwischen Produktmerkmalen abgelegt sind. *Product* stellt damit das Wurzelelement in der Aggregationsstruktur dar, d.h., von einem Objekt der Klasse *Product* sind sämtliche Objekte des Aggregationsbaums erreichbar.

Abbildung 62 zeigt die Klassenstruktur des *RelationNetworkManager*.

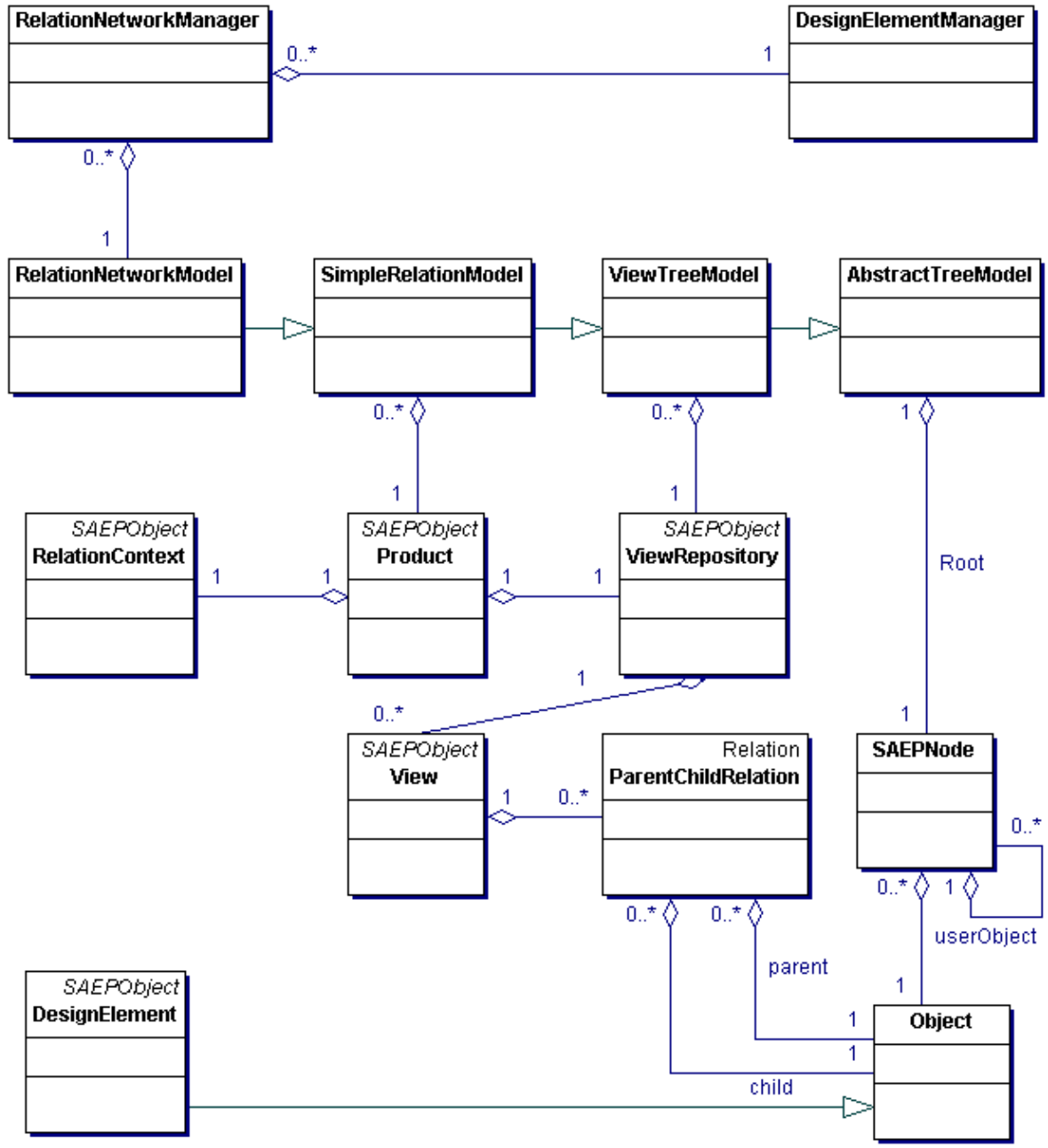


Abbildung 62: Die Klassenstruktur des Relation Network Managers

Der Beziehungsnetzwerkmanager verwaltet in seinem *RelationNetworkModel* die komplette Struktur eines Produkts. Er hat den Überblick über sämtliche Produktstrukturkomponenten, verwaltet die Sichten auf die Produktstruktur und hat Zugriff auf alle Produktmerkmale sowie die Beziehungen zwischen ihnen. Dazu greift er auf die Integrationsschicht zu, die einen transparenten Zugriff auf alle relevanten Objekte der angebundenen Autorensysteme, bzw. die SAEP-eigene Persistenzschicht bereitstellt.

Der *Relation Network Navigator* ist außerdem zuständig für die Evaluierung der Produkteigenschaften anhand der Produktanforderungen. Für die Auswertung werden lediglich quantitative Anforderungen herangezogen. Qualitative Anforderungen können in der vorliegenden

Implementierung nicht für eine rechnerunterstützte Evaluierung von Produkteigenschaften herangezogen werden.

Für die rechnerunterstützte Evaluierung von Produkteigenschaften werden komponentenweise die *PropertyRequirementRelation*-Objekte herangezogen und sukzessive die Anforderungs-Eigenschaftspaare abgearbeitet. Zunächst wird der Wert der Produkteigenschaft ermittelt. Ist es eine einfache Eigenschaft (direkt spezifiziert) so kann der Wert direkt übernommen werden. Handelt es sich um eine aggregierte Eigenschaft, muss zunächst ihr Wert mit Hilfe des *ConstraintSolvers* ermittelt werden. Für die Überprüfung wird der resultierende Wert als Ausprägung übernommen. Danach wird überprüft, ob die Einheiten der Paarung kompatibel sind. Ist dies der Fall, werden sie gegebenenfalls auf die gleiche Größenordnung umgerechnet (normiert). Ist eine Anforderung mit einem Einzelwert ausgeprägt, wird danach einfach überprüft, ob die Werte der Anforderung und der Eigenschaft in der betrachteten Paarung übereinstimmen. Ist die Anforderung durch einen Wertebereich definiert, wird überprüft, ob der Wert der Produkteigenschaft innerhalb der durch die Anforderung vorgegebenen Intervallgrenzen liegt. Ergebnis der Einzelüberprüfung ist ein Boolscher Wert, der TRUE ist, wenn die Gleichung, bzw. die Ungleichungen erfüllt sind, bzw. FALSE, wenn sie nicht erfüllt sind. Als Ergebnis der Gesamtanalyse werden zwei Hash-Tabellen erzeugt. In der einen werden die *PropertyRequirementRelation*-Objekte als Schlüsselattribut mit dem jeweiligen Ergebnis der Einzelüberprüfung als Wertattribut abgelegt. In der anderen Hashtabelle werden die Ergebnisse der Einzelüberprüfung als Schlüsselattribut mit dem *PropertyRequirementRelation*-Objekt als Wertattribut eingetragen. Auf diese Weise kann entweder nach erfüllten, bzw. nicht erfüllten Anforderungen (Schlüssel ist das Ergebnis der Auswertung) oder das Ergebnis für eine bestimmte Paarung gesucht werden (Schlüssel ist die Paarung).

Der *Relation Network Navigator* kann außerdem überprüfen, ob den Produktstrukturkomponenten Anforderungen ohne verknüpfte Produkteigenschaften, bzw. umgekehrt, ob es Eigenschaften ohne zugeordnete Anforderungen gibt. Diese unvollständigen Paarungen werden bei der Auswertung nach Anforderungskonformität der Produkteigenschaften ignoriert.

4.5.4 Der *Attribute Library Manager*

Der *Attribute Library Manager* stellt die Funktionalität für die Verwaltung von Produktmerkmalsbibliotheken, sowie für die (manuelle) Zuordnung von Produktmerkmalen zu Elementen der Produktstruktur bereit. Dazu gehört auch die Verwaltung von Sichten auf die hierarchische Struktur von Sammlungen von Produktmerkmalen, die mit Objekten der Klasse *View* dargestellt werden. Wie in Abbildung 63 erkennbar ist, ist die Klasse *AttributeTreeModel* von der Klasse *ViewTreeModel* abgeleitet. Damit ist die Funktionsweise zur Verarbeitung von *ParentChildRelation*-Objekten in Objekte der Klasse *SAEPNode* analog zu der Funktionsweise in der Klasse *RelationNetworkModel*. Gleiches gilt für die Verwaltung von Sichten in *ViewRepository*-Objekten. *AttributeTreeModel* stellt zusätzlich Funktionalität für die Ableitung von Produktmerkmalsbibliotheken bereit. Dabei werden alle Objekte der Ur-

sprungsbibliothek geklont, also genaue Kopien der Ursprungsobjekte erstellt. Nach der Ableitung einer Bibliothek kann die neue (kopierte) Bibliothek, nach den Bedürfnissen des Benutzers angepasst werden. Das heißt, bestehende Sichten können verändert oder entfernt und neue hinzugefügt werden oder Produktmerkmale aus der Sammlung entfernt oder neue Produktmerkmale eingefügt werden.

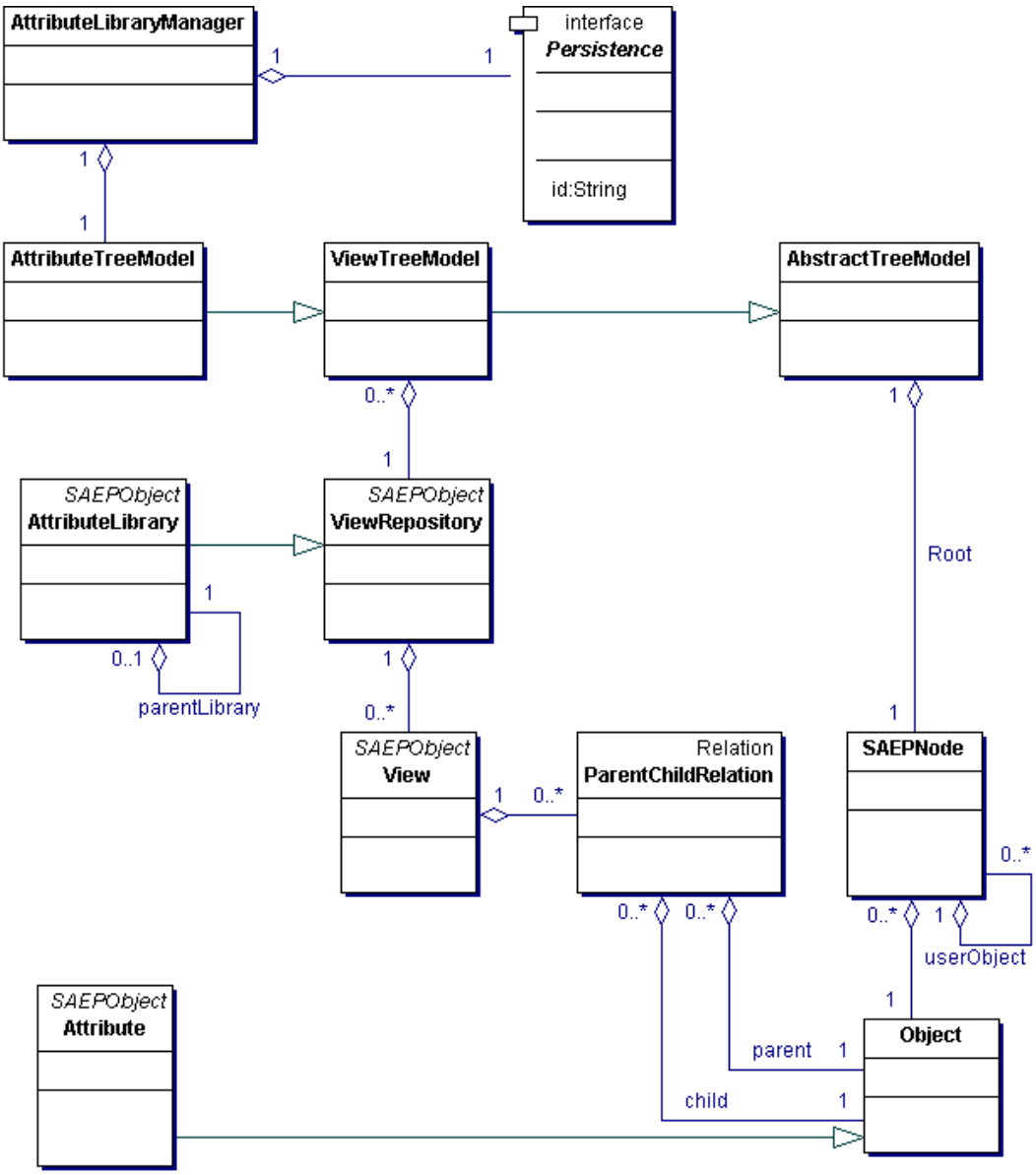


Abbildung 63: Klassenstruktur des Attribute Library Managers

4.6 Fazit

Die in diesem Abschnitt vorgestellten Datenmodelle und Vorgehensweisen bilden die Grundlage für die Implementierung der eigentlichen Programmlogik sowie der notwendigen Rahmenfunktionalität z.B. für die dauerhafte Speicherung von Arbeitsergebnissen. Die Benutzungsschnittstelle, die die Interaktionen des Benutzers mit dem Programm steuert und

auswertet, wird im folgenden Abschnitt, begeleitend zur Verifikation des Konzepts und der Implementierung vorgestellt.

5 Verifikation

Für die Verifikation wurde als Beispiel eine PKW-Bremsanlage[54] ausgewählt, die in einer Weiterentwicklung mit einem Antiblockiersystem (ABS) ausgestattet werden soll. Abbildung 64 zeigt den schematischen Aufbau der hydraulischen Anlage (ohne elektronische Komponenten).

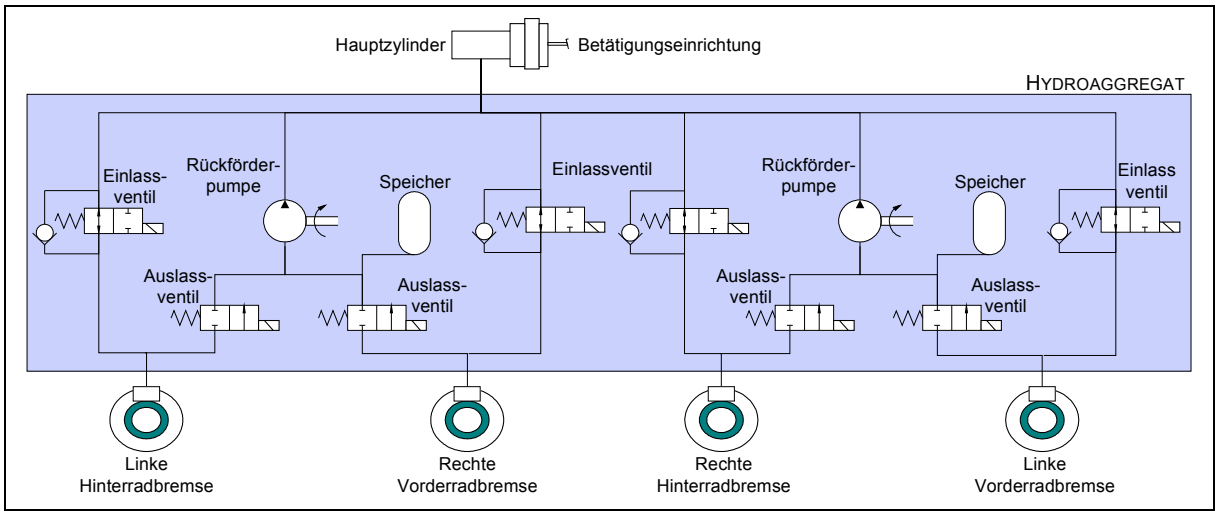


Abbildung 64: Hydrauliksystem einer ABS-Anlage [Bosc-2003]

In dem Szenario wird die hydraulische Bremsanlage um elektrische und elektronische Komponenten ergänzt. Dies sind im wesentlichen Raddrehzahlsensoren, mir deren Hilfe ein drohendes Blockieren des Rades anhand einer starken Radverzögerung erkannt werden kann, elektrisch ansteuerbare Ventile (je gebremstes Rad mindestens ein Einlass- und ein Auslassventil), die die Bremskraft steuern, eine Rückförderpumpe, die die durch die Auslassventile freigesetzte Bremsflüssigkeit zurück in das Hauptreservoir pumpt, sowie ein Controller, der die Informationen der Sensoren erfasst, verarbeitet und über einen Leistungsverstärker die Ventile ansteuert, um so die Bremskraft zu regeln [BrBi-2003].

In dem Verifikationsszenario werden die einzelnen Tätigkeiten, die von SAEP unterstützt werden, anhand der Neu- bzw. Weiterentwicklung der Bremsanlage erläutert[55].

---

54 Die Bremsanlage ist kein konkreter, industrieller Anwendungsfall, sondern wird im Rahmen der Dissertation plausibel der Wirklichkeit nachempfunden.

55 Der Begriff „Neukonstruktion“ bedeutet in diesem Zusammenhang, dass ein neues Produkt entsteht, das aber an Vorgängermodelle angelehnt ist. Insofern ist es im strengen Sinne eine Weiterentwicklung eines bestehenden Produkts bzw. eine Änderungskonstruktion.

5.1 Definition von Produktmerkmalen

Die Definition von Produktanforderungen steht am Anfang der Produktentwicklung[56] und zieht sich über alle Produktentwicklungsphasen hinweg (siehe auch Abschnitt 2.1.2). Das bedeutet, dass es eine initiale Menge von Anforderungen zu Beginn der Entwicklungstätigkeiten gibt, die im Laufe des Prozesses erweitert und konkretisiert wird.

5.1.1 Verwaltung von Produktanforderungen in Merkmalsbibliotheken

Für die Verwaltung von Produktmerkmalen in Merkmalsbibliotheken ist der *Attribute Library Manager* zuständig. Produktmerkmale werden von ihm in Merkmalsbibliotheken verwaltet und können aus diesen ausgewählt und den Elementen der Produktstruktur zugeordnet werden. Merkmale können entweder Anforderungen oder Eigenschaften sein. Der Benutzer entscheidet je nach Tätigkeit, welche Art von Merkmalen er auswählen möchte. Da sich der Benutzer im vorliegenden Szenario in der Phase der Anforderungsdefinition befindet, werden im folgenden lediglich Anforderungen betrachtet. Die Auswahl von Produkteigenschaften kann aber auf die gleiche Weise erfolgen.

Abbildung 65 zeigt eine Bibliothek (*Library*) für die Produktgruppe „Scheibenbremsen“. Die hierarchische Struktur wird durch Konkretisierungsbeziehungen zwischen den Merkmalen hergestellt. Das bedeutet, dass sich Merkmale auf einer höheren Ebene in Merkmale auf tiefer gelegenen Ebenen konkretisieren. Zum Beispiel wird das Merkmal „Ergonomie“ in das Merkmal „Bedienungskomfort“ und dieses weiter in „FeinfuehligeBremsbetaetigung_mit_BKV“ konkretisiert (BKV = „Bremskraftverstärker“). Jedes Merkmal kann dabei mehrere abstrakte Merkmale konkretisieren (mehrere „Elternknoten“ besitzen), wie z.B. das Merkmal „MaximaleBremspedalkraft“, wie in Abbildung 65 dargestellt (Anmerkung: „MaximaleBremspedalkraft“ wird intern als gerichtete Bereichsanforderung im System abgebildet, die Bezeichnung dient nur zur leichteren Navigierbarkeit durch den Benutzer). Dadurch werden vordefinierte Merkmalsmengen wesentlich übersichtlicher, und der Benutzer kann gezielt zu den relevanten Merkmalen hingeführt werden.

Anforderungen werden in SAEP unterschieden in ausgeprägte Anforderungen und in nichtausgeprägte Anforderungen. Anforderungen können schon mit Werten belegt sein, bevor ein konkretes Produktmodell existiert. Bei Bremsanlagen für PKW schreibt der Gesetzgeber beispielsweise einen maximalen Bremspedalweg von 120 mm vor. Diese Anforderung muss also von allen PKW-Bremsen erfüllt werden und kann demnach in der entsprechenden Bibliothek schon ausgeprägt sein. Anforderungen, wie z.B. das Gewicht des Pedals, können individuell abhängig von der jeweiligen Konstruktionsaufgabe sein und sind demnach nicht in Bibliotheken ausgeprägt. Eigenschaften können dagegen prinzipiell nicht in Bibliotheken ausgeprägt werden, da diese von der jeweiligen Realisierung durch ein konkretes Bauteil abhängig sind.

56 Die Frage, wie relevante Anforderungen ermittelt werden können, ist nicht im Blickfeld dieser Arbeit, sondern wie gefundene Anforderungen verarbeitet und mit den entwickelten Produkteigenschaften verknüpft werden können.

Anforderungen werden je nach ihrem Charakter unterschiedlich dargestellt. Ausgeprägte, quantitative Anforderungen werden als grüne Symbole (Icons) mit der Beschriftung „123“ dargestellt, während ausgeprägte, qualitative Anforderungen mittels blauer Symbole mit der Beschriftung „abc“ symbolisiert werden. Nicht ausgeprägte Merkmale sind rot und mit einem Fragezeichen beschriftet dargestellt.

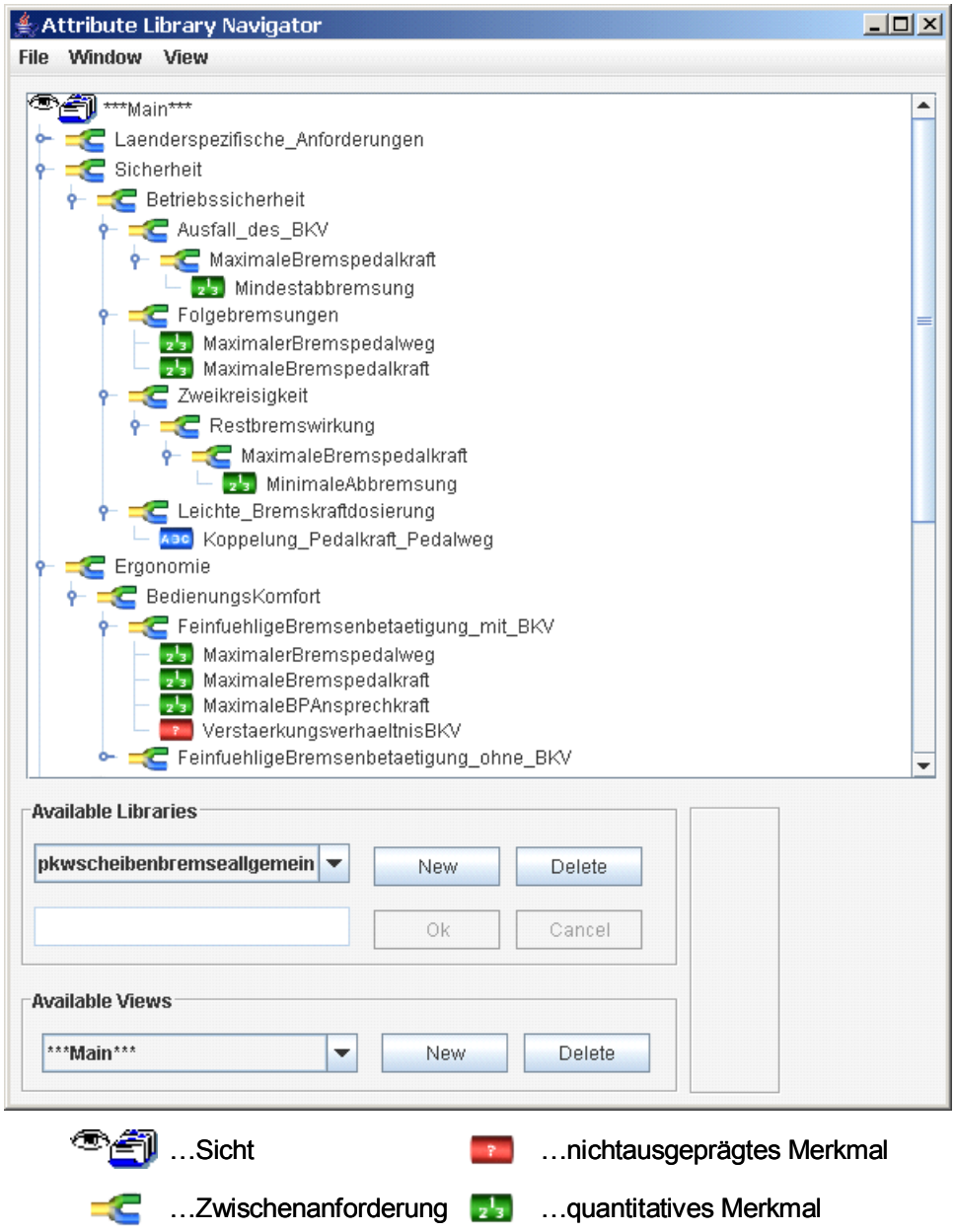


Abbildung 65: Programmoberfläche zur Verwaltung von Merkmalsbibliotheken (Ausschnitt)

Anforderungsbibliotheken befinden sich ihrerseits auf verschiedenen Konkretisierungsebenen. Das bedeutet, dass Anforderungen, die allgemein für den Bereich Maschinenbau relevant sind, in einer Bibliothek zusammengefasst werden. Aus dieser Bibliothek können dann wiederum weitere Bibliotheken, etwa für die Bereiche Automobil-, Flugzeug- oder Schiffbau abgeleitet werden, indem die Anforderungsstruktur kopiert und unter neuem Namen angelegt

wird. Danach werden irrelevante Anforderungen gelöscht und für den jeweiligen Bereich relevante Anforderungen hinzugefügt. Diese Bibliotheken können beliebig weiter konkretisiert werden in z.B. unternehmensspezifische bis hin zu produktgruppenspezifischen Anforderungsbibliotheken. Diese dienen als Basis für die Definition der Anforderungen an ein zu entwickelndes Produkt. Es können Anforderungen aus beliebig vielen Bibliotheken ausgewählt und zugeordnet werden, beispielsweise aus solchen, die speziell den Bereich „Umweltschutz“ abdecken.

Da in dem Szenario eine herkömmliche hydraulische Bremsanlage durch ein ABS ergänzt werden soll, das unter anderem elektrische und elektronische Komponenten (elektrisch ansteuerbare Hydraulikventile, Controller für die Regelung der Ventile) umfasst, treten neue Anforderungen für den Entwurf der Bremsanlage hinzu. Diese betreffen beispielsweise die Sicherheit bei Versagen der elektrischen/elektronischen Komponenten und die Versorgung mit elektrischer Energie. Ein weiterer wichtiger Punkt ist die Störanfälligkeit durch elektromagnetische Einwirkungen anderer Fahrzeugkomponenten sowie die elektromagnetische Störwirkung auf andere Komponenten. Die letzten beiden Punkte werden unter dem Begriff Elektromagnetische Verträglichkeit (EMV) zusammengefasst[57] [Bosc-2003]. Störeinwirkungen aus der Zündanlage können die Funktionsfähigkeit des ABS beeinträchtigen, da die Geräte meist im Motorraum in der Nähe der Bremsenergieversorgung untergebracht sind. Aber auch andere elektronische Komponenten, die in der Nähe verbaut sind, sowie deren Versorgungsleitungen für elektrische Energie sind potentielle Störquellen. Hier bewirken sowohl die Störeinkopplungen durch Galvanische Einkopplung (gleiche Impedanz der Komponenten) als auch Störeinkopplungen durch elektromagnetische Wechselfelder eine Beeinflussung der Steuerungselektronik [Rode-2000]. Abbildung 66 zeigt eine Merkmalsbibliothek für den Bereich Elektronik.

57 Nach VDE 0870 ist die Elektromagnetische Verträglichkeit (EMV) definiert als „Fähigkeit einer elektrischen Einrichtung, in ihrer elektromagnetischen Umgebung zufriedenstellend zu funktionieren, ohne diese Umgebung, zu der auch andere Einrichtungen gehören, unzulässig zu beeinflussen.“ [Meus-1999].

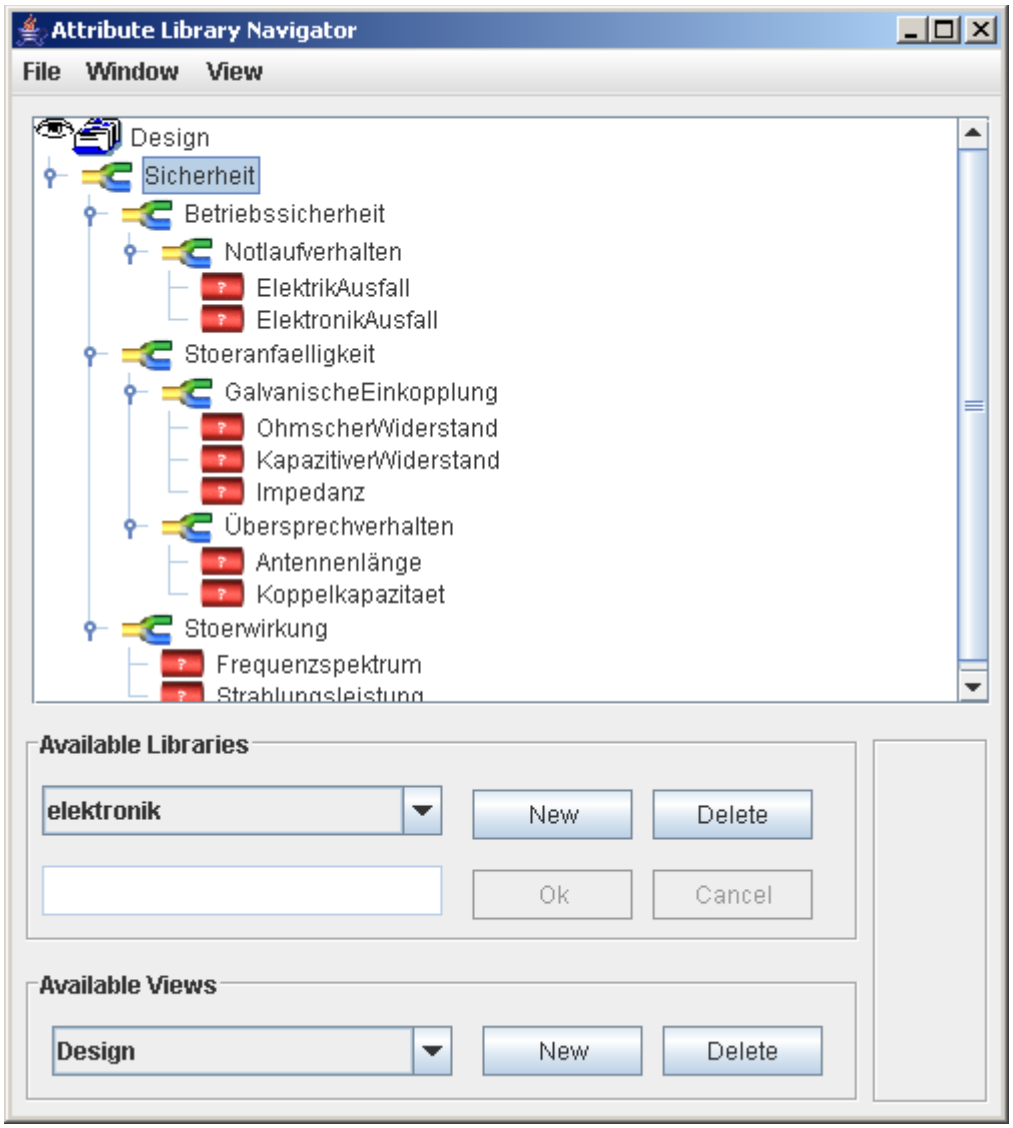


Abbildung 66: Merkmalsbibliothek für elektronische Komponenten (Ausschnitt)

5.1.2 Ausprägung von Merkmalen

Merkmale können durch verschiedene Tätigkeiten ausgeprägt werden. Produkteigenschaften werden vorzugsweise in den dedizierten Produktmodellierungswerkzeugen definiert und, sofern eine Schnittstelle zu SAEP besteht, in das SAEP-eigene Modell übernommen. Falls dies nicht möglich sein sollte, können sie auch manuell durch den Benutzer mit Hilfe der SAEP-Benutzungsoberfläche definiert werden. Dies geschieht ausschließlich im Zusammenhang mit einem bestimmten Produkt und nicht im Kontext der Anforderungsbibliotheken. Anforderungen können dagegen sowohl in Merkmalsbibliotheken ausgeprägt werden als auch im Zusammenhang mit einem konkreten Produkt, Bauteil oder einer Baugruppe.

Abbildung 67 zeigt die Programmoberfläche zur Beschreibung von Produktmerkmalen. Der gerade am System angemeldete Benutzer wird automatisch als Verantwortlicher eingetragen, wenn er ein neues Produktmerkmal anlegt. Zudem wird das aktuelle Datum als Erstellungsdatum eingetragen. Der Benutzer muss definieren, welcher Art das neue Merkmal ist, also ob es

sich um eine Anforderung (voreingestellte Auswahl) oder um eine Eigenschaft handelt[58]. Des Weiteren muss der Benutzer angeben, ob es sich um ein definierendes oder ein ausschließendes Merkmal handelt. Als weitere Möglichkeit kann der Benutzer die Wichtigkeit der Anforderung (bzw. die Gewichtung) mit einem stilisierten Schieberegler (Slider) eintragen. Auf der rechten Seite der Oberfläche wird das Merkmal definiert, also seine qualitative bzw. quantitative Ausprägung. In dem Beispiel ist eine Bereichsanforderung zu sehen, deren beide Intervallgrenzen gültige Werte sind, und keiner von beiden Grenzwerten ist Optimierungsrichtung.

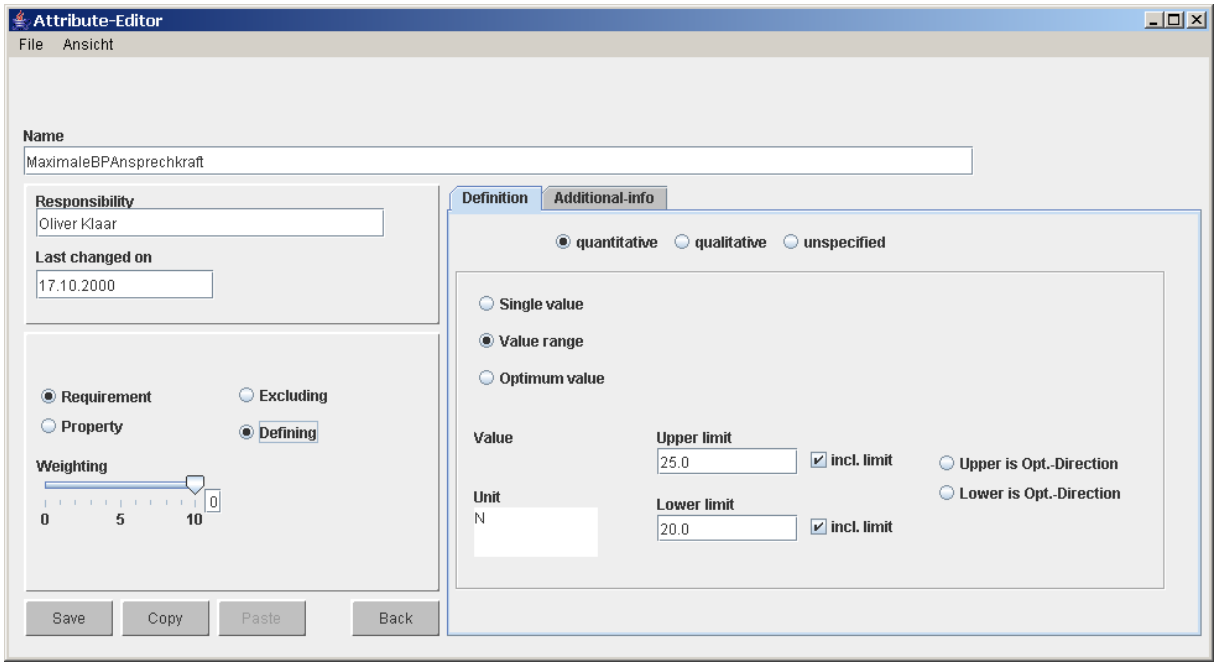


Abbildung 67: Programmoberfläche zur Definition von Produktmerkmalen

5.1.3 Modellierung von Produktstrukturen

Bei der Modellierung von Anforderungen steht das zu entwickelnde (Teil-)Produkt bzw. System im Mittelpunkt. Ihm und seinen Unterkomponenten werden Anforderungen bei der Anforderungsmodellierung zugeordnet. Dabei ist zum einen zu berücksichtigen, dass die (hierarchische) Produktstruktur im allgemeinen erst während des Produktentwicklungsprozesses festgelegt wird und zum anderen, dass im Laufe des Entwicklungsprozesses verschiedene Sichten auf die Produktstruktur relevant werden. Wird ein Produkt entworfen, können mit dem *Design Element Manager* von SAEP hierarchische Produktstrukturen aufgebaut werden. Diese Produktstrukturen können in verschiedenen Sichten angelegt werden. Das bedeutet, dass man zunächst eine Entwurfssicht erstellt, die z.B. die Produktfunktionen widerspiegelt.

---

58 Der Typ „Anforderung“ ist Standardeinstellung, da die Definition von Produkteigenschaften vorwiegend in externen Werkzeugen vorgenommen wird.

Zu einem späteren Zeitpunkt werden konkretere Produktstrukturen aus dieser erzeugt, wie z.B. Montagesichten, die dieselben Produktkomponenten und Unterbaugruppen beinhalten, jedoch anders gegliedert sind. Dies wird ermöglicht durch die Implementierung von nicht-streng hierarchischen Produktstrukturen. Das heißt, eine Komponente kann mehrere Unterkomponenten haben und zusätzlich auch mehrere übergeordnete Komponenten, deren Bestandteil sie selbst ist. Abbildung 68 zeigt den *Design Element Manager*, der eine (noch unvollständige) Sicht „system“ auf die Produktstruktur einer herkömmlichen Bremsanlage zeigt.

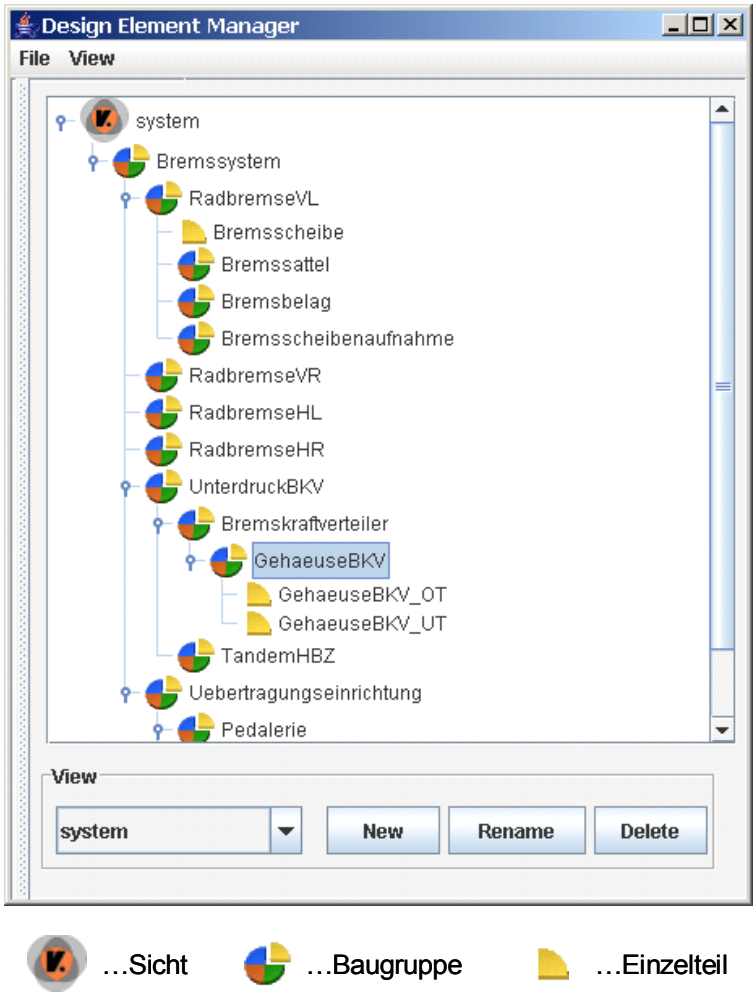


Abbildung 68: Programmoberfläche zur Verwaltung von Produktstrukturen

Um Anforderungen an Produktkomponenten in SAEP zuzuordnen, navigiert der Benutzer mit der Maus durch die für die Aufgabe relevanten Anforderungsbibliotheken, wählt die entsprechenden Anforderungen aus, kopiert sie und fügt sie beim entsprechenden Element der Produktstruktur ein (siehe Abbildung 69). Falls Anforderungen in der Bibliothek noch nicht ausgeprägt sind, kann er diese aus dem *Design Element Manager* mit Hilfe der in Abbildung 67 abgebildeten Komponente zur Definition von Produktmerkmalen ausprägen.

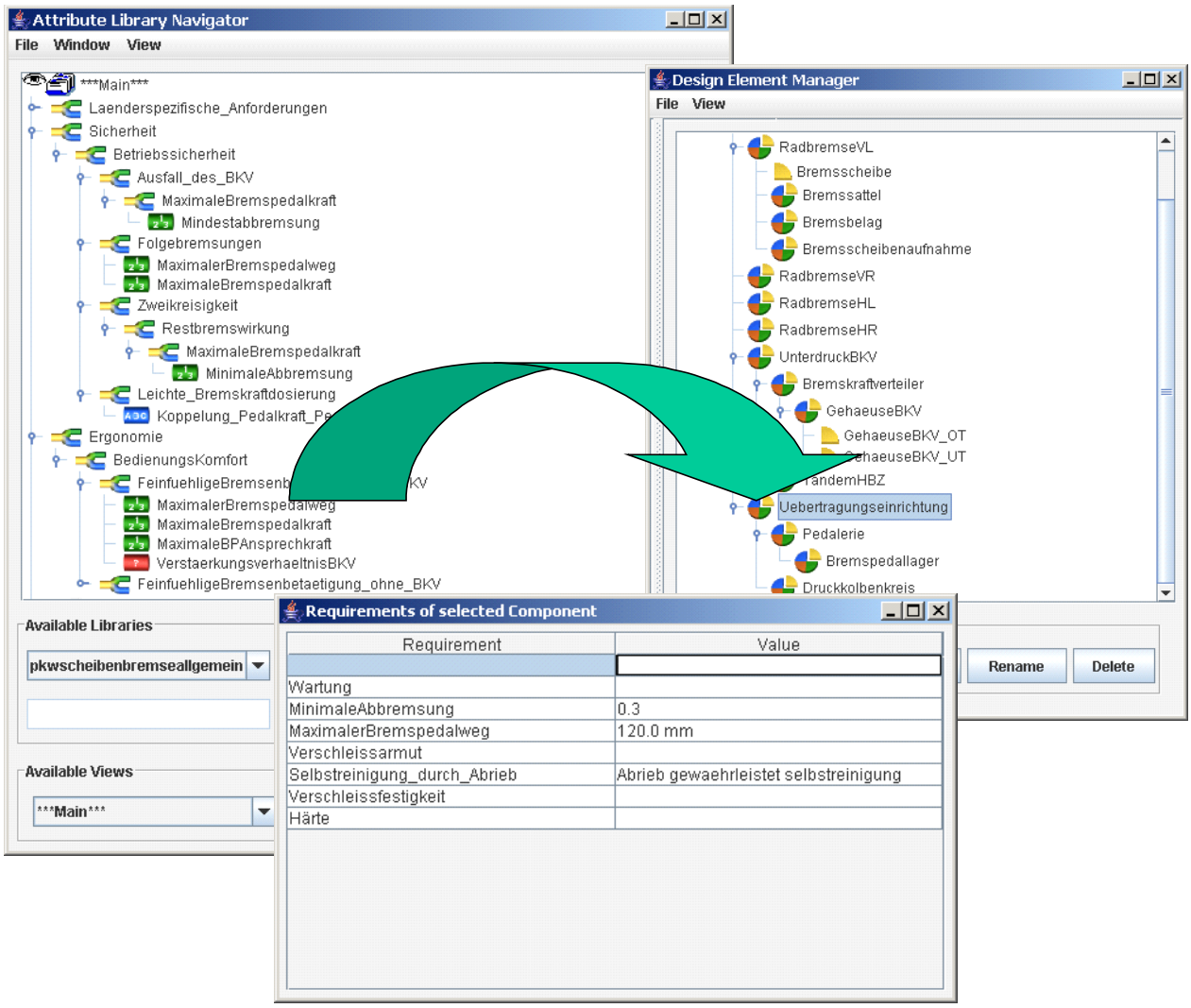


Abbildung 69: Zuordnung von Produktmerkmalen zu Elementen der Produktstruktur

Eine besondere Art von Produktstrukturen sind Schablonen (*Templates*), in denen Produktkomponenten mit ihren Merkmalen sowie den Aggregationsbeziehungen zwischen den Strukturelementen auf einer abstrakten Ebene vordefiniert sind (siehe Abschnitt 3.2.5, Seite 55 und Abschnitt 3.3.4). Die Modellierung von Schablonen unterscheidet sich prinzipiell nicht von der Modellierung von konkreten Produktstrukturen. SAEP ermöglicht es aber, Schablonen zu konkreten Produkten zu instanziieren, indem die Schablonenobjekte im Strukturbaum zu Objekten der Klasse *Assembly* (Baugruppe), oder *Part* (Einzelteil) konkretisiert werden. Das bedeutet, dass Kopien der Schablonenobjekte angefertigt werden mit allen Attributen der Schablonenobjekte sowie deren zugeordneten Produktmerkmalen. Diese werden an Stelle der Schablonenobjekte in den Strukturbaum eingefügt. Schablonen sind vor allem dann nützlich, wenn sie zur Instanziierung von Produktkomponenten mit stabilen Strukturen benutzt werden, also beispielsweise bei Standardteilen. Insofern eignen sie sich vor allem für Variantenkonstruktionen, in denen lediglich einzelne Parameter verändert werden.

Abbildung 70 zeigt eine Schablone Bremssystem und die zugeordneten Schablonenobjekte in einem Produktstrukturbaum.

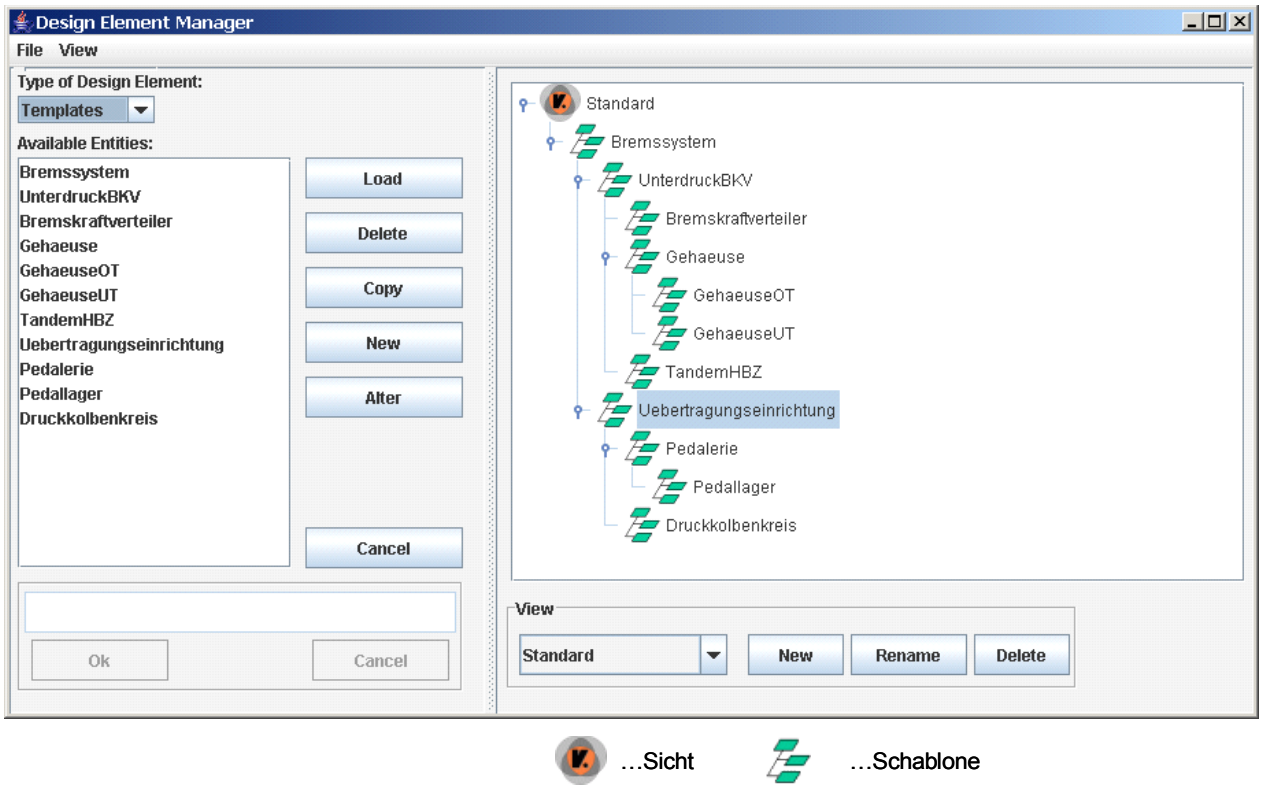


Abbildung 70: Modellierung von Schablonen mit dem Design Element Manager

Veranlasst der Benutzer eine Instanziierung der Schablone zu einer konkreten Baugruppe oder einem Einzelteil, können im nachfolgenden Schritt die neu erzeugten Elemente mit Elementen aus externen Werkzeugen verknüpft werden.

5.2 Verknüpfung von Produktstrukturelementen und deren Merkmalen

Bei der Kopplung von SAEP mit externen Werkzeugen zur Produktmodellierung sind, wie in Abschnitt 3.2.4 erläutert, drei Szenarien möglich:

1. Die Modellentitäten des externen Systems sind Vorlage für die Erzeugung von Modellentitäten in SAEP.

2. die Modellentitäten in SAEP sind Vorlage für die Erzeugung von Modellentitäten im externen System.

3. Externes System und SAEP haben jeweils eigene Modelle, die miteinander verknüpft werden sollen.

Die ersten beiden Varianten sind im Bezug auf die Verknüpfung der Modellentitäten relativ einfach zu realisieren. „Relativ“ bedeutet in diesem Zusammenhang, dass die Hauptproblematik in der Realisierung der Schnittstellen zu den externen Systemen liegt. Für jedes im jewei-

ligen Originalsystem erzeugte Element wird ein entsprechendes Element im Sekundärsystem erzeugt und mit diesem verknüpft. Eine Voraussetzung dafür ist, dass in den Schnittstellen entsprechende Methoden gegeben sind. Falls eine Echtzeitkopplung erreicht werden soll, ist eine weitere Voraussetzung für den ersten Fall, dass das angebundene System SAEP über die Schnittstelle mitteilen kann, dass eine Modellentität erzeugt wurde und diese auch abgefragt werden kann.

Bei Variante drei, also der Verknüpfung von in SAEP bereits bestehenden Produktstrukturelementen mit Elementen aus externen Werkzeugen ist ein zweistufiges Verfahren zweckmäßig, wie in Abschnitt 3.2.4 erläutert. Zunächst müssen die korrespondierenden Komponenten der Produktmodelle aufeinander abgebildet werden, um anschließend deren Merkmale miteinander zu verknüpfen.

5.2.1 Verknüpfung von korrespondierenden Produktkomponenten

Für die Verknüpfung von korrespondierenden Produktkomponenten bietet SAEP grundsätzlich zwei Möglichkeiten an:

- Automatische Erkennung von korrespondierenden Produktstrukturelementen, ihren Eigenschaften und deren Verknüpfung mit entsprechenden SAEP-internen Entitäten nach Bestätigung durch den Benutzer
- Manuelle Verknüpfung von korrespondierenden Produktstrukturelementen und Eigenschaften

Die beiden Vorgehensweisen finden nicht isoliert voneinander statt sondern parallel. SAEP versucht zunächst, die im fremden System vorhandenen Produktstrukturelemente den im SAEP-eigenen Modell vorhandenen anhand ihrer Bezeichnung zuzuordnen. Gelingt dies nicht, muss der Benutzer eingreifen und die entsprechenden Komponenten manuell miteinander verknüpfen. Dabei greift SAEP auf alle Komponenten eines SAEP-intern abgespeicherten Produkts zu, ohne Berücksichtigung der sie enthaltenden Sichten. Aus den automatisch oder manuell identifizierten Komponenten wird eine neue Sicht auf die Produktstruktur erzeugt, deren Struktur sich aus den Aggregationsbeziehungen im Modell des externen Werkzeugs ergibt.

Für die Ankopplung von CAD-Systemen wurde der *BrepBrowser* implementiert, der über die *CADServices*-Schnittstelle (siehe Abschnitt 4.3.2, Seite 98) auf CAD-Systeme zugreift. In dem Szenario wird das Geometriemodell der Baugruppe „Bremse“ in das SAEP-Modell überführt. Abbildung 71 zeigt die Benutzungsoberfläche des BrepBrowsers mit einem Ausschnitt der Baugruppe. Die Benutzungsoberfläche des *BrepBrowser* ist dreigeteilt. Auf der linken Seite ist die hierarchische, topologische Produktstruktur im BRep-Modell von der Baugruppe bis hinunter zum Eckpunkt dargestellt. In der Mitte wird die dreidimensionale Geometrie in tesselierter Darstellung angezeigt. Auf der rechten Seite sind die Eigenschaften des in der BRep-Hierarchie ausgewählten Elements eines CAD-Modells in tabellarischer Form aufgelistet.

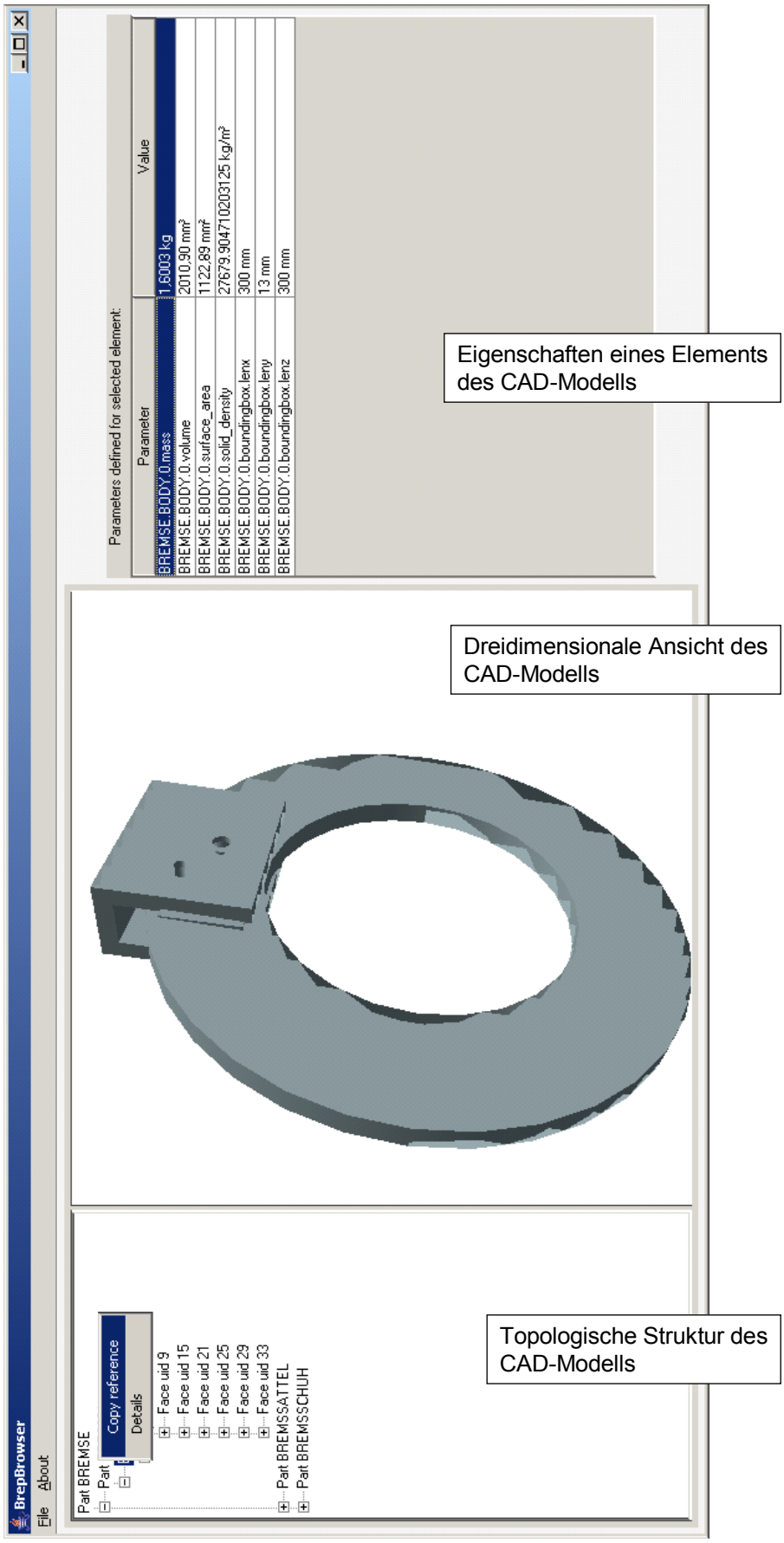


Abbildung 71: Ausschnitt aus der Benutzungsoberfläche des BrepBrowser mit der Darstellung einer Baugruppe und der Modellhierarchie

Nach dem Anstoßen des Analysevorgangs werden die identifizierten Elemente des SAEP-internen Produktmodells im *Product Manager* angezeigt. Auf Knopfdruck („Copy“) können diese übernommen und im *Product Structure Navigator* angezeigt werden. Abbildung 72 zeigt den *Product Manager* mit den identifizierten Elementen und Abbildung 73 den *Product Structure Navigator* mit den identifizierten Elementen, eingeordnet in eine Sicht, die die Aggregationsbeziehungen im CAD-Modell widerspiegelt.

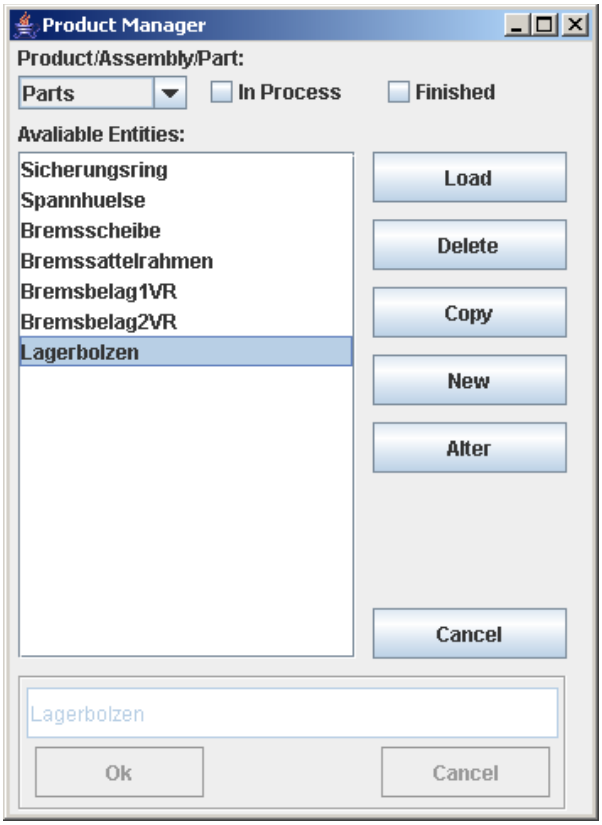


Abbildung 72: Oberfläche des Product Manager mit den vorhandenen Einzelteilen des Produkts "Radbremse " und der identifizierten Produktkomponente „Lagerbolzen“

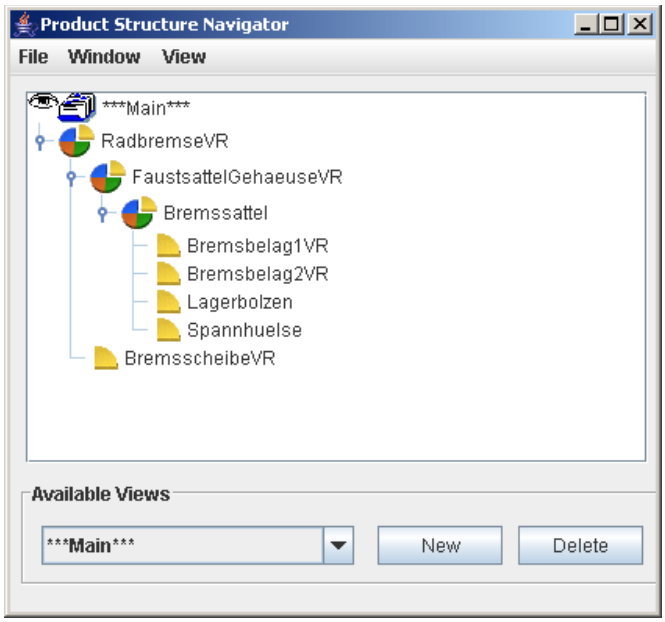


Abbildung 73: Oberfläche des Product Structure Navigator mit identifizierten Produktkomponenten

Nicht identifizierte Elemente des CAD-Modells konnten entweder nicht erkannt und müssen daher manuell verknüpft werden, oder sie sind noch gar nicht im SAEP-internen Modell enthalten und müssen neu erzeugt werden.

Um vorhandene, aber nicht identifizierte Elemente miteinander zu verknüpfen, werden diese im *Product Manager* ausgewählt und in die CAD-Sicht im *Product Structure Navigator* einkopiert. Dies geschieht analog zur Vorgehensweise bei der manuellen Modellierung von Produktstrukturen (siehe Abschnitt 5.1.3). Die Stelle im Produktstrukturbaum muss vom Benutzer manuell anhand der Produktstruktur im *BrepBrowser* ermittelt werden. Im *Product Structure Navigator* wird danach die eingefügte Komponente kopiert und ihr Äquivalent im BrepBrowser markiert. Per Mausklick werden die beiden Komponenten dann miteinander verknüpft (Befehl „Copy reference“ im Kontextmenü, siehe Abbildung 71).

Nicht im SAEP-Modell vorhandene Elemente werden in der Brep-Baumstruktur innerhalb des *BrepBrowser* ausgewählt, in die Zwischenablage kopiert (Befehl „Copy reference“ im Kontextmenü, siehe Abbildung 71) und im *Product Structure Navigator* an der entsprechenden Stelle eingefügt. Intern wird dadurch ein neues Element in SAEP erzeugt und mit dem in der Zwischenablage gespeicherten Element aus dem *BrepBrowser* verknüpft.

Werden Schablonenobjekte mit Elementen aus dem *BrepBrowser* verknüpft, ist die Vorgehensweise identisch mit der oben beschriebenen. Sind Schablonenobjekte einmal erstellt, können sie für eine automatische Verknüpfung der externen Produktkomponenten herangezogen werden, wenn diese konstante Bestandteile einer durch die Schablone definierten Produktfamilie sind. Die Verknüpfung von Schablonenobjekten mit externen Modellentitäten bietet sich naturgemäß vor allem bei Standardteilen an.

5.2.2 Verknüpfung von korrespondierenden Produktmerkmalen

Sind korrespondierende Produktkomponenten erkannt und miteinander verknüpft, können deren Merkmale in Beziehung zueinander gebracht werden. Hierbei gibt es zwei Möglichkeiten:

- Direkte Verknüpfung von zwei Merkmalen: zwei Merkmale, beispielsweise eine explizit gegebene Produkteigenschaft und eine Anforderung, können direkt auf einander abgebildet werden. Dies ist beispielsweise bei dem Material eines Einzelteils gegeben.
- Indirekte Verknüpfung von mehreren Merkmalen: Wie in Abschnitt 3.2.6 dargestellt, ist es notwendig, implizit gegebene Eigenschaften eines Produkts berücksichtigen zu können. Beispielsweise ergibt sich die räumliche Gesamtausdehnung einer Baugruppe in einer bestimmten Richtung aus den Gesamtlängen ihrer Einzelteile in dieser Richtung unter Berücksichtigung ihrer Koordinaten im Raum.

Bei der direkten Verknüpfung von zwei Merkmalen werden in SAEP die Bedingungen für eine Übereinstimmung festgelegt. Dies geschieht durch die Definition des zu verknüpfenden Merkmals, das vom Benutzer mit Hilfe des *Attribute Library Manager* aus der Merkmalsbibliothek zuvor ausgewählt und der entsprechenden Produktkomponente im *Product Structure Navigator* zugeordnet wurde. Dabei kann definiert werden, ob entweder die Bezeichnung eines Merkmals relevant ist, seine Maßeinheit bei einem quantitativen Merkmal oder die Kombination aus beidem. Sind Merkmale vorhanden, die durch SAEP nicht verknüpft werden konnten, müssen diese manuell durch den Benutzer verknüpft werden.

Für die Darstellung von einander zugeordneten Merkmalen und die manuelle Verknüpfung ist der *Relation Network Navigator* zuständig. Abbildung 74 zeigt seine Benutzungsoberfläche mit den aus dem CAD-Modell extrahierten Produkteigenschaften und ihren Verknüpfungen mit den zuvor im *Product Structure Navigator* definierten Anforderungen.

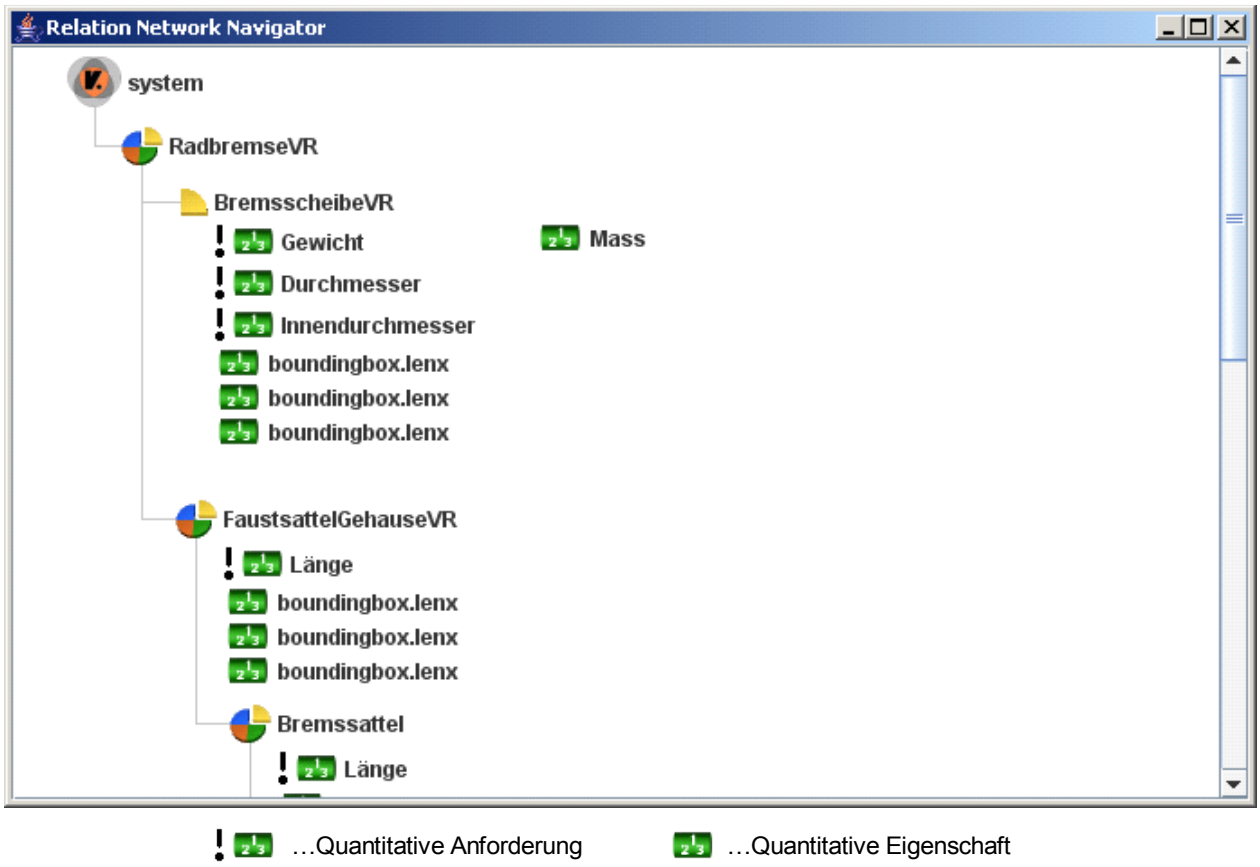


Abbildung 74: Benutzungsoberfläche des Relation Network Navigator

Wie in Abbildung 74 zu erkennen ist, sind nicht alle Eigenschaften mit Anforderungen verknüpft und nicht alle Anforderungen mit Eigenschaften. Die Eigenschaft „Mass“ (=„Gewicht“) konnte anhand der vorgegebenen Maßeinheit automatisch zugeordnet werden und wird daher rechts neben der entsprechenden Anforderung („Gewicht“) in dem Diagram angezeigt. Die fehlenden Verknüpfungen können aber manuell vom Benutzer erstellt werden, indem im *Relation Network Navigator* mit dem Befehl „Create Relation“ im Kontextmenü eine Verbindung zwischen den entsprechenden Merkmalen erzeugt wird. Abbildung 75 zeigt die Benutzungsoberfläche des *Relation Network Navigators* beim Markieren der zu verknüpfenden Merkmale und den Dialog zur Erzeugung einer Verknüpfungsbeziehung zwischen Anforderung und Eigenschaft. Name und Kommentar können dabei optional angeben werden.

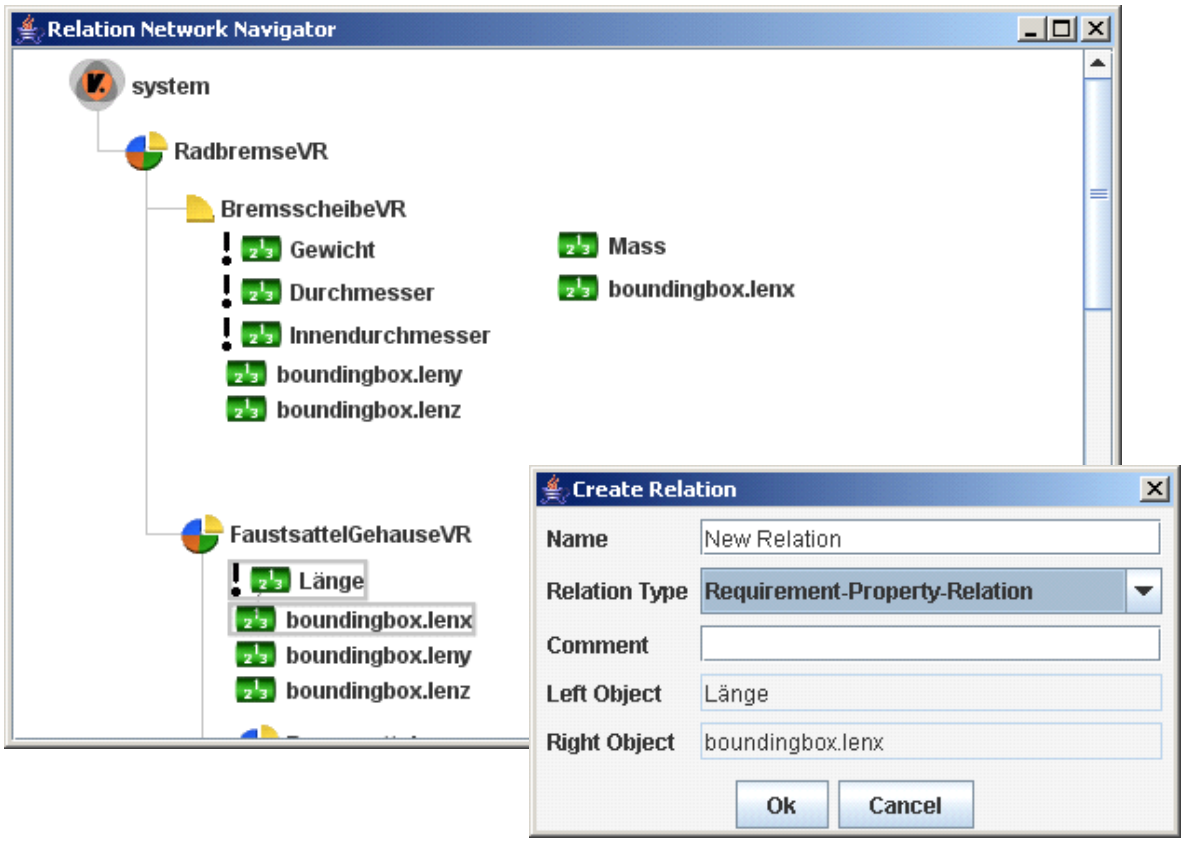


Abbildung 75: Manuelle Erzeugung einer Verknüpfungsbeziehung zwischen einer Anforderung und einer Produkteigenschaft

Anhand des Diagramms, das die mit Anforderungen verknüpften Eigenschaften einander gegenüberstellt, ist sofort ersichtlich, für welche Anforderung noch keine Eigenschaft, bzw. für welche Eigenschaft noch keine Anforderung definiert ist (Diese stehen allein in einer Zeile). Die mit Anforderungen verknüpften Eigenschaften können nun anhand dieser auf Knopfdruck evaluiert werden. Abbildung 76 zeigt den Dialog mit dem Ergebnis der Evaluierung für die Komponente „Bremsscheibe“. In der Tabelle sind die Ausprägungen der Eigenschaften mit den Ausprägungen der zugeordneten Anforderungen über einen Operator verknüpft. Dieser ergibt sich aus der Art der Anforderung. Ist ein exakter Wert für die Eigenschaft durch die Anforderung vorgegeben, wird der Operator „=“ für die Überprüfung eingesetzt. Ist eine Anforderung zugeordnet, die einen zulässigen Wertebereich für die Eigenschaft vorgibt, werden entsprechende Vergleichsoperatoren wie „<“, „>“, „>=“ oder „<=“ eingesetzt.

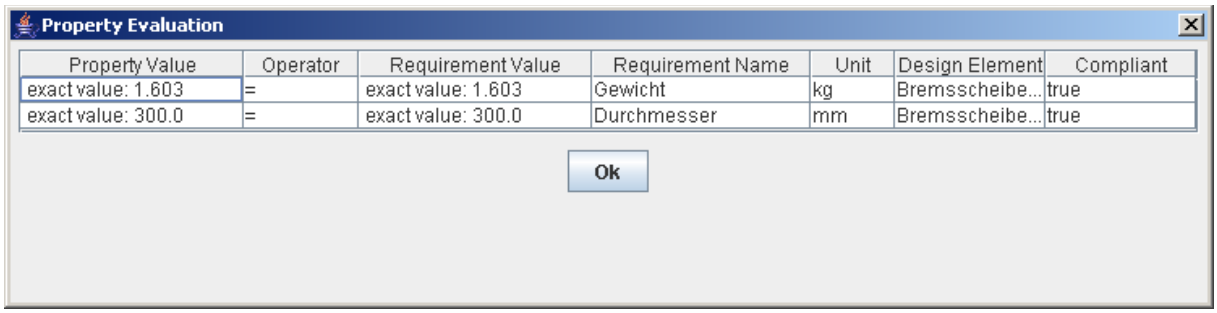


Abbildung 76: Dialog zur Evaluierung von Eigenschaften anhand ihrer zugeordneten Anforderungen

Im obigen Beispiel konnte eine Anforderung direkt mit einer explizit gegebenen Eigenschaft verknüpft werden (Anforderung „Durchmesser“ mit Eigenschaft „boundingbox.lenx“). Der Innendurchmesser der Bremsscheibe ist aber beispielsweise nicht explizit im CAD-Modell als Parameter ablesbar, sondern muss über eine geeignete Berechnungsvorschrift errechnet werden. Diese können mit dem im nächsten Abschnitt beschriebenen Constraintmodellierer definiert werden

5.3 Modellierung von Abhängigkeiten

5.3.1 Manuelle Modellierung von Abstrakten Beziehungen

Mit Hilfe der Benutzungsoberfläche des *Relation Network Navigators* können Beziehungsnetzwerke in Produktstrukturen definiert werden. Dabei sind die Merkmale der Produktkomponenten Knoten, die durch Abstrakte Abhängigkeiten (*AbstractRelation*, siehe Abschnitt 4.1.5, Seite 86) als Kanten miteinander verbunden sind.

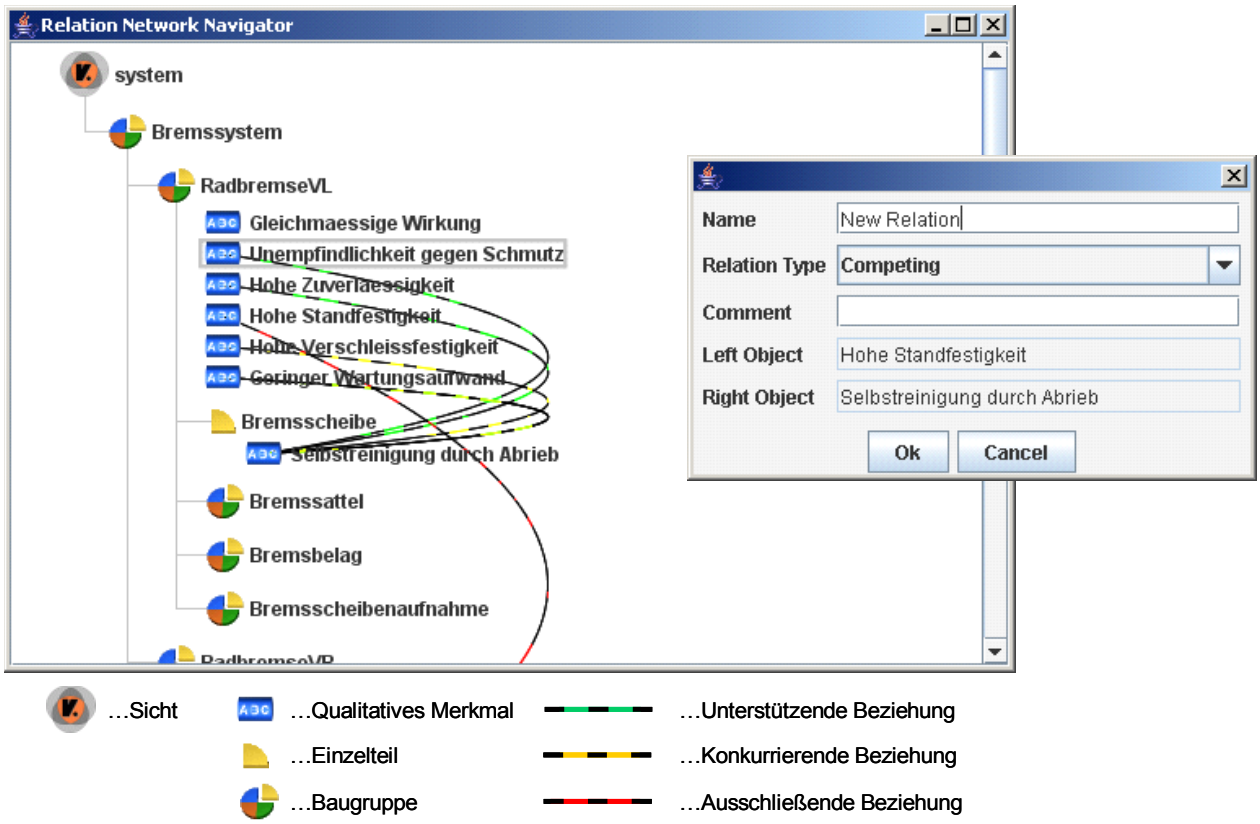


Abbildung 77: Programmoberfläche zur Erzeugung von und Navigation in Beziehungsnetzwerken

In Abbildung 77 sind einige Abhängigkeiten zwischen qualitativen Produktmerkmalen einer Radbremse dargestellt. Das Prinzip der Selbstreinigung einer Bremsscheibe durch den Abrieb bei einer Bremsung hat unterschiedliche Auswirkungen auf andere, allgemeine Merkmale der Radbremse. Sie dient z.B. einer hohen Zuverlässigkeit der gesamten Radbremse. Dafür muss aber ein höherer Verschleiß in Kauf genommen werden.

Um eine neue Beziehung zwischen zwei Produktmerkmalen zu erzeugen, muss der Benutzer zunächst durch einen Klick mit der rechten Maustaste ein Produktmerkmal auswählen und mit der Maus über das Produktmerkmal fahren, das mit dem ersten Merkmal verknüpft werden soll. Nach Loslassen der Maustaste zeigt das System einen Dialog zur Erzeugung einer neuen Beziehung an. In diesem Dialog kann der Benutzer optional eine Bezeichnung und einen Kommentar für die Beziehung eingeben und die Art der Beziehung aus einer Liste auswählen (Konkurrierend, Ausschließend oder Unterstützend). Die Art der Beziehung wird durch eine farbliche Markierung der gebogenen Beziehungskante hervorgehoben. Grün-schwarze Kanten zeigen Unterstützende Beziehungen an, gelb-schwarze Kanten Konkurrierende Beziehungen und rot-schwarze Kanten Ausschließende Beziehungen.

Analog zum Produktstruktur-Navigator können im *Relation Network Navigator* die Produktstrukturkomponenten auch Schablonenobjekte sein. Die Modellierung von Schablonenobjekten, deren Merkmalen, sowie den Abhängigkeiten zwischen diesen, erfolgt ebenso wie bei konkreten Produktstrukturelementen. Auch hier können aus den Schablonenobjekten konkrete

Baugruppen oder Einzelteile erzeugt werden. Zusätzlich zu den Strukturelementen und deren zugeordneten Merkmalen werden bei der Instanziierung einer Schablone die Abhängigkeiten zwischen diesen kopiert und übernommen.

5.3.2 Manuelle Modellierung und Auswertung von mathematisch beschreibbaren Abhängigkeiten zwischen Produktmerkmalen

Zwischen Produktmerkmalen, die verschiedenen Produktkomponenten zugeordnet sind, können mathematisch beschreibbare Abhängigkeiten definiert werden (siehe Abschnitt 4.1.5, Seite 88). Dazu wird die Programmoberfläche des *Constraint-Modellierers* (*Constraint Modeler*) benutzt. In ihm können ausgehend von einem Wurzelknoten (*Root*) Gleichungen bzw. ganze Gleichungssysteme modelliert werden.

In dem Beispielszenario soll der in Abschnitt 5.2.2 erwähnte Innendurchmesser der Bremsscheibe berechnet werden. Dazu muss zunächst die Gleichung zur Berechnung des Wertes erstellt werden. Es bietet sich an, den Wert mit Hilfe der Länge der Innenkante (Innenumfang) zu bestimmen, da diese als Parameter direkt aus dem CAD-Modell abgelesen werden kann. Dieser ergibt sich allgemein als $D = U/\pi$. Da sich bei dem betreffenden Modell, wie anhand des markierten Flächenelements in Abbildung 78 ersichtlich, der Innenumfang aus zwei offenen Halbzylindern zusammensetzt[59], muss der Wert der Kantenlänge verdoppelt werden. Das bedeutet, die korrekte Gleichung ist in diesem Fall $D = 2*edge.length/\pi$, mit „edge.length“ als Länge der Innenkante.

59 Dies ist eine Eigenart des verwendeten CAD-Systems Pro/Engineer.

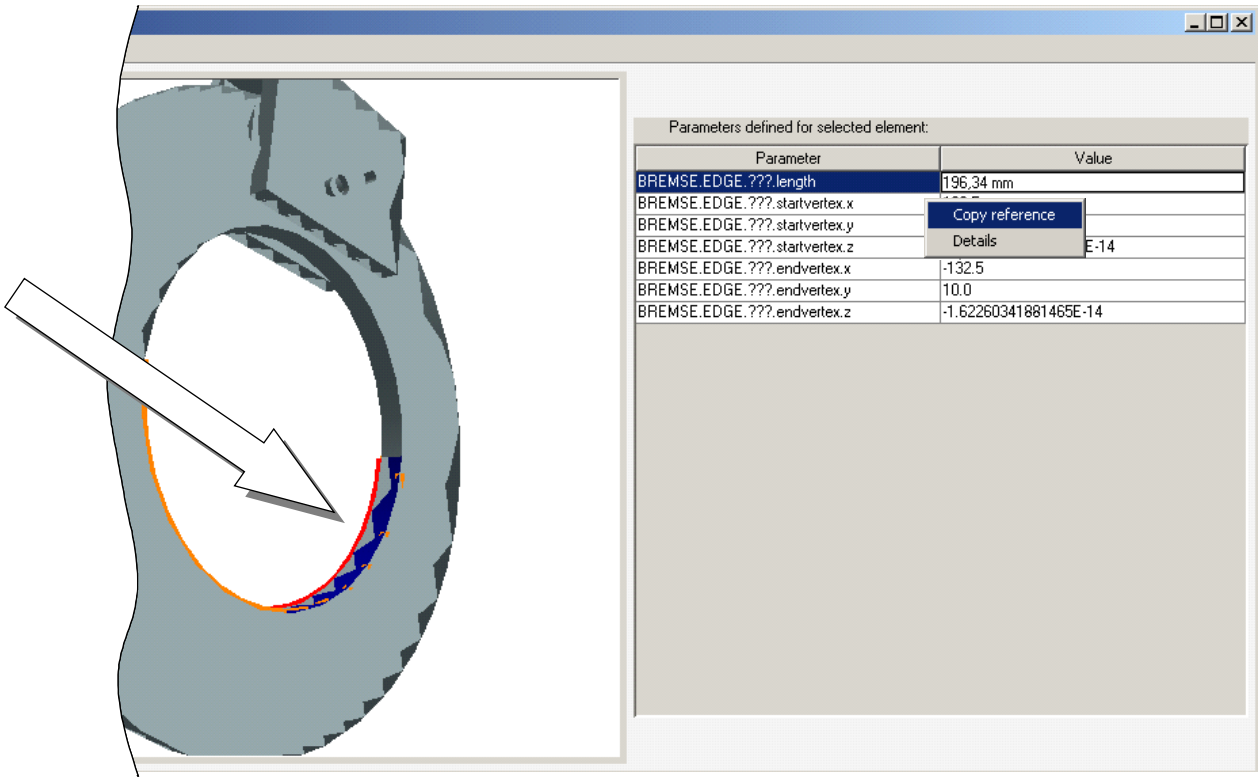


Abbildung 78: Auswahl eines Parameters im Brep Browser

Die oben genannte Formel wird mit Hilfe der Benutzungsoberfläche des Constraint-modellierers zusammengesetzt. Den Knoten des Binärbaums werden dazu die entsprechenden Operatoren bzw. Konstanten oder Produktmerkmale zugeordnet. Operatoren bilden neue Knoten in dem Baum, während Konstanten und Produktmerkmale die Blätter des Baums darstellen. Der Benutzer wählt dazu den entsprechenden Knoten aus und fügt diesem einen Operator, bzw. eine Konstante oder ein vorher ausgewähltes Produktmerkmal hinzu. Soll eine neue Gleichung eingefügt werden, ist der Wurzelknoten Ausgangspunkt. Abbildung 79 zeigt die Benutzungsoberfläche des Constraint-Modellierers mit der modellierten Gleichung. Der Wert der Kantenlänge wurde dabei direkt aus dem Geometriemodell im *Brep Browser* übernommen.

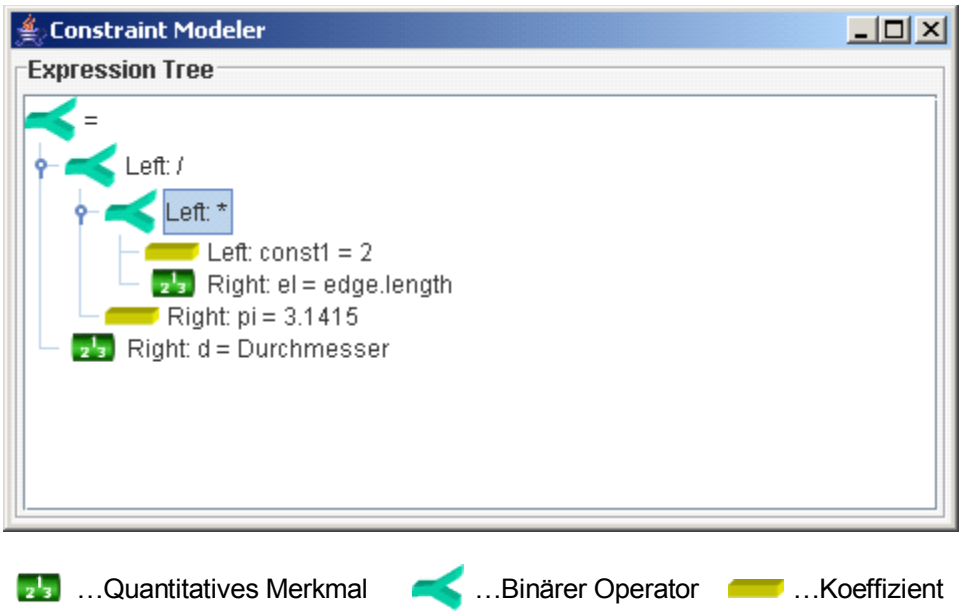


Abbildung 79: Benutzungsoberfläche des Constraint Modellierers

Beim Hinzufügen eines neuen Knotens innerhalb einer Gleichung (also eine Ebene unterhalb des Wurzelknotens) muss der Benutzer entscheiden, ob der neue Knoten als linke oder rechte Seite des darüber liegenden Binären Operators eingefügt werden soll. Produktmerkmale werden direkt in der Benutzungsoberfläche des *Brep Browser*, des *Relation Network Navigators* oder aus einer Merkmalsbibliothek im *Attribute Library Manager* ausgewählt und als Wert an die entsprechende Stelle des Binärbaums eingefügt. Soll ein Wert oder seine Maßeinheit verändert werden, kann der Benutzer dies in dem Dialog nachträglich durchführen. Abbildung 80 zeigt den Dialog für das Einfügen eines zuvor ausgewählten Produktmerkmals.

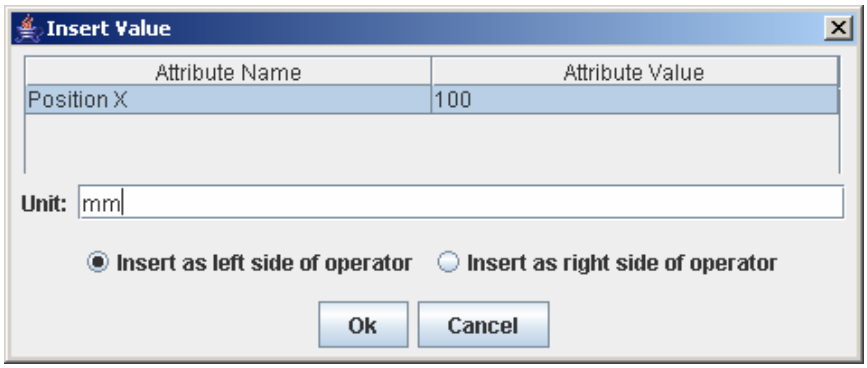


Abbildung 80: Dialog für das Einfügen eines Produktmerkmals

Beim Einfügen eines Binäroperators muss sein mathematisches Symbol angegeben werden, das aus einer Auswahlliste gewählt wird. Zusätzlich muss die Seite des ihm im Binärbaum übergeordneten Operators angegeben werden, auf der der Operator eingefügt werden soll. Abbildung 81 zeigt den Benutzerdialog zum Einfügen eines neuen Operators.

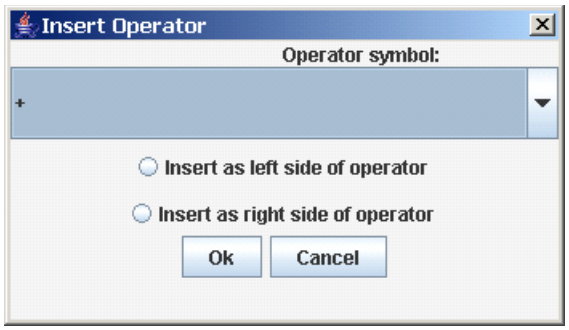


Abbildung 81: Benutzerdialog für das Einfügen eines binären Operators

Die Auswertung von Gleichungssystemen erfolgt mit Hilfe des in SAEP integrierten Constraintsolverpakets *IA_Solver* von T.J.Hickey (siehe auch Abschnitt 4.5.1). Für den Constraintsolver wurde eine Oberfläche implementiert, die vor allem zu Überprüfungszwecken dient. Die Ergebnisse der Auswertung werden primär intern benutzt, um nicht-ausgeprägte Merkmale mit den errechneten Werten zu versehen. Abbildung 82 zeigt die Oberfläche des Constraintsolvers. In der Abbildung ist das Ergebnis der Berechnung des Innendurchmessers zu sehen (unterstes Textfeld, Parameter „d“). Der Wert ist intern bereits als Ausprägung der Eigenschaft übernommen worden und kann mit einer entsprechenden Anforderung im *Relation Network Navigator* verknüpft werden.

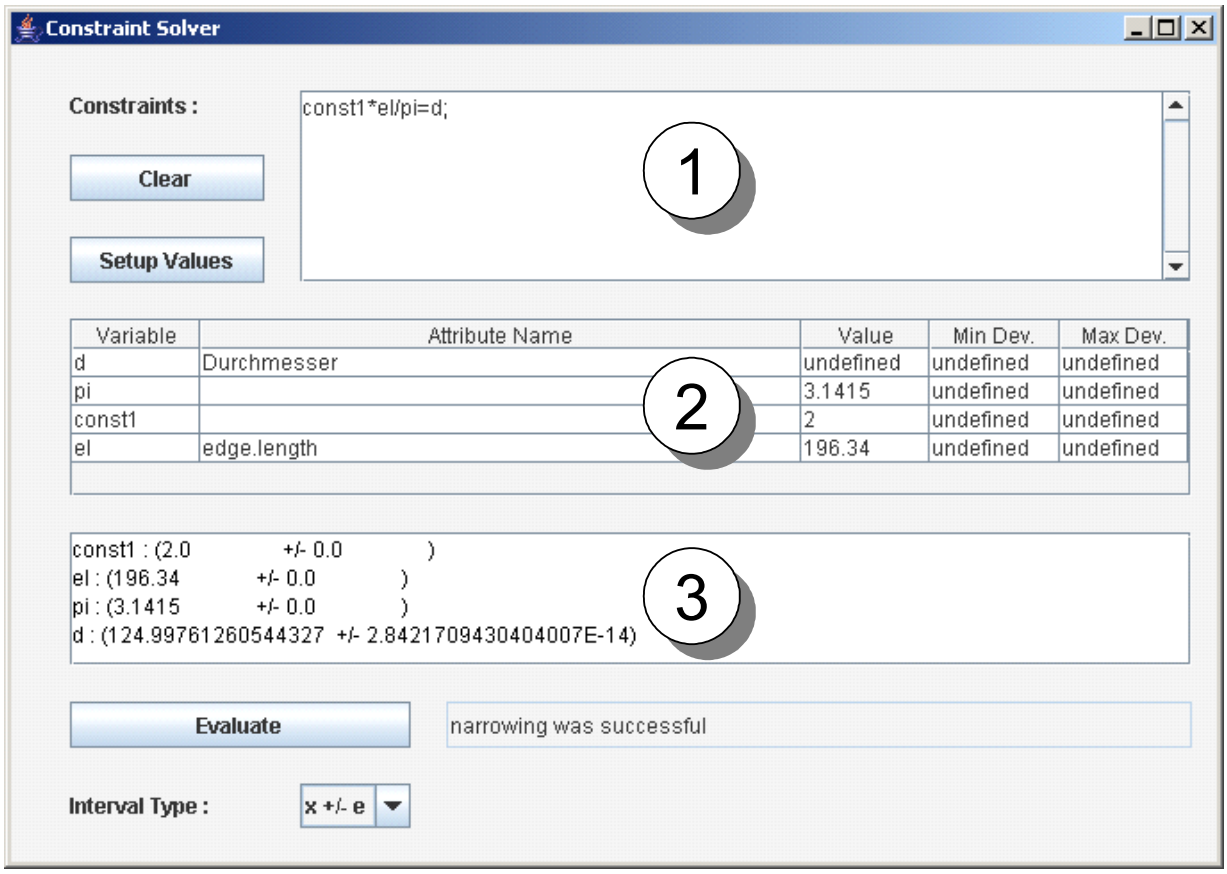


Abbildung 82: Oberfläche des Constraintsolvers

Die Benutzungsoberfläche bietet drei Hauptanzeigeelemente (von oben nach unten):

1. Ein Textfeld mit der Textrepräsentation des Binärbaums, bzw. der Binärbäume, aus denen ein Gleichungssystem aufgebaut ist. Sie wird aus den Binärbäumen abgeleitet. In der vorliegenden Version der Software werden eventuell notwendige Klammern ausgelassen. Dies kann die Überprüfung der Korrektheit einer Formel durch den Benutzer zwar erschweren, hat aber keine Auswirkung auf die Korrektheit der Auswertung, da die Basis für die Berechnungen die Binärbäume sind, die vom Benutzer angelegt wurden. In den Binärbäumen ist die Rangfolge der Operationen eindeutig.
2. Eine Tabelle mit den verwendeten Parametern und deren Ausprägung vor der Evaluierung. In der Tabelle ist in der ersten Spalte die in der Gleichung verwendete Variable bzw. der Parametername vermerkt. In der zweiten Spalte ist der Name des jeweiligen Produktmerkmals eingetragen. In den letzten drei Spalten ist die Ausprägung der Parameter angegeben.
3. Ein Textfeld mit den Werten der Parameter nach der Evaluierung des Gleichungssystems. Zu erkennen ist in dem Textfeld, dass die vorher nicht ausgeprägte Variable „d“ der Gleichung mit dem Wert „124,99761260544327“ und einer geringen Abweichung errechnet wurde[60]. Da der Constraintsolver intern nur mit Intervallen arbeitet, sind die Werte in dem Textfeld ebenfalls als Wertebereich dargestellt. Es kann umgeschaltet werden zwischen einer Intervalldarstellung (z.B. [10, 12]) und einer Darstellung durch einen Mittelwert mit symmetrischer oberer und unterer Abweichung, wie sie in Abbildung 82 dargestellt ist.

5.3.3 Verwaltung, Modellierung und Auswertung von Kontaktsystemen

In der vorliegenden Implementierung werden die Komponenten von Wirkräumen durch die Definition von geeigneten Sichten auf die Produktstruktur im *Product Structure Navigator* definiert. Die Funktionalität zur Verwaltung und Modellierung von Kontaktsystemen ist durch die Benutzungsoberfläche des Kontaktsystemmanagers (*Contact System Manager*) realisiert. Er wird durch einen entsprechenden Befehl im Kontextmenü des *Product Structure Navigator* aufgerufen. um den dort ausgewählten Views Kontaktsysteme zuzuordnen.

5.3.3.1 Verwaltung von Kontaktsystemen

Kontaktsysteme werden in Kontaktsystembibliotheken verwaltet. Dazu werden sie in geeignete Kategorien eingeteilt. Diese unterstützen den Benutzer bei der Auswahl der für einen bestimmten Wirkraum relevanten physikalischen oder sonstigen Zusammenhänge[61]. Die Bibliotheken werden als Baumstruktur dargestellt. Die Kategorien bilden die Knoten des Baums, während die Kontaktsysteme in den Blättern des Baums enthalten sind. Abbildung 83 zeigt

60 Die Abweichung ist durch die internen Intervallberechnungsverfahren des verwendeten Constraintsolverpakets bedingt.

die Oberfläche zur Verwaltung von Kontaktsystemen in Bibliotheken. Außerdem sind in Abbildung 83 die Dialoge zur Erzeugung von Kategorien und zur Definition der initialen Informationen von Kontaktsystemen dargestellt, d.h. eine Bezeichnung und ein Kommentar. Um eine neue Kategorie bzw. ein neues Kontaktsystem anzulegen, wählt der Benutzer den Knoten im Baum aus, dem das neue Element hinzugefügt werden soll. Durch die Betätigung der rechten Maustaste kann der Benutzer die gewünschte Aktion auslösen. Neben der Erzeugung von Kategorien und Kontaktsystemen können bestehende Elemente des Baumes auch wieder gelöscht werden. Die Struktur eines Kontaktsystems wird mit Hilfe eines speziellen Editors modelliert, dessen Oberfläche rechts neben der zur Darstellung der Kontaktsystembibliothek angeordnet ist. Durch Wahl des Menüpunktes „Edit Contact System“ wird das gewählte Kontaktsystem in den Editor geladen und kann dort modelliert werden.

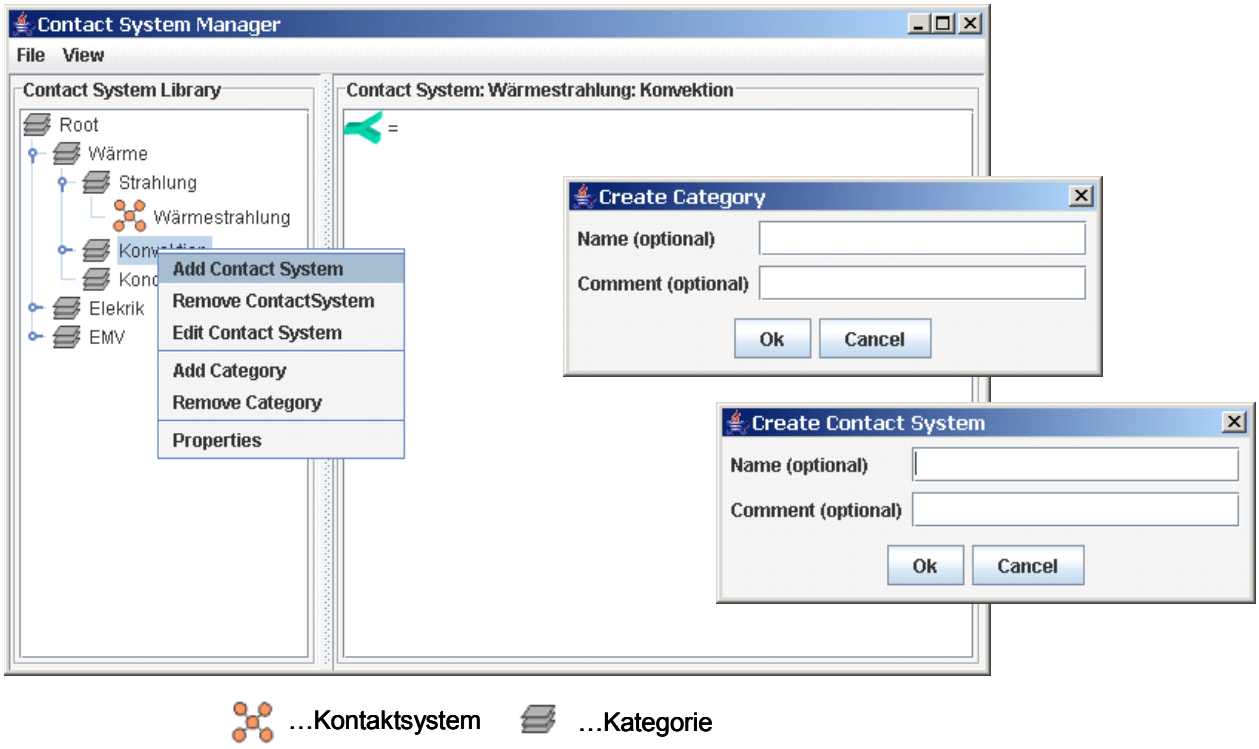


Abbildung 83: Oberfläche zur Verwaltung von Kontaktsystembibliotheken

5.3.3.2 Modellierung von Kontaktsystemen

Für die Modellierung von Kontaktsystemen wird ebenfalls die Programmoberfläche des Constraint-Modellierers (*Constraint Modeler*) benutzt[62], denn die Komponenten der Oberfläche sind sowohl für die Definition von einfachen Gleichungen geeignet, die mathematisch

61 Kontaktsysteme eignen sich nicht nur für die Repräsentation physikalischer Zusammenhänge. Es sind auch andere Anwendungsfälle denkbar. Siehe dazu auch Abschnitt 5.4.

62 Die Benutzungsoberfläche des Constraint-Modellierers ist in einem sogenannten *JPanel* gekapselt. JPanels sind Container-Objekte, die beliebige Bedienelemente beinhalten können und ihrerseits in beliebige Container-Objekte eingefügt werden können. Dadurch können geschachtelte Oberflächen zusammengestellt werden.

beschreibbare Abhängigkeiten zwischen Produktmerkmalen beschreiben, als auch für die Definition von Kontaktsystemen. Je nach Anwendungsfall wird eine andere, darunter liegende Komponente (der sogenannte *Controller*) von der Oberfläche angesteuert, die die jeweiligen Konstrukte zusammensetzt und in der Persistenzschicht abspeichert. Bei den Komponenten der Oberfläche werden je nach Anwendungsfall die Menüs und Dialoge ausgetauscht, die die entsprechenden Aktionen auslösen.

Bei der Modellierung von Kontaktsystemen wird zunächst die Rumpfgleichung in Form eines Binärbaums aus mathematischen Ausdrücken aufgebaut. Zusätzlich werden die Stellen im Baum markiert, an denen die Kontaktsystemgleichung um die Eigenschaften neu zum Wirkraum hinzugefügter Komponenten erweitert wird. Diese können entweder weitere mathematische Operatoren beinhalten (Knoten im Baum) oder direkt mit Produktmerkmalen gekoppelt werden (Blätter des Baums). An den Knoten werden binäre *CSPort*-Objekte eingefügt. Sie basieren auf Binären Operatoren. Das dort spezifizierte Operatorsymbol gibt an, wie neue Äste eingefügt werden sollen (beispielsweise mittels einer Addition). Abbildung 84 zeigt die Benutzungsoberfläche zur Modellierung von Kontaktsystemen. Den Symbolen für die graphische Repräsentation der im Binärbaum vorhandenen Objekte (Binäre Operatoren, Binäre CSPorts etc.) sind rechts die jeweiligen mathematischen Symbole zugeordnet sowie die Seite des binären Operators, an der sie stehen (*Left, Right*).

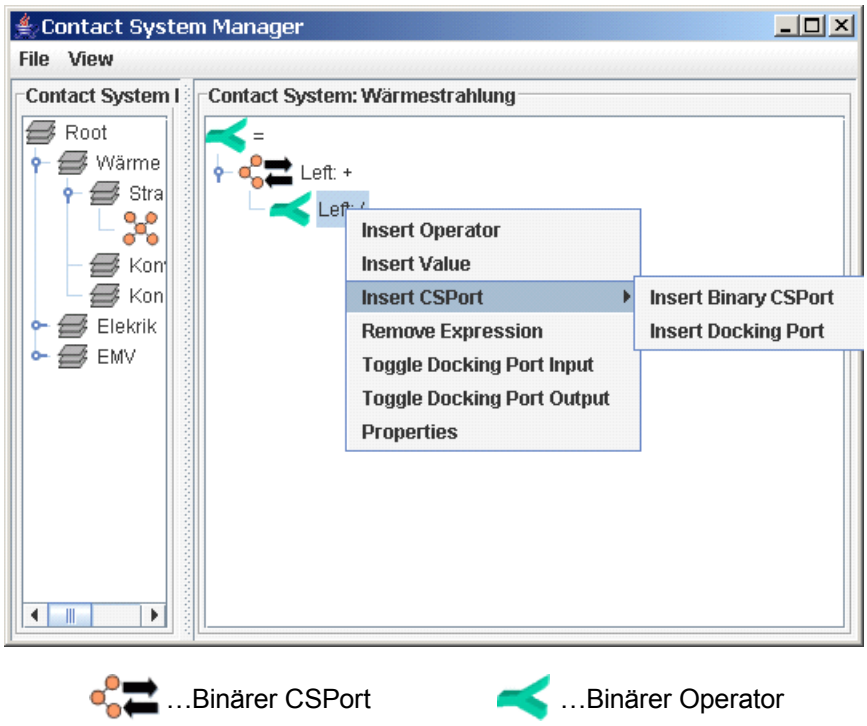


Abbildung 84: Benutzungsoberfläche zur Modellierung von Kontaktsystemen

Wird ein neuer Ausdruck, ein binäres *CSPort*-Objekt oder ein Input- oder Output-Port unter einen Knoten des Baums eingefügt, muss der Benutzer die Seite des binären Operators angeben, an der das neue Objekt eingefügt werden soll. Für die verschiedenen Elemente, die eingefügt werden können, sind spezielle Benutzerdialoge vorgesehen, in denen die erforderli-

chen Informationen angegeben werden können (für binäre Operatoren und Produktmerkmale als Koeffizienten siehe Dialoge in Abschnitt 5.3.2).

Soll ein *CSPort*-Objekt eingefügt werden, so muss der Operator angegeben werden, mit dem ein neuer Ast bei Hinzufügen einer neuen Produktkomponente in den Wirkraum mit dem vorhandenen Binärbaum verknüpft werden soll. Außerdem muss, wie bei allen Elementen des Binärbaums, angegeben werden, auf welcher Seite des übergeordneten Binäroperators das *CSPort*-Objekt einzufügen ist. Zusätzlich kann bei einem *CSPort*-Objekt optional eine Bezeichnung und ein Kommentar angegeben werden. Abbildung 85 zeigt den Dialog für das Erzeugen eines *CSPort*-Objekts.

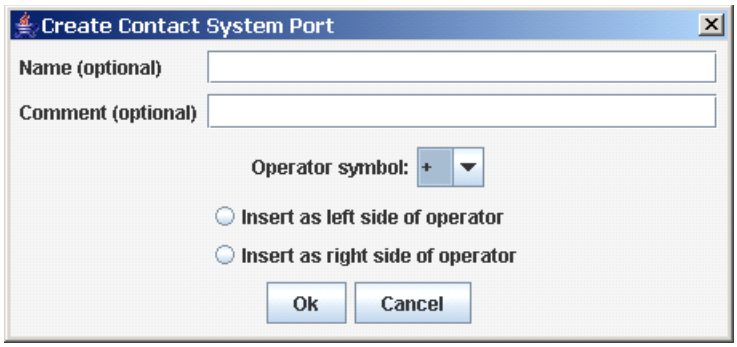


Abbildung 85: Dialog für das Erzeugen eines CSPort-Objekts

An den Blättern des Binärbaums werden sogenannte Input-, bzw. Output-Ports eingefügt, die direkt mit Produktmerkmalen verknüpft werden. Input-Ports nehmen Eigenschaften der hinzuzufügenden Komponenten auf, die die Ausgangseigenschaft eines Kontaktsystems beeinflussen. Output-Ports nehmen Eigenschaften auf, die notwendig sind, um die Ausgangseigenschaft, beispielsweise an einer bestimmten Stelle im Raum, zu berechnen. Nimmt man als Beispiel ein Kontaktsystem zur Bestimmung der Wärmestrahlung an einem bestimmten Punkt im (Wirk-)Raum, so werden die Koordinaten der Strahler sowie ihre Strahlungsintensität in den Input-Ports abgelegt. Die Koordinaten der Komponenten, die der Wärmestrahlung ausgesetzt werden, werden in den Output-Ports abgelegt. In den Input- und Output-Ports muss definiert werden, welche Bezeichnung und welche Maßeinheit ein Produktmerkmal tragen muss, um an dieser Stelle eingefügt zu werden. Dazu werden die in dem vorher aus der Merkmalsbibliothek ausgewählten Produktmerkmal bereits enthaltenen Informationen als Voreinstellung übernommen, die aber vom Benutzer angepasst werden können. Zusätzlich gibt der Benutzer an, ob es sich bei dem Port um einen Input- oder einen Output-Port handelt. Des Weiteren muss der Benutzer angeben, ob ein Port als Eigenschaft des Kontaktsystems aufgefasst wird oder als Anforderung, die durch das Kontaktsystem definiert wird. Im Falle einer Anforderung als Output-Port wird bei einer Auswertung des Kontaktsystems die Anforderung durch Berechnung ausgeprägt. Diese wird bei der Auswertung durch das System automatisch mit der mit dem Port verknüpften Eigenschaft in Beziehung gebracht. Dadurch entsteht eine Bedingung, der die Komponente genügen muss, bzw. eine Bedingung, der die Umgebung einer Komponente genügen muss (siehe auch Abschnitt 2.1.2 und Abschnitt 3.3.5). Diese Unterscheidung wird aber durch den Benutzer getroffen, indem er das Ergebnis

entsprechend interpretiert und die entsprechenden Konsequenzen trifft. Entweder passt der Benutzer die Umgebung an die Komponente an (Komponente kann nicht alternativ ausgeführt werden) oder er passt die Komponente an die Umgebung an. Für das System ist dieser Unterschied nicht relevant.

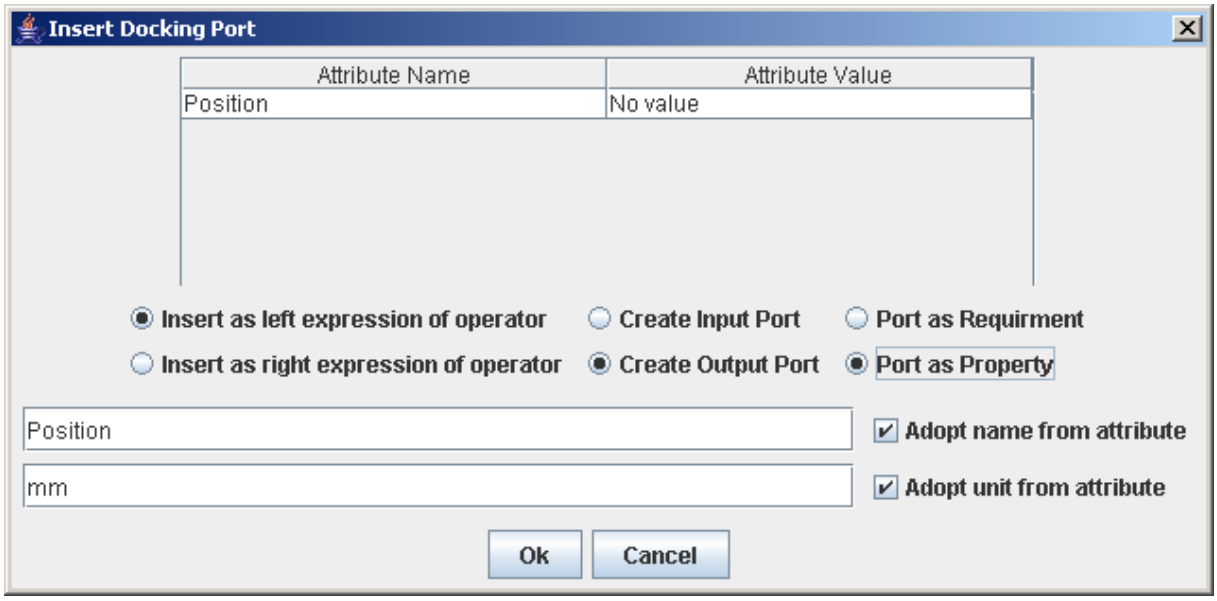


Abbildung 86: Dialog zur Erzeugung eines Ports

CSPort-Objekte und Input-, bzw. Output-Ports werden bei der Erstellung des Kontaktsystems einmalig angelegt. Bei jeder Erweiterung des Wirkraums werden die binären *CSPort*-Objekte geklont, mit den Eigenschaften der neuen Komponenten belegt und die geklonten Objekte mit dem entsprechenden Operator in den Baum gehängt[63].

Im Rahmen des Szenarios wird der ABS-Controller in den Motorraum eingebaut. Der Controller benötigt für eine ausreichende Wärmeabfuhr ein Milieu mit entsprechender Wärmeabfuhrkapazität. Im vorliegenden Szenario wird ein Wärmetransport über Strahlung betrachtet. Die physikalischen Zusammenhänge sind bewusst vereinfacht dargestellt, da dies zum einen einem besseren Verständnis der Funktionsweise des Systems dient und zum anderen exakte Berechnungen ohnehin in dedizierten Werkzeugen bzw. mit spezialisierten Methoden durchgeführt werden müssen. Die Wärmeabfuhr über die Umgebungsluft ist abhängig von ihrer Temperatur. Diese wird näherungsweise durch die Umgebungsluft des Fahrzeugs, durch den Wärmeübergang über Karosserieteile, wie Motorhaube oder Kotflügel sowie insbesondere durch die Wärmeabstrahlung des Motorblocks bestimmt. Für eine exakte Bestimmung der Temperatur müssen außerdem die Strömungsverhältnisse im Motorraum berücksichtigt werden (z.B. Berücksichtigung von Zonen mit Wärmestau). Diese werden aber im Szenario aufgrund ihrer Komplexität vernachlässigt. Des Weiteren wird der Wärmeeintrag der Ka-

63 Wird die erste Komponente eingefügt, werden die bereits angelegten Kontaktsystemports mit den Eigenschaften der Komponente belegt.

rosserieteile nicht durch das System berechnet (Abhängigkeit von der Sonneneinstrahlung etc.), sondern als einfache Eigenschaft der betreffenden Teile angegeben.

Aus den oben beschriebenen Sachverhalten lässt sich ein Kontaktsystem aufbauen, das die Intensität der Wärmestrahlung an einer bestimmten Stelle im Motorraum angibt. Die Wärmeabgabeeigenschaften der Komponenten sowie deren Positionen relativ zueinander werden in die Berechnungen einbezogen. Die Wärmeaufnahme des Controllers wird als Bedingung definiert, die von der Position des Controllers relativ zu den Wärmequellen abhängig ist. Die durch das Kontaktsystem beschriebene Wärmestrahlung wird intern mit der Bedingung des Controllers verknüpft. Abbildung 87 zeigt das Kontaktsystem zur Beschreibung von Wärmestrahlungsfeldern mit entsprechenden Kontaktsystem-Ports. Man erkennt rechts auf der Darstellung einen Binärbaum, der einen mathematischen Zusammenhang zwischen der räumlichen Position und der Wärmeausstrahlung einer (punktförmigen) Wärmequelle und der resultierenden Wärmestrahlung an einem Punkt P in der Umgebung der Wärmequelle herstellt.

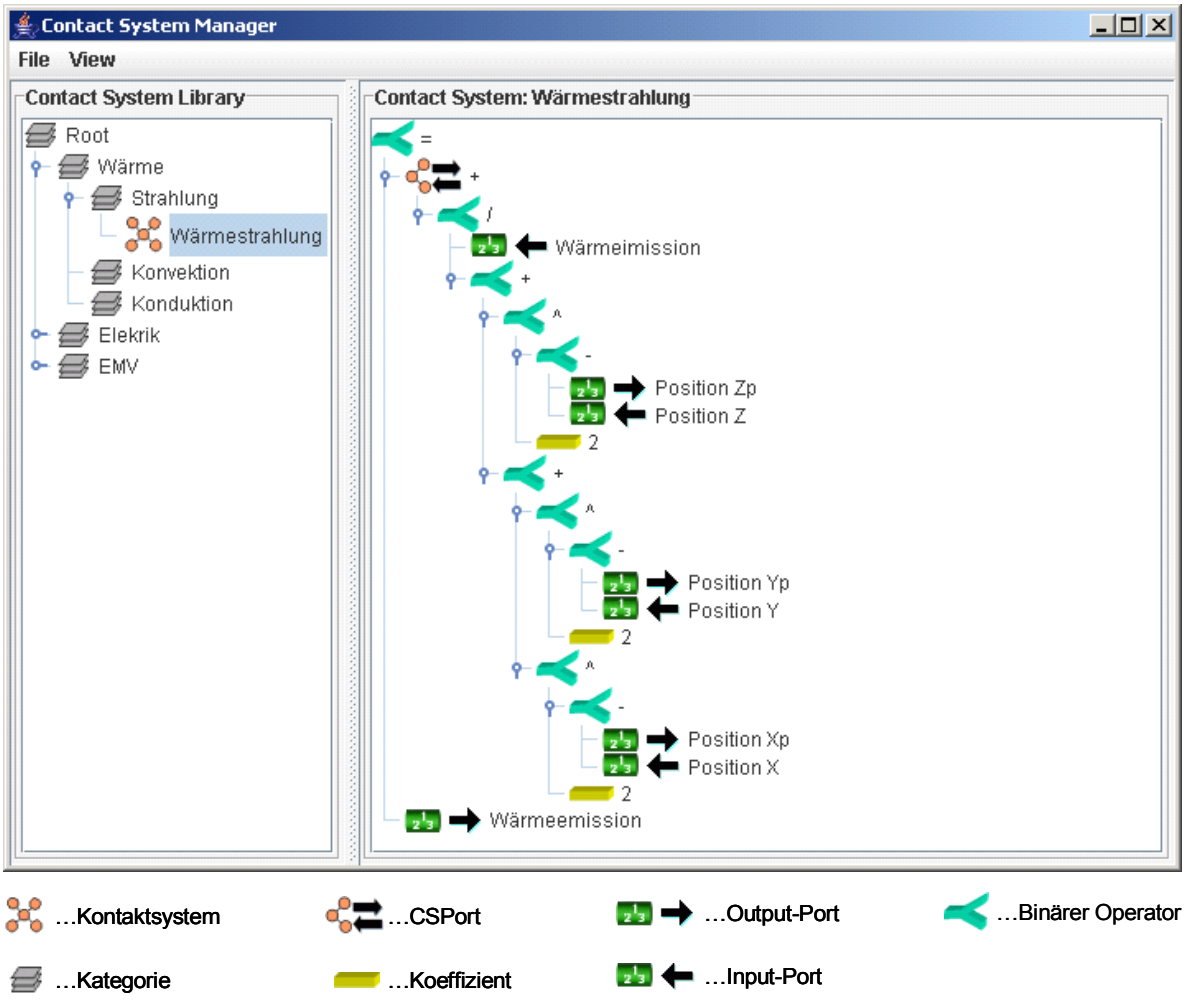


Abbildung 87:Kontaktsystem-Manager mit Kontaktsystem zur Beschreibung eines Wärmestrahlungsfeldes

Das mit den Komponenten im Wirkraum initialisierte Kontaktsystem ist in Abbildung 88 dargestellt. Für eine bessere Übersichtlichkeit ist lediglich der Knoten mit „Kotflügel Rechts“ aufgeklappt. Die anderen Knoten sind in ihrer Struktur aber identisch aufgebaut. Man kann erkennen, wie durch Hinzufügen weiterer Komponenten in den Wirkraum auch automatisch die Kontaktsystemgleichung erweitert wird.

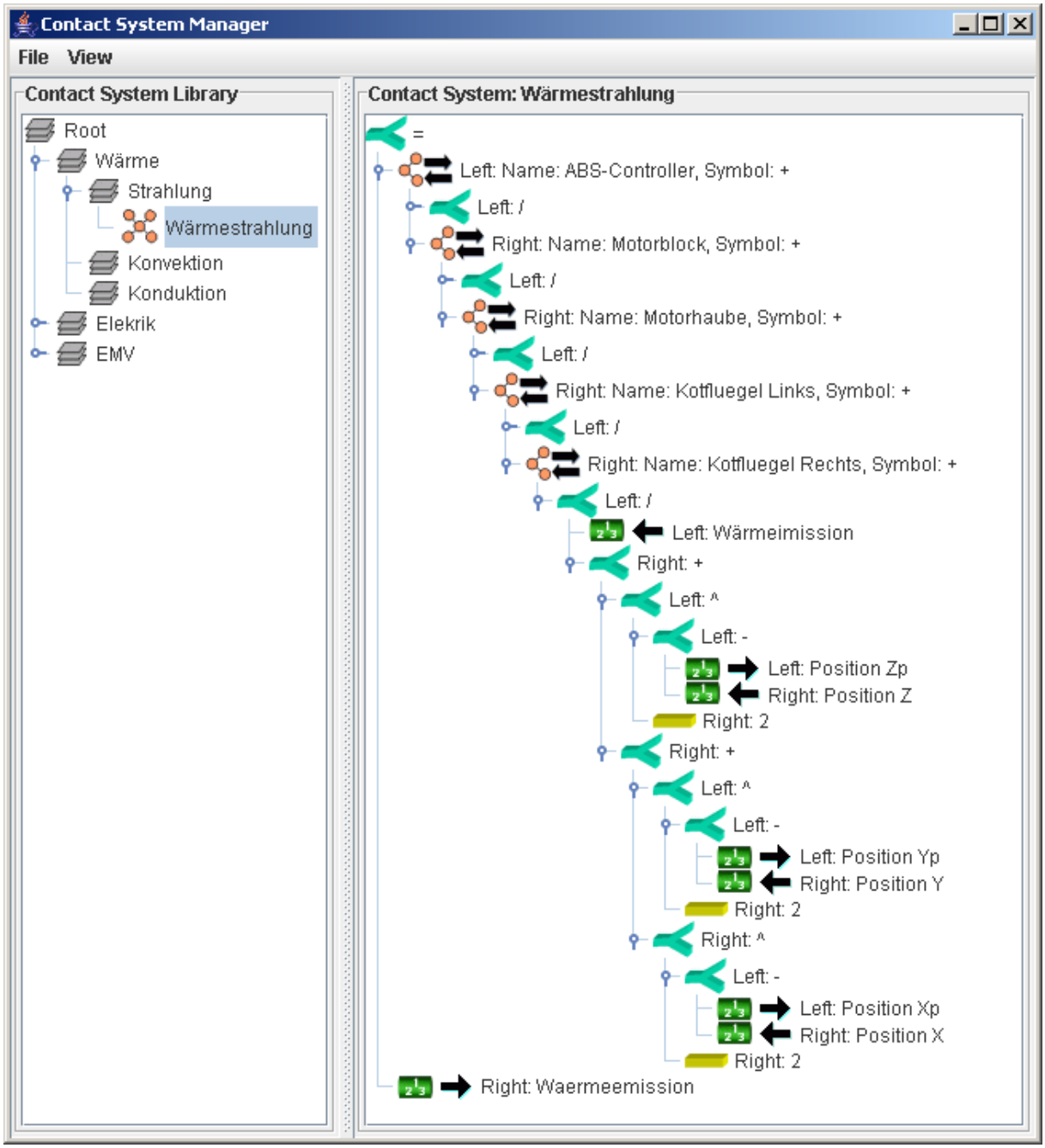


Abbildung 88: Mit den Komponenten des Wirkraums initialisiertes Kontaktsystem

Ein weiteres Kontaktsystem wird durch das Bordnetz dargestellt. Controller, Leistungsverstärker für die Ansteuerung der Ein-, bzw. Auslassventile und Fahrzeugbeleuchtung werden als

Verbraucher über ihre Leistungsaufnahme in das Kontaktsystem eingebunden[64]. Eine weitere, verbrauchskritische Komponente ist die Rückförderpumpe des ABS, die die Bremsflüssigkeit zurück in das Reservoir pumpt. Sie ist bei allen geregelten Bremssystemen notwendig, bei denen der Bremsdruck über ein Auslassventil reduziert wird [BrBi-2003]. Bei einer starken Bremsung mit ABS-Eingriff (d.h. geregelte Reduzierung des Bremsdrucks) steigt der Stromverbrauch der Pumpe sprunghaft an und kann, wenn mehrere weitere Verbraucher eingeschaltet sind, die Leistungskapazität des Bordnetzes übersteigen. In diesem Fall kann das System nicht den erforderlichen Druckabbau gewährleisten, um das Blockieren eines Rades zu verhindern. Ebenso kann die Versorgungsspannung des Controllers unter den zulässigen Wert fallen, was einen Ausfall der Regelung bedeuten würde.

Lichtmaschine und Batterie werden als Stromquellen mit ihrer elektrischen Leistung in das Kontaktsystem eingebunden. Abbildung 89 zeigt schematisch das Bordnetz mit einigen angeschlossenen Verbrauchern und Stromquellen sowie dem Laderegler, der, je nach beanspruchter Leistung durch die Verbraucher und verfügbarer Leistung des Generators[65], den Batterieladestrom regelt, bzw. bei zu geringer Leistung des Generators elektrische Energie aus der Batterie in das Bordnetz einspeist .

64 In modernen Fahrzeugen sind wesentlich mehr Verbraucher vorhanden. Moderne Fahrzeuge sind allein mit 20 bis 60 Steuergeräten ausgestattet, dazu kommen noch die Aktoren, die durch sie angesprochen werden. Für die Zukunft wird mit elektrischen Spitzenleistungen bis zu 10kW im Bordnetz gerechnet [Bosc-2003].

65 Die verfügbare Momentanleistung des Generators hängt von seiner Drehzahl ab.

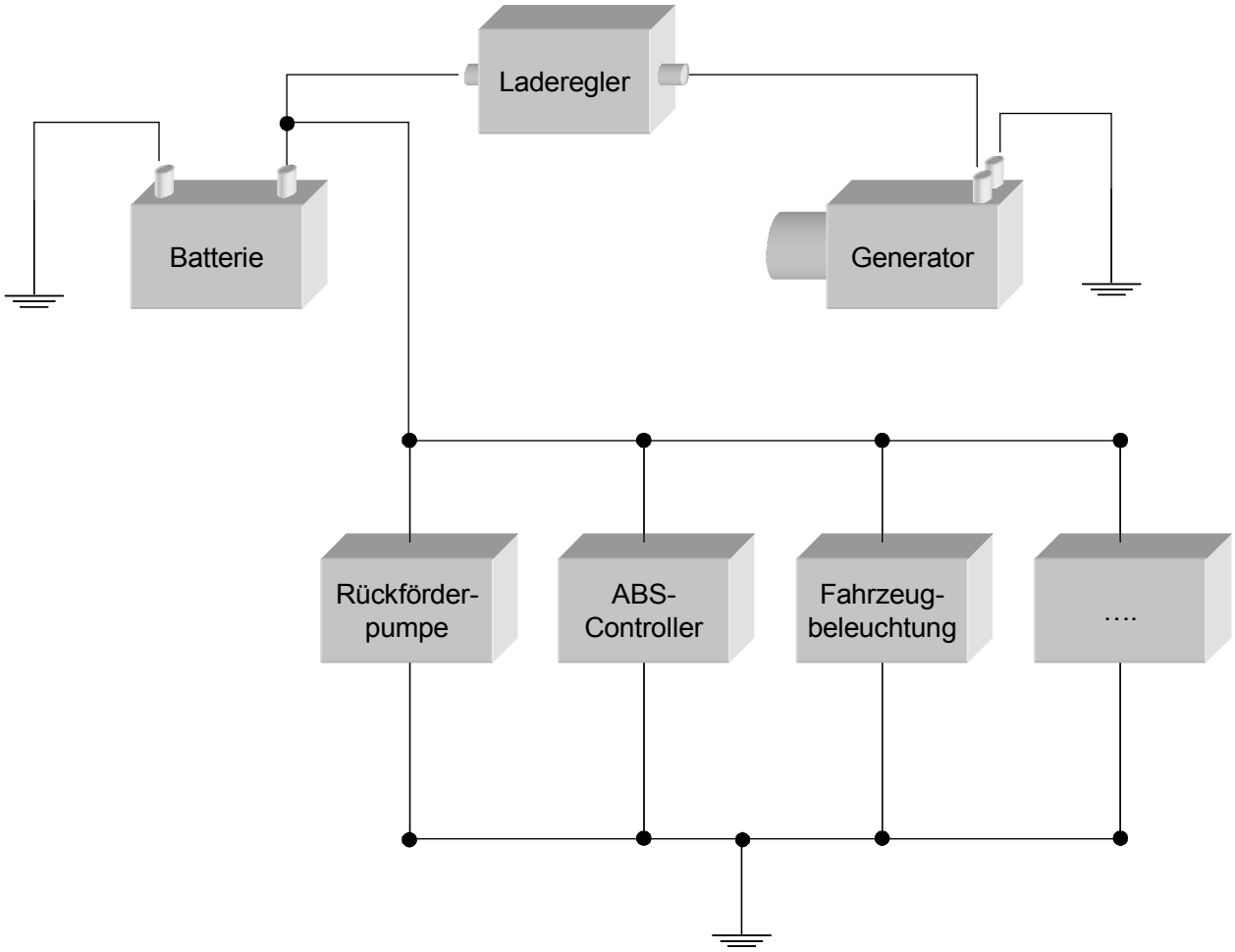
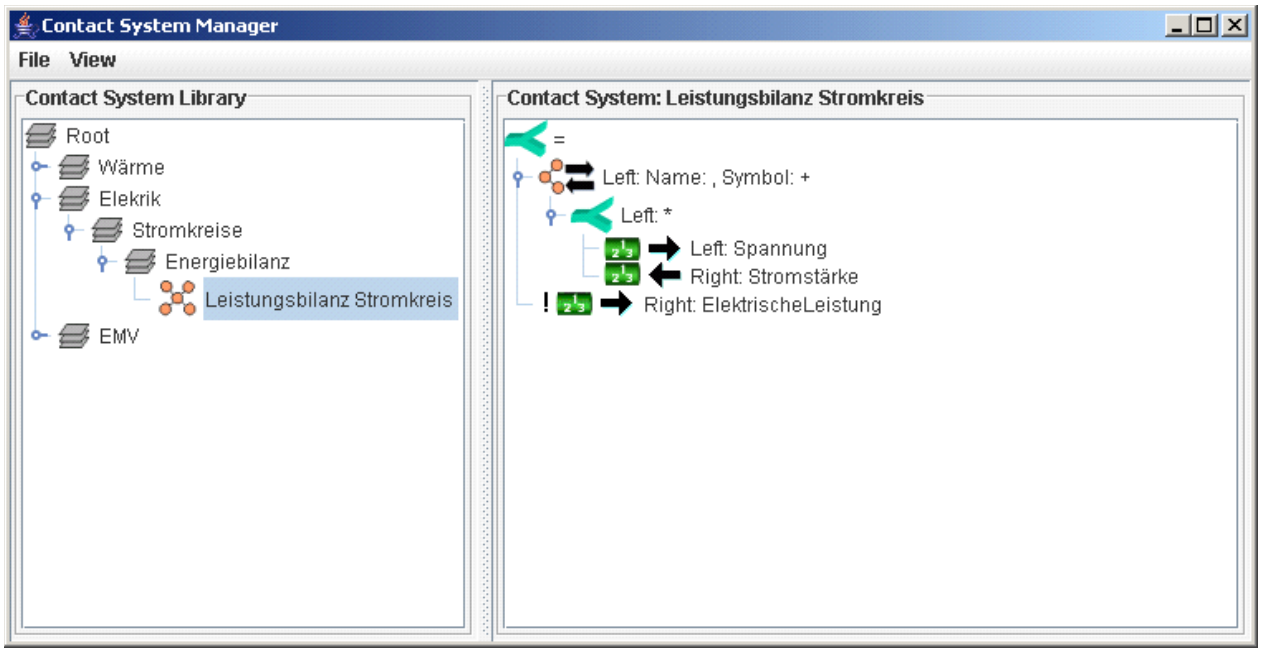


Abbildung 89: Kfz-Bordnetz mit angeschlossenen Verbrauchern und Stromquellen

Aus den oben beschriebenen Sachverhalten lässt sich ein Kontaktsystem erstellen, das in Abbildung 90 dargestellt ist. Fest vorgegeben sind hierbei die benötigten Spannungen und Stromstärken der Verbraucher. Die Leistung der Stromquelle der Stromquelle als Anforderung ist als Ausgangsport auf der rechten Seite der Gleichung angegeben.



! …Output-Port als Anforderung

Abbildung 90: Kontaktsystem für die Leistungsbilanz in Stromkreisen

Werden Verbraucher hinzugefügt, werden sie mit dem *CSPort*-Objekt verknüpft. Abbildung 91 zeigt das erweiterte Kontaktsystem der Leistungsbilanz des Bordnetzes, in das die Komponenten „ABS-Controller“, „Rueckfoerderpumpe“, „Fahrzeugbeleuchtung“ und „Leistungsverstärker“ eingefügt wurden.

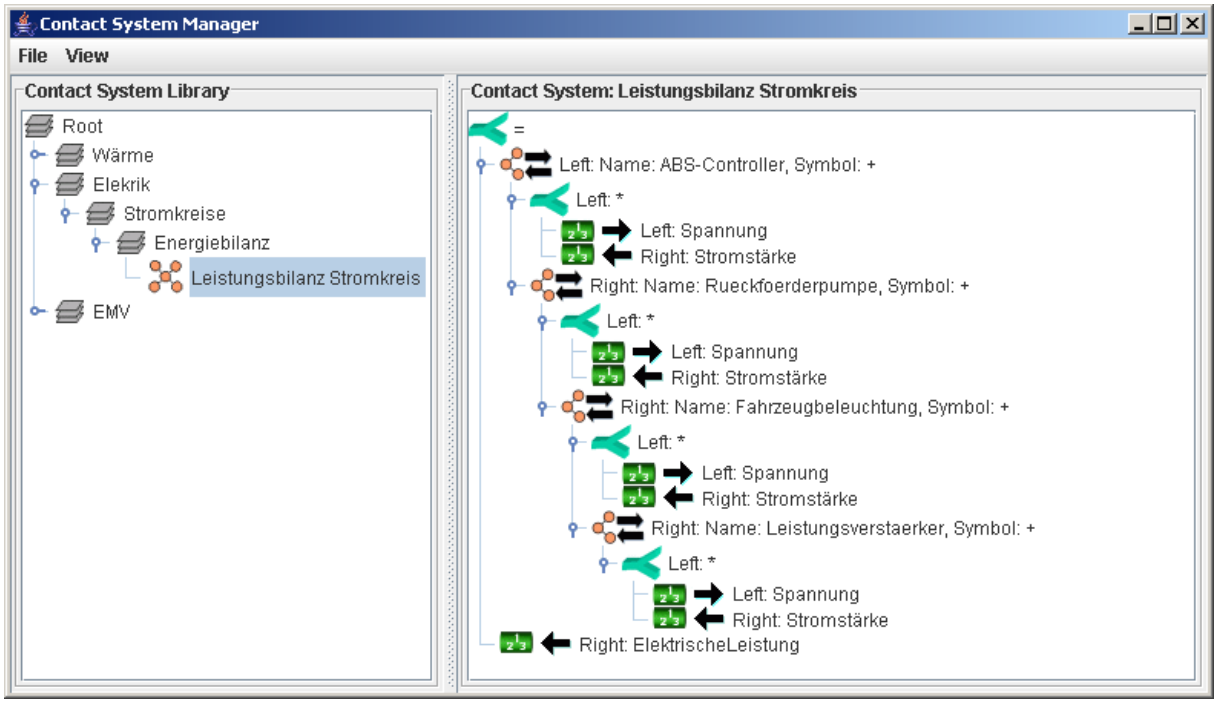


Abbildung 91: Erweitertes Kontaktsystem des Bordnetzes

Zu beachten ist hierbei, dass es sich noch nicht um eine auswertbare Gleichung handelt, da beim zuletzt eingefügten *CSPort*-Objekt die rechte Seite seines Operators für die nächste einzufügende Komponente vorgesehen und daher nicht belegt ist (der *Right*-Knoten von „Leistungsverstaerker“ fehlt im Binärbaum). Um die vervollständigte Kontaktsystemgleichung auswerten zu können, wird daher das *CSPort*-Objekt aus dem Baum entfernt und der darunter liegende Operator an seiner Stelle als rechte Seite des darüber liegenden Operators eingefügt. Abbildung 92 zeigt die aufbereitete Kontaktsystemgleichung. Das zuletzt eingefügte *CSPort*-Objekt wurde im Binärbaum durch seinen darunter liegenden binären Operator („+“) ersetzt.

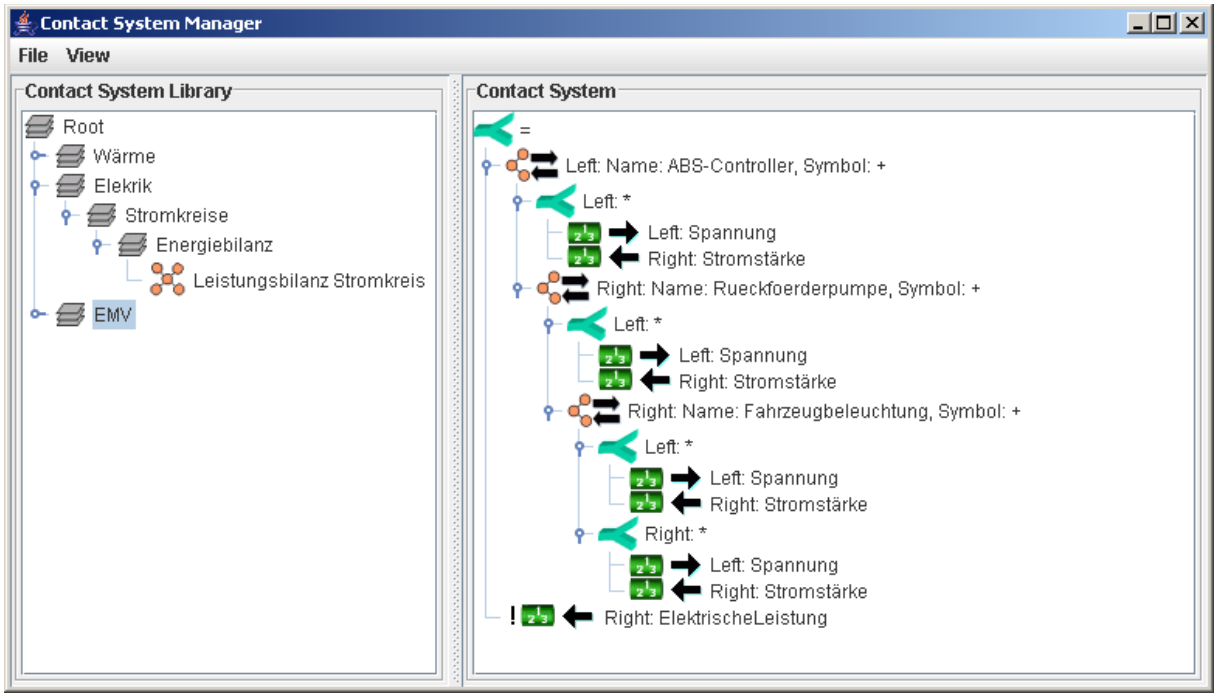


Abbildung 92: Auswertbare Kontaktsystemgleichung nach der Aufbereitung

Nun kann die Gleichung durch den Constraintsolver ausgewertet und die benötigte Leistung der Stromversorgung bestimmt werden.

5.3.3.3 Auswertung von Kontaktsystemen

Für die Auswertung von (aufbereiteten) Kontaktsystemgleichungen ist ebenfalls der in SAEP integrierte Constraintsolver zuständig. In Abbildung 93 ist die Oberfläche des Constraintsolvers mit der ausgewerteten Kontaktsystemgleichung der Leistungsbilanz im Bordnetz zu sehen. Auffallend sind in der Abbildung die Bezeichnungen der Variablen. Sie wurden mit Hilfe eines Algorithmus zur Erzeugung eindeutiger IDs vergeben, um eine eindeutige Identifikation der Parameter in der Gleichung zu gewährleisten. Dies ist eine Voraussetzung für den Gleichungslösungsalgorithmus des verwendeten Constraintsolvers. Zu erkennen ist in dem un-

tersten Textfeld, das die Ausprägung aller Parameter enthält, dass die benötigte, vorher nicht ausgeprägte Leistung (P) des Bordnetzes mit dem Wert 504 W errechnet wurde[66].

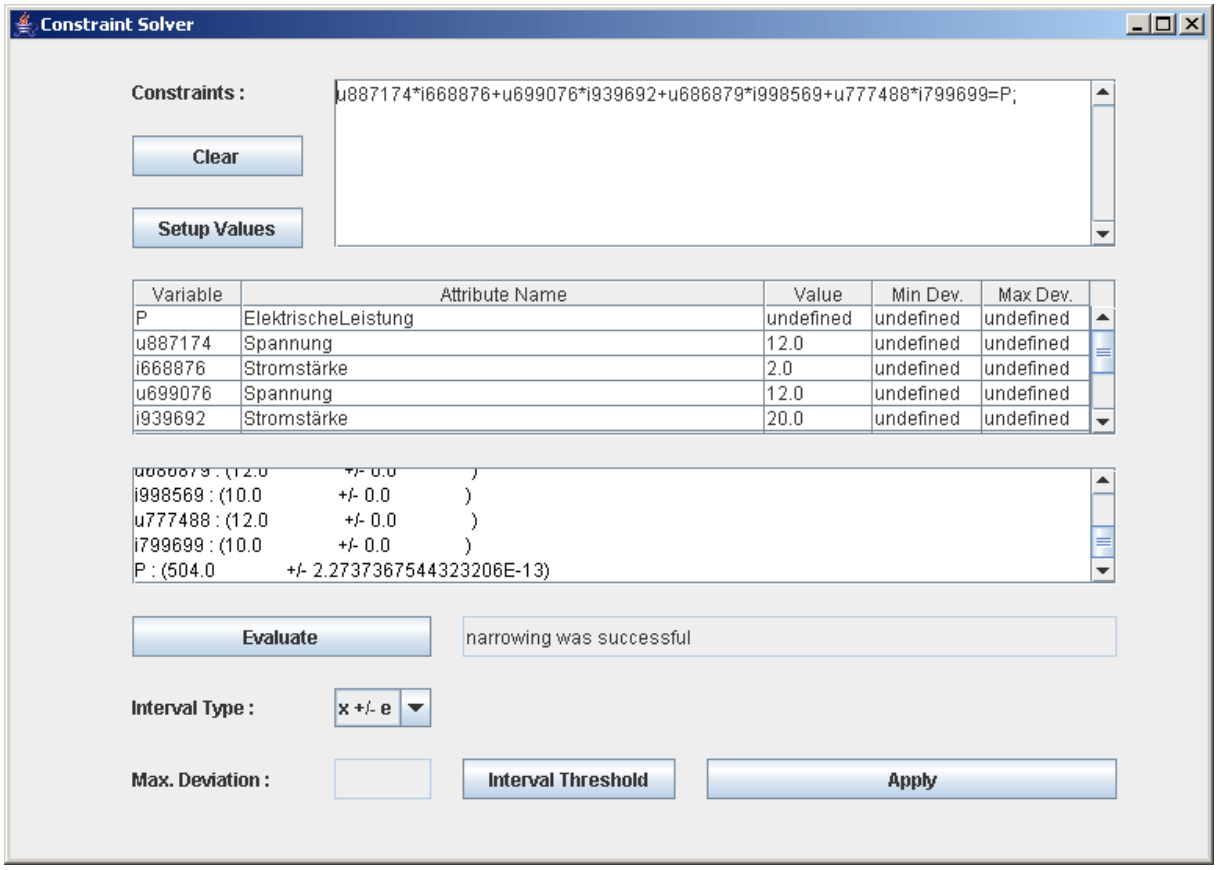


Abbildung 93: Kontaktsystemgleichung des Bordnetzes im Constraintsolver

Das interessante Ergebnis der Auswertung ist in diesem Fall aber, dass die Stromversorgung des Bordnetzes und damit des Generators (die Batterie darf für die Auslegung der Leistungsfähigkeit des Bordnetzes keine Rolle spielen) mindestens 504 W betragen muss. Dieser Wert wird direkt als Anforderung an den Generator übernommen. Bei Hinzufügen oder Entfernen von Verbrauchern in das Bordnetz wird automatisch die Anforderung an den Generator angepasst. In Rahmen des Szenarios soll angenommen werden, dass die Leistung des Generators im *Product Structure Navigator* mit 500 VA angegeben wurde. Da die Ausgangsgröße der Kontaktsystems als Anforderung definiert wurde, kann nun die Eigenschaft des Generators, (nämlich „Elektrische Leistung = 504 VA“) anhand der errechneten Anforderung überprüft werden. Abbildung 94 zeigt die Gegenüberstellung der Leistung als Eigenschaft des Generators mit der aus dem Kontaktsystem errechneten Anforderung.

66 Die geringe Abweichung von 2,2E-13 ist in der Größenordnung des errechneten Wertes vernachlässigbar.

Abbildung 94: Gegenüberstellung der Eigenschaft "Elektrische Leistung" des Generators mit der errechneten Anforderung

Wie zu erkennen ist, ist die modellierte Eigenschaft des Generators nicht anforderungskonform. Da die Anforderung bei Hinzufügen neuer elektrischer Verbraucher in den Wirkraum „Bordnetz“ sofort neu berechnet werden kann, wird damit eine sofortige Rückmeldung an die für die Auslegung der Stromversorgung zuständigen Personen möglich. Umgekehrt kann im Falle einer vorgegebenen maximalen Leistung des Generators eine die Leistung des Generators übersteigende Elektrikkonfiguration sofort ausgeschlossen werden.

5.4 Bewertung und Ausblick

Die Tragfähigkeit der im Konzept vorgestellten Methoden und rechnerunterstützten Hilfsmittel wurde anhand des Beispiels der Neuentwicklung eines PKW-Bremssystems mit Antiblockiersystem (ABS) nachgewiesen.

Zunächst wurde anhand des Beispiels gezeigt, wie die Definition von Anforderungen als Produktmerkmale und deren Zuordnung zu Produktkomponenten durch das System unterstützt wird. Im Unterschied zu bisher verfügbaren Lösungen liegen diese Informationen nicht isoliert im Anforderungsmodellierungssystem vor, sondern sind mit den Produktkomponenten samt deren Eigenschaften aus den externen Modellierungssystemen interaktiv verknüpft. Das bedeutet, dass bei der Änderung einer Produkteigenschaft im externen System sofort eine eventuelle Verletzung der zugehörigen Anforderung erkannt wird. Durch die Möglichkeit, auch komplexe mathematische Zusammenhänge zwischen Produktmerkmalen in SAEP abzubilden, ist es möglich, die Produktinformationen aus beliebig strukturierten und detaillierten Modellen für eine Evaluierung auf Basis von Anforderungen aufzubereiten. Dies ist insbesondere im Zusammenhang der Geometriemodellierung mit 3D-CAD-Systemen wichtig, da hier die produktbeschreibenden Informationen in einem sehr hohen Detaillierungsgrad vorliegen, während Anforderungen normalerweise in einem niedrigen Detaillierungsgrad gegeben sind[67].

Mit dem Konzept der Kontaktsysteme konnte gezeigt werden, dass ein großer Teil der Abhängigkeiten zwischen Produktmerkmalen anhand in einem bestimmten Bereich allgemein gültiger Zusammenhänge vom System selbst erkannt werden kann. Dabei ist der Ansatz universell für Zusammenhänge aus den unterschiedlichsten Bereichen einsetzbar (Elektrik, EMV,

67 Dies liegt in der Natur der Anforderungsdefinition. Anforderungen werden zum Großteil vor der Produktentwicklung definiert, wenn noch keine detaillierte Vorstellung über das Produkt existiert.

Thermodynamik etc.). Da die konkreten Zusammenhänge zwischen den einzelnen Produktmerkmalen nicht vorher vom Benutzer modelliert werden müssen, ist diese Vorgehensweise nicht nur bei Varianten- und Änderungs- sondern insbesondere bei Neukonstruktionen hilfreich.

Der Aufwand für die Anpassung des Systems an unternehmensspezifische Gegebenheiten (Customizing) bei seiner Einführung und für die Pflege im produktiven Einsatz hängt von verschiedenen Faktoren ab:

- **Produkt**: Je komplexer ein Produkt ist, d.h. je mehr Abhängigkeiten zwischen den Systemkomponenten und ihren Merkmalen bestehen, desto höher ist der Aufwand, beispielsweise für die Erstellung und Pflege von Schablonen. Dieser Aufwand kann durch einen „sauberen" Entwurf der Produkte hinsichtlich Modularisierung stark verringert werden, da dadurch die Abhängigkeiten zwischen den Produktkomponenten und ihren Merkmalen minimiert werden. Ein weiterer Ansatzpunkt für eine Verringerung des Aufwands ist der Einsatz von Standardkomponenten, da dadurch einmal erzeugtes und abgebildetes Wissen wieder verwendet werden kann. Der Einsatz von Standardkomponenten erlaubt es auch, Wissen in Form von Schablonen vorzudefinieren und bereits mit dem System auszuliefern.

  Die Ansätze zur Produktmodularisierung und dem Einsatz von Standardkomponenten sind aber bei der Produktentwicklung aus verschiedenen Gründen ohnehin zu verfolgen, da sich dadurch unter anderem die Qualitätssicherung stark vereinfachen lässt [VDI-1993].

- **Markt**: Die Gegebenheiten des Marktes nehmen ebenfalls Einfluss auf den Aufwand zur Einführung und Pflege des Systems. Gesetzliche Rahmenbedingungen (Sicherheitsvorschriften, Umweltschutz etc.) können beispielsweise den Umfang der zu berücksichtigenden Merkmale und damit auch den Umfang der potentiellen Abhängigkeiten zwischen ihnen erhöhen. Ein weiterer Faktor ist der angestrebte Innovationsgrad[68] bei neuen Produkten. Bei einem hohen Innovationsgrad verringert sich meistens der Anteil an wieder verwendbaren Komponenten. Zumindest aber werden Komponenten neu kombiniert, woraus sich neue, vorher unbekannte Abhängigkeiten ergeben können. Auch hier ist entscheidend, wie ein Produkt hinsichtlich Modularisierung und Verwendung von Standardkomponenten ausgelegt ist.

- **Organisation**: Ebenfalls wichtig für den Aufwand zur Einführung des Systems und seiner Pflege im produktiven Betrieb ist die Organisation des Produktentstehungsprozesses. So haben Umfang und Qualität (hinsichtlich Eindeutigkeit, Redundanzfreiheit und Struktur) der Dokumentation der Arbeitsergebnisse[69] einen entscheidenden Einfluss auf den Aufwand. Je höher der Grad an erfassten Informationen und je einfa-

---

68 Unter „Innovation" werden in diesem Zusammenhang neuartige Produktfunktionalität bzw. neuartige Realisierungsansätze für bereits bekannte Funktionalität verstanden.

69 Der Begriff „Arbeitsergebnisse" umfasst hierbei alle Informationen, die entlang der Produktlebensdauer relevant sind, wie z.B. Produktanforderungen und -Modelle, Fertigungsinformationen oder Wartungsinformationen.

> cher sie einer Verarbeitung durch den Rechner zugeführt werden können, desto geringer wird der Aufwand sein. Ein weiterer wichtiger Punkt ist EDV-Infrastruktur, die im Unternehmen vorhanden ist. Sie entscheidet darüber, wie und in welchem Umfang Informationen aus den verwendeten Systemen in SAEP genutzt werden können[70].

Eine genauere Abschätzung des Aufwands für die Einführung und Pflege des gesamten Systems im betrieblichen Einsatz kann nur individuell und firmenspezifisch erfolgen.

Die obigen Ausführungen gelten nicht oder nur begrenzt für den Ansatz der Wirkräume und Kontaktsysteme, da Kontaktsysteme auf allgemein gültigen Zusammenhängen basieren. Kontaktsystembibliotheken können daher schon vor Auslieferung des Systems definiert und mit dem System zusammen angeboten werden. Sie sind daher auch bei Neuentwicklungen ohne vorbereitende Maßnahmen einsetzbar.

Der materielle Nutzen, den die Einführung von SAEP in industrielle Produktentwicklungsprozesse bewirkt, kann nicht genau beziffert werden. Der Aufwand für die manuelle Verknüpfung von Produkteigenschaften und Anforderungen in SAEP beispielsweise scheint bei oberflächlicher Betrachtung hoch zu sein. Dieser Aufwand relativiert sich aber, wenn berücksichtigt wird, dass exakt die gleichen Verknüpfungen auch in konventionellen Produktentwicklungsprozessen hergestellt werden müssen, da sonst eine Evaluierung der Produkteigenschaften nicht durchgeführt werden kann. In herkömmlichen Entwicklungsprozessen müssen diese Verknüpfungen in Ermangelung eines Integrationssystems, im Unterschied zur Vorgehensweise in SAEP, aber bei jeder Konstruktionsänderung neu erstellt werden. Die mangelnde Sichtbarkeit von Abhängigkeiten zwischen Produktmerkmalen aus unterschiedlichen Ingenieursdisziplinen hat sich aufgrund der rasch zunehmenden Komplexität in technischen Produkten zu einem ernsthaften Problem entwickelt. Die zunehmende Zahl von Rückrufaktionen in der deutschen Automobilbranche, um nur ein Beispiel zu nennen, ist ein Beleg dafür. Die überwiegende Zahl der Fälle ist dabei nicht durch Fertigungsfehler bedingt, sondern durch Fehler in der Produktentwicklung aufgrund mangelnder Kenntnisse über die Zusammenhänge des komplexen Systems „Automobil“.

Die unterschiedlichen Teilsysteme von SAEP richten sich an verschiedene Benutzergruppen. Die Definition von Produktanforderungen ist häufig Aufgabe von dem Vertrieb zuzuordnenden Mitarbeitern. Dazu gehören Vertriebsingenieure oder Applikationsingenieure, die technische Randbedingungen für den Einsatz eines zu entwickelnden Systems beim Kunden aufnehmen, diese an die eigene Produktentwicklungsabteilung weiterleiten und die Konstruktionsergebnisse zusammen mit dem Kunden evaluieren. Die Evaluierung von Produkteigenschaften muss aber auch direkt vom Konstrukteur durchgeführt werden, da dann der Vorteil der direkten Kopplung der Systeme am effektivsten ausgenutzt werden kann. Sie ist aber andererseits auch im Aufgabenbereich der Qualitätssicherung zu sehen. Für die Pflege der Modellinhalte von SAEP, insbesondere was die Verknüpfung von Produktmerkmalen aus verschiedenen Ingenieursdisziplinen und die Erstellung von Schablonen betrifft, müssen Mit-

---

70 Wichtige Eigenschaften der verwendeten EDV-Systeme sind beispielsweise der Umfang der abgebildeten Informationen und deren Struktur (natürlichsprachlich oder formalisiert) sowie die Möglichkeiten, von außen auf diese Informationen über geeignete Schnittstellen zuzugreifen.

arbeiter mit einem entsprechend breit gefächerten Wissen an dem System geschult werden. Die Kontaktsystembibliotheken können aufgrund ihrer Allgemeingültigkeit dagegen bereits mit dem System mitgeliefert werden.

Die Verifikation hat die Tragfähigkeit der Konzepte gezeigt. Es wurde aber auch noch Forschungs- und Entwicklungsbedarf in einigen Bereichen erkannt. Die Zuordnung von extern modellierten Produktmerkmalen zu SAEP-internen Produktmerkmalen muss noch verbessert werden. Eine Möglichkeit ist beispielsweise, für ein Produktmerkmal mehrere alternative Bezeichnungen für den Namensabgleich mit den externen Merkmalen anzugeben. So könnte beispielsweise vordefiniert werden, dass das Vorhandensein einer Zeichenkette „boundingbox.len“ in einem CAD-Modellparameter immer auf ein Längenmaß verweist.

Außerdem wurde die Problematik der Lösbarkeit von Gleichungssystemen nicht betrachtet. Für komplexe, nicht lösbare Gleichungssysteme müssen Strategien entwickelt werden, die beispielsweise eine geeignete Auftrennung der Gleichungssysteme in lösbare Teilsysteme vorschlagen. Eine andere Möglichkeit ist auch die rechnerunterstützte Analyse von Gleichungssystemen.

Eine zukünftige Erweiterung der Funktionalität sollte für Kontaktsystemblätter außerdem verschiedene Betriebsmodi berücksichtigen, die vom Benutzer bzw. automatisiert vom System gewählt werden können. Ebenso sind Bedingungsoperatoren wünschenswert, die es ermöglichen, Betriebsmodi von bestimmten Bedingungen (z.B. Zuständen anderer Komponenten) abhängig zu machen. Dazu gehört auch eine Aktivierbarkeit von verschiedenen Output- und Input-Ports, um in ein und demselben Kontaktsystem mehrere Bedingungen zu simulieren. Beispielsweise kann im elektrischen Bordnetz eines Kraftfahrzeugs die benötigte Leistung der Verbraucher als fest vorgegeben und die bereitzustellende Leistung durch die Stromversorgung als Anforderung angesehen werden. Andererseits kann im Bordnetz auch die maximal zu beanspruchende Leistung eines Verbrauchers als Anforderung bei konstanter Leistung der Stromversorgung angenommen werden. Das Kontaktsystem bliebe in beiden Fällen das gleiche.

Die Komplexität der physikalischen Zusammenhänge macht eine Kopplung externer Systeme für die Evaluierung von Kontaktsystemen notwendig. Diese wurde zwar im Modell und in der Systemarchitektur von SAEP berücksichtigt, wird aber nicht durch entsprechende Oberflächen unterstützt und muss noch erprobt werden.

6 Zusammenfassung

Seit einigen Jahren machen sich zwei Trends in der Entwicklung technischer Erzeugnisse zunehmend bemerkbar, die die Qualität der entwickelten Erzeugnisse zu beeinträchtigen drohen: zum einen wird die Zeit, die für die Produktentwicklung zur Verfügung steht, aufgrund der wachsenden internationalen Konkurrenz immer geringer. Zum anderen bewirken der rasch voranschreitende technische Fortschritt und die damit einhergehenden, steigenden Kundenerwartungen eine Vervielfachung der Komplexität der Produkte. Die Komplexität ergibt sich vor allem durch die enge Verzahnung von verschiedenen Ingenieursdisziplinen in den Produkten und durch die damit hervorgerufenen Abhängigkeiten zwischen den Produktmerkmalen, die oft auch noch unterschiedlichen Domänen zugeordnet sind. Die führt zu häufig hochkomplizierten aber unzuverlässigen Erzeugnissen, da die in der Entwicklung und Erprobung unerkannt gebliebenen Abhängigkeiten, die zum Produktversagen führen können, erst in der Phase der Produktnutzung offenbar werden. Dies macht die Bedeutung der Beherrschung von Abhängigkeiten zwischen Produktmerkmalen deutlich. Ein weiteres Problemfeld ist die Evaluierung von Produkteigenschaften auf der Grundlage von Produktanforderungen.

Für die Produktentwicklung werden rechnerunterstützte Produktmodellierungswerkzeuge benutzt. Mit ihrer Hilfe wird das zu fertigende Produkt modelliert und in einem rechnerinternen Modell abgebildet. Um die Datenmodelle der einzelnen Produktmodellierungswerkzeuge über den Produktentstehungszyklus zu verwalten und Änderungen an diesen zu verfolgen, werden Produktdatenmanagementsysteme (PDM-Systeme) bzw. Product Lifecycle Management Systeme (PLM-Systeme) eingesetzt. Sie benutzen dazu Metadaten, mit deren Hilfe Produktdatenmodelle beschrieben werden. Metadaten umfassen beispielsweise das Änderungsdatum, Versionsnummern, verantwortliche Personen und Abteilungen. PDM/PLM-Systeme greifen aber nicht auf die produktbeschreibenden Informationen innerhalb der Produktdatenmodelle zu. Sie beschränken sich auf die Verwaltung der Modelle an sich. Dadurch können durch diese Systeme keine konkreten Aussagen über Auswirkungen, die sich durch Änderungen innerhalb der Modelle auf andere Bereiche des zu entwickelnden Produkts ergeben, getroffen werden.

Gemeinsam ist allen Produktmodellierungswerkzeugen, dass sie weitgehend isolierte, auf ihre spezialisierte Domäne beschränkte Sachverhalte abbilden. Dennoch wirken sich Änderungen in den von ihnen abgebildeten Produkteigenschaften auf Eigenschaften in anderen Produktbereichen aus. Daher wurden Ansätze für den Austausch von Produktdaten entwickelt, die eine Interaktion zwischen den Systemen ermöglichen sollen. Man kann unterscheiden zwischen Ansätzen, die eine Datentransformation durchführen und integrierten Ansätzen. Bei Ansätzen der Datentransformation findet der Datenaustausch gerichtet statt, d.h. Änderungen an den Daten, die durch das empfangende System durchgeführt wurden, sind für das sendende System nicht sichtbar. Bei Integrationsansätzen wird ein einziges Datenmodell für alle integrierten Systeme verwendet. Bei Änderungen des Datenbestandes durch die Systeme wird das Datenmodell ergänzt und fortgeschrieben. Änderungen werden durch diesen Ansatz dokumentiert aber nicht im Sinne einer Analyse von Abhängigkeiten evaluiert. Mögliche Aus-

wirkungen von Änderungen in einem Teilbereich des integrierten Modells auf andere Teilbereiche des Modells werden daher nicht erkannt.

Es existiert eine Reihe von Forschungsansätzen und kommerziell verfügbaren Werkzeugen, in denen Abhängigkeiten zwischen Produktmerkmalen abgebildet und ausgewertet werden können. Die zentrale Klasse von Produktmodellierungswerkzeugen im Maschinenbau sind 3D CAD-Systeme. In 3D CAD-Systemen wird die dreidimensionale Fertiggestalt des virtuellen Produkts vom Anwender definiert und in einem rechnerinternen Produktmodell abgebildet. Moderne CAD-Systeme erlauben es darüber hinaus, Abhängigkeiten, sogenannte Constraints, zwischen den Eigenschaften des virtuellen Produkts zu definieren. Eine andere wichtige Klasse von Produktmodellierungs- und Simulationssystemen sind System Engineering (SE) Werkzeuge. Sie bilden das zu entwickelnde Produkt auf einer abstrakten und systemorientierten Ebene ab. Systeme werden in SE-Werkzeugen als hierarchisch gegliederte Konstrukte aufgefasst, die ihrerseits aus Teilsystemen oder Einzelkomponenten bestehen, die untereinander über Verknüpfungen Informationen austauschen können. Die Abbildung und Auswertung von Abhängigkeiten beschränkt sich bei diesen Systemen auf ihre spezialisierte Domäne. Eine Unterstützung bei der Erkennung von implizit gegebenen Abhängigkeiten wird von keiner Lösung angeboten.

Die oben erläuterten Defizite führten zu der Entwicklung eines Systems zur rechnerunterstützten Merkmalsgewinnung und -verknüpfung aus Produktdatenmodellen zur Erkennung und Auswertung von Abhängigkeiten für die integrierte Produktentwicklung.

Es wurden Lösungen für die Verknüpfung von Produktmerkmalen entwickelt, die in externen Modellierungswerkzeugen definiert werden. Dabei stand im Vordergrund, die korrespondierenden Merkmale *effizient zu identifizieren und zu verknüpfen*[71]. Hauptanwendungsfall war hierbei die Verknüpfung von Produkteigenschaften aus externen Produktdatenmodellen mit Produktanforderungen. Die Produkteigenschaften werden direkt anhand der Anforderungen evaluiert. Für die Definition von Anforderungen wurden am RPK entwickelte Konzepte übernommen und erweitert. Zunächst können die Merkmale manuell durch den Anwender in entsprechenden Benutzungsoberflächen ausgewählt und mit den identifizierten, korrespondierenden Merkmalen verknüpft werden. Der Lösungsansatz bietet außer der direkten Verknüpfung zweier Merkmale auch noch die Möglichkeit, diese über *komplexe, mathematische Zusammenhänge* zu verknüpfen. Dies ist beispielsweise dann notwendig, wenn die einer Anforderung zugeordnete Eigenschaft nicht explizit in ihrem Ursprungsdatenmodell definiert ist, sondern sich implizit aus anderen Eigenschaften ergibt. Zum Beispiel kann es in einem Geometriemodell notwendig sein, den Durchmesser eines Kreisrings durch die Kantenlängen der ihn begrenzenden Flächen zu bestimmen.

Da der Vorgang der manuellen Verknüpfung von Produktmerkmalen angesichts großer Informationsmengen sehr aufwendig ist, wurden Konzepte für eine Unterstützung des Anwenders durch den Rechner entwickelt. Der Ansatz der *Schablonen* beschreibt ein Verfah-

---

71 Der Begriff „Merkmal“ wird hierbei als Oberbegriff für die Begriffe „Eigenschaft“ (Ist-Eigenschaft) und „Anforderung“ (Soll-Eigenschaft) benutzt

ren, wie bei einer Anpassungs- und Variantenkonstruktion ähnliche Produktkomponenten durch eine abstrakte Beschreibung der Gemeinsamkeiten beschrieben werden können. Aus diesen abstrakten Beschreibungen werden konkrete Produktkomponenten instanziiert, die die vordefinierten Merkmale bereits aufweisen und nur durch die fallspezifischen Merkmale ergänzt werden müssen. Ein weiterer Ansatz zur Identifikation von einander entsprechenden Produktmerkmalen ist die Analyse von Merkmalsbezeichnungen sowie der verwendeten Maßeinheiten zur Beschreibung der Merkmale. Bei der Analyse der Merkmalsbezeichnungen werden *Reguläre Ausdrücke* benutzt, um Muster in Zeichenketten zu definieren, denen die Merkmalsbezeichnungen für eine Auswahl entsprechen müssen. Bei der Analyse der Maßeinheiten werden die Maßeinheiten zweier Merkmale miteinander verglichen. Dabei können die Maßeinheiten entweder direkt miteinander verglichen werden oder nach einer vorherigen Normierung, die die Größenordnungen einander angleicht.

Des Weiteren wurden Lösungsansätze entwickelt, mit deren Hilfe *Abhängigkeiten zwischen Produktmerkmalen domänenübergreifend abgebildet, verfolgt und ausgewertet* werden können. Abhängigkeiten zwischen Produktmerkmalen können durch den Benutzer manuell definiert werden. Es wurden verschiedene Arten von Abhängigkeiten formuliert, mit denen die Merkmale in Beziehung gebracht werden können. *Abstrakte Beziehungen* beschreiben Zusammenhänge zwischen Produkteigenschaften, die nicht quantitativ dargestellt werden können. Dazu gehören *Konkurrierende Beziehungen*, *Ausschließende Beziehungen* und *Unterstützende Beziehungen*. Abhängigkeiten, die quantitativ dargestellt werden können, werden mit Hilfe von *Mathematischen Abhängigkeiten* modelliert. Diese werden durch Gleichungen beschrieben, deren Parameter durch die Produkteigenschaften repräsentiert werden. Die Gleichungen werden intern als Binärbäume aus Operatoren und Ausdrücken repräsentiert. Operatoren verknüpfen Ausdrücke miteinander. Die Ausdrücke können entweder aus einzelnen Parametern oder aus weiteren Ausdrücken bestehen.

Angesichts der großen Menge von Informationseinheiten, zwischen denen Abhängigkeiten bestehen können, wurde das *Konzept der Wirkräume und Kontaktsysteme* entwickelt. Mit seiner Hilfe ist es möglich, implizit gegebene Abhängigkeiten zwischen Produktmerkmalen, die sich durch allgemeingültige Zusammenhänge in einem bestimmten Kontext ergeben, automatisch zu erkennen, explizit zu formulieren und auszuwerten. Der Kontext wird durch einen Wirkraum definiert. In einem Wirkraum wird festgelegt, welche Produktkomponenten sich potentiell gegenseitig beeinflussen können und welche allgemeinen Zusammenhänge in dem Wirkraum gelten. Die allgemeinen Zusammenhänge, über die sich Komponenten in einem Wirkraum gegenseitig beeinflussen können, werden durch Kontaktsysteme beschrieben. Beispielsweise kann der Motorraum als ein Wirkraum definiert werden, in dem sich die dort vorhandenen Komponenten, wie Motorblock, Lichtmaschine oder Batterie gegenseitig über das Kontaktsystem *Wärmestrahlung* gegenseitig beeinflussen. Kontaktsysteme sind definiert über eine Rumpfgleichung, die bei Hinzufügen einer weiteren Komponente zum Wirkraum um die Merkmale der Komponente erweitert wird. Der Zustand eines Kontaktsystems wird über die entsprechenden Merkmale der im Wirkraum enthaltenen Komponenten bestimmt. Im obigen Beispiel also durch die Wärmeemission der Komponenten und ihrer Position und Orientierung im Motorraum.

Um die einem Kontaktsystem hinzuzufügenden Merkmale einer Komponente zu bestimmen, wurde der Lösungsansatz der *Binären Kontaktsystemports* sowie der *Input- und Output-Ports* entwickelt. *Binäre Kontaktsystemports* markieren die Stellen im Binärbaum der Kontaktsystemgleichung, an der die Gleichung bei Hinzufügen neuer Komponenten in den Wirkraum erweitert werden muss. Input-Ports definieren, welche Merkmale einer Komponente als Eingangsgrößen in die Kontaktsystemgleichung einfließen und damit den Zustand des Kontaktsystems beeinflussen. Output-Ports definieren, welche Merkmale für die Bestimmung der Ausgangsgrößen eines Kontaktsystems notwendig sind.

Die Verifikation des Konzepts erfolgte durch einen Softwareprototyp. Die Implementierung des Konzepts erfolgte in der Programmiersprache Java. Für die Persistierung der im System erzeugten Informationen wird eine relationale Datenbank benutzt, die über eine abstrahierte Zugriffsschicht an die Anwendungslogik angebunden ist. Die Zugriffschicht erlaubt eine flexible Erweiterbarkeit des Modells, ohne dass das zugrunde liegende Datenbankschema an neue Modellkonstrukte angepasst werden muss. Der Zugriff auf CAD-Systeme erfolgt mit Hilfe einer Implementierung der CADServices-Schnittstelle, die im Rahmen Verifikation auf ein Pro/Engineer Wildfire System zugreift.

Als Verifikationsszenario wurde die Weiterentwicklung einer bestehenden PKW-Bremsanlage und deren Erweiterung mit einem Antiblockiersystem (ABS) gewählt. Das Verifikationsszenario beginnt mit der Definition der Anforderungen und deren Zuordnung zu den Komponenten der Bremsanlage. Nach der Anforderungsdefinition wird die in einem CAD-System modellierte Baugruppe „Radbremse“ in das System übernommen und die im CAD-System modellierten Eigenschaften mit den vorher definierten Anforderungen verknüpft. Das Konzept der Wirkräume und Kontaktsysteme wird am Beispiel des elektrischen Bordnetzes des Fahrzeugs und des Wärmestrahlungsfeldes im Motorraum evaluiert. Hier konnte gezeigt werden, wie die Merkmale der Komponenten des ABS sich auf die Merkmale der vorhandenen Komponenten des Fahrzeugs auswirken und wie sich die Anforderungen an die vorhandenen Komponenten durch die neuen Komponenten verändern. Anhand des Verifikationsszenarios konnten die Vorteile des entwickelten Konzepts gezeigt werden. Diese lassen sich wie folgt zusammenfassen:

- Produkteigenschaften, die in externen Produktmodellierungswerkzeugen definiert sind, können direkt in einem integrierten Produktmodell anhand von Anforderungen evaluiert werden. Die Verknüpfung der Produktmerkmale wird durch das System unterstützt. Die Auswertung erfolgt durch das System.
- Das System unterstützt den Benutzer bei der Identifikation der korrespondierenden Merkmale und verringert damit erheblich den Aufwand für die Evaluierung von Produkteigenschaften.
- Abhängigkeiten zwischen Produktmerkmalen aus unterschiedlichen Domänen können produktmodell- und werkzeugübergreifend abgebildet, verfolgt und ausgewertet werden. Das System unterstützt den Benutzer bei der Identifikation von Abhängigkeiten zwischen Produktmerkmalen.

Dem Anwender wird durch das entwickelte und implementierte Konzept eine ganzheitliche Sicht auf das zu entwickelnde Produkt geboten. Damit ist eine Lösung für die Beherrschung der Komplexität von modernen, technischen Produkten gegeben, die den Anwender von der Definition der Produktanforderungen über die Evaluierung der Produkteigenschaften anhand von Anforderungen bis hin zur Erkennung, Darstellung und Auswertung von Abhängigkeiten unterstützt.

Literatur

[Bong-2002] Bongulielmi, L.: "Die Konfigurations- & Verträglichkeitsmatrix als Beitrag zur Darstellung konfigurationsrelevanter Aspekte im Produktentstehungsprozess". Dissertation, Zentrum für Produktentwicklung, Eidgenössische Technische Hochschule (ETH) Zürich, Zürich, 2002.

[Bosc-2003] Bosch: "Kraftfahrtechnisches Taschenbuch". Vieweg Verlagsgesellschaft, 2003.

[BrBi-2003] Breuer, B.; Bill, K.: "Bremsenhandbuch: Grundlagen, Komponenten, Systeme, Fahrdynamik". Wiesbaden, Vieweg, 2003.

[BuGZ-2002] Bungartz, H.-J.; Griebel, M.; Zenger, C.: "Einführung in die Computergrafik. Grundlagen, Geometrische Modellierung, Algorithmen". Braunschweig, Vieweg, 2002.

[Daut-2005] Dautelle, J. M.: "JScience: Java Tools and Libraries for the Advancement of Science". Webseite, 2005, http://www.jscience.org/

[Daut-2001] Dautelle, J. M.: "J.A.D.E: The Java Addition to the Default Environment". In: Dr Dobb's Journal of Software Tools 26, Ausgabe: 2, 2001, pp.52,54,56.

[DiOh-1994] Dietsche, K.-H.; Ohsmann, M.: "Mikrocontroller-Handbuch : Grundlagen, Hardware und Programmierung". Aachen, Elektor-Verlag, 1994.

[Ehrl-1995] Ehrlenspiel, K.: "Integrierte Produktentwicklung: Methoden für Prozeßorganisation, Produkterstellung und Konstruktion". München, Wien, Hanser-Verlag, 1995.

[Ehrl-2004] Ehrler, A.: "Eine integrierte Plattform für semantische Standards und Softwarekomponenten zur Unterstützung unternehmensübergreifender Produktentwicklungsprozesse". Dissertation, Forschungsberichte aus dem Institut für Rechneranwendung in Planung und Konstruktion der Universität Karlsruhe, Universität Karlsruhe, Karlsruhe, 2004.

[EiSt-2004] Eigner; Stelzer: "Produktdatenmanagement-Systeme". Heidelberg, Springer Verlag, 2004.

[EnTa-1994] Engeli, J.; Taiber, G.: "Feature-Based Modelling - Resultat eines tiefgreifenden Wandels". In: VDI-Z, Ausgabe: 5, 1994, pp.60-65.

[FFGR-2003] Fandel, G.; Francois, P.; Gubitz, K., et al.: "CAD-Marktstudie: Grundlagen, Methoden, Software, Marktanalyse". Institut für Automation, Informations- und Produktionsmanagement GmbH, 2003.

[Frie-2003] Friedl, J. E. F.: "Reguläre Ausdrücke". Köln, O'Reilly, 2003.

[Geba-2001] Gebauer: "Kooperative Produktentwicklung auf der Basis verteilter Anforderungen". Dissertation, RPK, Universität Karlsruhe, Karlsruhe, 2001.

[TFB-2003] Grabowski, H.: "TFB 16 - Rechnerintegrierte Konstruktion und Fertigung von Bauteilen - Ergebnisbericht". Kalrsruhe, Universität Karlsruhe, 2003.

[SFB-1999] Grabowski, H.: "Sonderforschungsbereich 346: Rechnerintegrierte Konstruktion und Fertigung von Bauteilen, Arbeits- und Ergebnisbericht, 1.1.1997 - 31.12.1999". Karlsruhe, SFB 346, 1999, pp.S.195 ff.

[Grab-2001] Grabowski, H.: "Rechnerunterstütztes Konstruieren und Erstellen von Fertigungsunterlagen." Karlsruhe, Institut für Rechneranwendung in Planung und Konstruktion (RPK), 2001

[GrAP-1994a] Grabowski, H.; Anderl, R.; Polly, A.: "Produktdatentechnologie: Aufbau und Entwicklungsmethodik". In: CIM Management 10, Ausgabe: 4, 1994.

[GrAP-1994b] Grabowski, H. A., R; Polly, A.: "Produktdatentechnologie: Das integrierte Produktmodell". In: CIM Management 10, Ausgabe: 5, 1994.

[GrAP-1994c] Grabowski, H. A., R; Polly, A.: "Produktdatentechnologie: Die Anwendungsprotokolle". In: CIM Management 10, Ausgabe: 6, 1994.

[Grae-1989] Grätz, J.: "Handbuch der 3D-CAD-Technik. Modellierung mit 3D-Volumensystemen". Erlangen, Siemens AG, 1989.

[HaFT-1996] Harashima, F., Fukuda T., Tomizuka, M.: "Mechatronics - 'What is it, Why, and How?' An Editorial," IEEE/ASME Transactions on Mechatronics 1, Ausgabe: 1, 1996, pp. 1-4.

[HiQE-2001] Hickey, T. J.; Ju, Q.; van Emden, M.: "Interval arithmetic: From principles to implementation". In: Journal of the ACM 48, Ausgabe: 5, 2001, pp.1038-1068.

[Hubk-1984] Hubka, V.: "Theorie Technischer Systeme". Berlin,Heidelberg,New York,Tokyo, Springer-Verlag, 1984.

[INCO-2005] INCOSE: "International Council on Systems Engineering Website". Webseite, 2005, http://www.incose.org

[Klae-1993] Klaeger, R.: "Modellierung von Produktanforderungen". Dissertation, Institut für Rechneranwendung in Planung und Konstruktion (RPK), Karlsruhe (TH), Karlsruhe, 1993.

[Koll-1985] Koller, R.: "Konstruktionslehre für den Maschinenbau". Springer-Verlag, 1985.

[KrTA-2002] Krause, F. L., Tang, T., Ahle,U.: "Abschlussbericht zum Projekt iViP - integrierte Virtuelle Produktentstehung". München, Carl Hanser Verlag, 2002.

[Kunz-2002] Kunze, H.: "Ein Beitrag zur Gewinnung von Funktionen für mechanische Produkte aus 3D-CAD-Gestaltmodellen". Dissertation, RPK, Universität Karlsruhe, Karlsruhe, 2002.

[MaEH-2004] Mahl, A.: "Konzeption und industrieller Einsatz einer Plattform für flexible, unter-nehmensübergreifende Kooperationen". In: Produktdaten Journal 2, Ausgabe, 2004, pp.46-50.

[Meus-1999] Meuser, A.: "Elektrische Sicherheit und Elektromagnetische Verträglichkeit". Berlin, VDE-Verlag, 1999.

[NN-2004a] N.N: "Mechatronik: Definition und Begriffsbestimmung". Webseite, 2004, http://www.mechatronik-portal.de/mechatronik_definition.html#mechatronik

[SML-2005] N.N: "Systems Modeling Language (SysML) Specification". PDF-Dokument, 2005, http://www.sysml.org/artifacts/spec/

[UML-2003a] N.N: "UML 2.0 Infrastructure Specification". PDF-Dokument, 2003, http://www.omg.org/docs/ptc/03-09-15.pdf

[UML-2003b] N.N: "UML 2.0 Superstructure Specification". PDF-Dokument, 2003, http://www.omg.org/docs/ptc/03-08-02.pdf

[UML-2003c] N.N: "UML 2.0 OCL Specification". PDF-Dokument, 2003, http://www.omg.org/docs/ptc/03-10-14.pdf

[UML-2003d] N.N: "UML 2.0 Diagram Interchange Specification". PDF-Dokument, 2003, http://www.omg.org/docs/ptc/03-09-01.pdf

[DRAG-2004] N.N: "Dragon - Development of an inteRActive EnGineering Portal for Open Networks". Webseite, 2004, http://www.dragon.uni-karlsruhe.de/

[NIST-2005] N.N: "National Institute of Standards and Technology". Webseite, 2005, http://www.nist.gov

[BFGo-2005] N.N: "BF Goodrich Homepage". Webseite, 2005, http://www.bf-goodrichtires.com

[Engi-2005a] N.N: "Engineous Software". sdfs, Webseite, 2005, http://www.engineous.-com

[OAI-2005] N.N: "Ohio Aerospace Institute". Webseite, 2005, http://www.oai.org

[Park-2005] N.N: "The Parker Hannifin Corporation". Webseite, 2005, http://www.parker.com/

[GE-2005] N.N: "General Electric". Webseite, 2005, http://www.ge.com

[OU-2005] N.N: "Ohio University". Webseite, 2005, http://www.ohio.edu/

[Rose-2005] N.N: "R.O.S.E Informatik Homepage". Heidenheim a.d. Brenz, ROSE Informatik GmbH, Webseite, 2005, http://www.rose.de/

[OMG-2005a] N.N: "Object Management Group". OMG, Webseite, 2005, http://www.omg.org

[NASA-2005a] N.N: "National Aeronautics and Space Administration Homepage". Webseite, 2005, http://www.nasa.gov

[NASA-2005b] N.N: "The NASA STEP Testbed". Webseite, 2005, http://step.nasa.gov/testbed

[UGS-2005] N.N: "UGS". UGS, Webseite, 2005, www.ugs.com

[UGS-2005a] N.N: "Teamcenter SLATE Brochure". UGS, PDF-Dokument, 2005, http://www.ugs.com/products/teamcenter/docs/br_teamcenter_slate.pdf

[UGS-2005b] N.N: "Teamcenter Requirements Brochure". UGS, PDF-Dokument, 2005, http://www.ugs.com/products/teamcenter/docs/fs_tc_requirements.pdf

[Sun-2005] N.N: "Java Technology". Sun Microsystems Inc., Webseite, 2005, java.sun.com

[MySQ-2005] N.N: "MySQL". MySQL AB, Webseite, 2005, www,mysql.com

[Mant-2005] N.N: "ManTIS Homepage". Webseite, 2005, http://www.omg.org/homepages/mfg/

[OMG-2005b] N.N: "OMG's CORBA Website". OMG, Webseite, 2005, http://www.corba.org/

[OMG-2005c] N.N: "Computer Aided Design Services Specification". OMG, Webseite, 2005, http://www.omg.org/docs/formal/05-01-07.pdf

[Ince-2005] N.N: "Incentrix GmbH". Webseite, 2005, http://www.incentrix.de

[Engi-2005b] N.N: "Providing organizations with a design and integration infrastructure that automates processes and enables collaborative product design". Engineous Software, Webseite, 2005, http://www.engineous.com/product_FIPER.htm

[VDI-1972] N.N: "Formgebung technischer Erzeugnisse; Empfehlung für den Konstrukteur". Berlin, Beuth Verlag, 1972

[VDI-2004] N.N: "VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme". Berlin, Beuth Verlag, 2004

[PaBe-1997] Pahl, G. B., W.: "Konstruktionslehre : Methoden und Anwendung". Heidelberg, Springer Verlag, 1997.

[ReYK-2002] Reese, G. Y., R.J.; King T.: "MySQL". San Francisco, O'Reilly, 2002.

[Rode-1991] Rodenacker, W. G.: "Methodisches Konstruieren - Grundlage, Methodik, praktische Beispiele". Berlin, Springer Verlag, 1991.

[Rode-2000] Rodewald, A.: "Elektromagnetische Verträglichkeit : Grundlagen - Praxis". Wiesbaden, Vieweg, 2000.

[AIAA-2000] Röhl, P. K., R.;Irani, R.;Sobolewski, M.; Kao, K.;Bailey,M.: "A Federated Intelligent Product Environment". In Conference Proceedings: 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, USA, American Institute of Aeronautics and Astronautics (AIAA), 2000.

[Roth-1994] Roth, K.: "Konstruieren mit Konstruktionskatalogen. Band 1: Konstruktionslehre". Berlin, Springer Verlag, 1994.

[WiWo-2005] RTR; WIW: "Mercedes-Benz: Größte Rückrufaktion aller Zeiten". Verlagsgruppe Handelsblatt GmbH, Webseite, 2005, http://www.wiwo.de

[Rude-1998] Rude, S.: "Wissensbasiertes Konstruieren". Habilitation, RPK, Universität Karlsruhe, Karlsruhe, 1998.

[Rzeh-1998] Rzehorz, C.: "Wissensbasierte Anforderungsmodellierung auf der Basis eines integrierten Produktmodells". Dissertation, Forschungsberichte des Instituts für Rechneranwendung in Planung und Konstruktion RPK, Universität Karlsruhe (TH), Karlsruhe, 1998.

[Samp-2003] Sampson, M. E.: "Connected Requirements". EDS, White Paper, 2003

[Schi-2002] Schildt, H.: "Java 2: The Complete Reference". Emeryville, McGraw-Hill Osbourne Media, 2002.

[Seil-1985] Seiler, W.: "Technische Modellierungs- und Kommunikationsverfahren für das Konzipieren und Gestalten auf der Basis der Modellintegration". Dissertation, RPK, Universität Karlsruhe, Karlsruhe, 1985.

[ShMä-1995] Shah, J.; Mäntylä, M.: "Parametric and feature-based CAD/CAM : concepts, techniques and applications". New York, Wiley, 1995.

[SpKr-1997] Spur, G.; Krause, F. L.: "Das virtuelle Produkt. Management der CAD-Technik". München, Carl Hanser Verlag, 1997.

[Szyk-2001] Szykman, S.: "The NIST Design Repository Project". Greenbelt MD, Präsentation, 2001, http://library.gsfc.nasa.gov/Announce/Sept2002/Simon-Szykman_Presentation.ppt.

[Szyk-2000] Szykman, S. S., R.D.;Bochenek C.;Racz, J.W.;Senfaute, J.: "Design Repositories: Next-Generation Engineering Design Databases". In: IEEE Intelligent Systems 15, Ausgabe: 3, 2000, pp.4855.

[SzSR-2001] Szykman, S. S., R.D.;Regli W.C.: "The role of knowledge in next-generation product development systems." In: ASME Journal of Computation and Information Science in Engineering 1, Ausgabe, 2001, pp.3-11.

[VDI-1993] VDI, H.: "VDI-Richtlinie 2221 - Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte". In Ingenieure V-Vd (Hrsg.), "VDI-Handbuch Konstruktion", Berlin, Beuth Verlag, 1993.

[Wate-2004] Waterbury, S. C.: "The NASA STEP Testbed The NASA STEP Testbed Pan Galactic Engineering Pan Galactic Engineering Framework (PGEF) Framework (PGEF) Status and Roadmap". NASA, Präsentation, 2004, http://ned.gsfc.nasa.gov/slides/PGEF_SEWG_20040525.pdf

[Wate-2003] Waterbury, S. C.: "The NASA STEP Testbed Pan Galactic Engineering Framework (PGEF)". NIST, Präsentation, 2003, http://syseng.nist.gov/aerospace-workshop/slides/Tuesday/waterbury.ppt

[Wate-2005b] Waterbury, S. C.: "Project Home Site for the Development of The Pan Galactic Engineering Framework (PGEF)". NASA, Webseite, 2005, http://pangalactic.us/

[WeDe-2003] Weber, C. D. T.: "New Theory-based Concepts for PDM and PLM". In Conference Proceedings: ICED 03 International Conference on Engineering Design, Stockholm, 2003.

Abkürzungsverzeichnis

AES	Anforderungsentwicklungssystem
AP	Application Protocol
API	Application Programming Interface
BMBF	Bundesministerium für Bildung und Forschung
B-Rep	Boundary Representation
CAD	Computer Aided Design
CCO	Common Core Object
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
CSG	Constructive Solid Geometry
DMU	Digital Mock-Up
DRAGON	Development of an Interactive Engineering Portal for Open Networks
DRP	Design Repository Project
ERP	Enterprise Resource Planning
FEM	Finite Element Methode
FIPER	Federated Intelligent Product Environment
FMEA	Failure Mode and Effects Analysis
FOD	Function Oriented Design
GUI	Graphical User Interface
INCOSE	International Council on Systems Engineering
ISO	International Organisation for Standardisation
iViP	Innovative Technologien und Systeme für die integrierte, virtuelle Produktentstehung
LKT	Lehrstuhl für Konstruktionstechnik/CAD
ManTIS	Manufacturing Technology and Industrial Systems Task Force
MRO	Maintenance, Repair and Overhaul
NASA	National Aeronautics and Space Administration
NIST	National Institute for Standards
OMG	Object Management Group
PDD	Property Driven Development/Design
PDM	Produktdatenmanagement
PGEF	Pangalactic Engineering Framework
PLM	Product Lifecycle Management
PPM	Integriertes Produkt- und Produktionsmodell
RPK	Institut für Rechneranwendung in Planung und Konstruktion
RTM	Requirement Traceability Management
SAEP	System zur anforderungsbasierten Evaluierung von Produkteigenschaften
SCM	Supply Chain Management
SDK	Software Development Kit
SE	Systems Engineering
SE DISG	Systems Engineering Domain Special Interest Group

SFB	Sonderforschungsbereich
STEP	Standard for the Exchange of Product Model Data
SysML	System Modeling Language
TFB	Transferbereich
TÜV	Technischer Überwachungsverein
UCL	University College of London
UDF	User Defined Features
UIP	Unternehmensübergreifende Integrationsplattform
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
XML	eXtensible Markup Language