Eine Methode für eine automatisierte Informationsextraktion aus großen CAD-Datenbeständen zur Bauteilsuche und Klassifikation

Bergische Universität Wuppertal
Fakultät 7 – Maschinenbau und Sicherheitstechnik
Fachgebiet Maschinenbauinformatik

Dissertation zur Erlangung der Würde eines
DOKTORS DER
INGENIEURWISSENSCHAFTEN
Dr.-Ing.

von
M.Sc. Robin Roj

Wuppertal 2016

Vorwort

Besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr.-Ing. Woyand, für die angenehme Zusammenarbeit und aufschlussreiche Betreuung über mehrere Jahre hinweg. Die zahlreichen Diskussionen und langen Gespräche haben dafür gesorgt, dass der rote Faden nie verloren gegangen ist und die Arbeit zu jedem Zeitpunkt gut vorangehen konnte.

Auch meinem Büropartner, Herrn Simionescu, ist für diverse Brainstorming-Sitzungen zu danken, die zur Entwicklung der Algorithmen beigetragen haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AFR – Automated Feature Recognition
ANS – American National Standards
API – Application Programming Interface
B-REP – Boundary Representation
CAD – Computer Aided Design
CAE – Computer Aided Engineering
CAM – Computer Aided Manufacturing
CATIA – Computer Aided Three-Dimensional Interactive Application
CBD – Case Based Design
CFD – Computational Fluid Dynamics
CNC – Computerized Numerical Control
COM – Component Object Model
CSG – Constructive Solid Geometry
DA – Design Automation
DEE – Design and Engineering Engine
DFA – Design for Assembly
DEKLARE – Design Knowledge Acquisition and Redesign Environment
DIN – Deutsches Institut für Normung
DKAP – Design Knowledge Acquisition Process
DM – Data Mining
DNS – Desoxyribonukleinsäure
DOM – Document Object Model
DPM – Daten Prozess Management
DPN – Disassembly Petri Net
DTD – Document Type Definition
DWG – Drawing
DXF – Drawing Interchange Format
EDV – Elektronische Datenverarbeitung
ERP – Enterprise Ressource Planning
ETIM – Elektrotechnisches Informationsmodell
FDG – Feature Dependency Graph
FEA – Finite Element Analysis
FEM – Finite Elemente Methode
FMEA – Failure Mode and Effects Analysis
FU3 – Fertigungsunterlage materialabhängig (3D)
FUM – Fertigungsunterlage materialabhängig

GUI – Graphical User Interface
HG – Hierarchical Graph
I-DEAS – Integrated Design and Engineering Analysis Software
IGES – Initial Graphics Exchange Specification
IPE – Integrierte Produktentwicklung
IPO – IGES/PDES Organization
IPS – Intelligent Personal Assistant
ISBN – Internationale Standardbuchnummer
ISO – International Organization for Standardization
IT – Informationstechnik
ITV – Invariant Topology Vector
JT – Jupiter Tessellation
KBE – Knowledge Based Engineering
KBS – Knowledge Based Systems
KD – Knowledge Discovery
KDD – Knowledge Discovery in Databases
KD – Knowledge Engineering
KM – Knowledge Management
KOMPRESSA – Knowledge-Oriented Methodology for the Planning and Rapid Engineering of Small-Scale Applications
MBD – Model Based Definition
MOKA – Methodology for Knowledge Based Engineering Applications
MSG – Model Signature Graph
MTAD – Multiple Tool Axis Direction
NBS – National Bureau of Standards
NIST – National Institute of Standards and Technology
PAG – Property Adjacency Graph
PDES – Product Data Exchange using STEP
PDM – Product Data Management
PGH – Pairwise Geometric Histogram
PLM – Product Lifecycle Management
PML – Product Markup Language
RE – Requirements Engineering
SAP – Systemanalyse und Programmentwicklung
SAX – Simple API for XML
SQL – Structured Query Language
STAD – Single Tool Axis Direction
STEP – Standard for the Exchange of Product Model Data

STL – Standard Triangulation Language
SUV – Sports Utility Vehicle
SVG – Scalable Vector Graphics
TIF – Tagged Image File Format
TRIZ – Teoria Reschenija Isobretatjelskich Sadatsch (Theorie des erfinderischen Problemlösens)
UNSPSC – United Nations Standard Products and Services Code
US PRO – United States Product Data Association
VBA – Visual Basic for Applications
VKI – Verteilte Künstliche Intelligenz
VPE – Virtuelle Produktentwicklung
VRML – Virtual Reality Modeling Language
W3C – World Wide Web Consortium
WRL – World
XML – Extensible Markup Language
XPS – Expertensystem
XSL-FO – Extensible Stylesheet Language - Formatting Objects
XSLT – Extensible Stylesheet Language Transformation

Abstract

Die Entwicklung von neuen Produkten läuft in der modernen Industrie größtenteils unter Zuhilfenahme von diversen Softwaresystemen ab, die neben CAD-Dateien auch andere Formate in einer Vielzahl erzeugen. Eine Herausforderung, die dadurch entsteht, ist es die Datenmengen in geeigneter Art und Weise zu verwalten, um so die Vollständigkeit sicherzustellen und einfache Zugangsmöglichkeiten zu gewährleisten.

In der vorliegenden Arbeit wird für die Lösung dieses Problems ein Konzept vorgestellt, das ausgehend von einer ungeordneten Menge an CAE-Modelldaten die Möglichkeiten einer automatisierten Sortierung der Dateien beschreibt. Ein besonderer Schwerpunkt wird dabei auf die CAD-Modelle gelegt, die in der Regel in proprietärer Form vorliegen, aber im Zuge der Analyse in diverse herstellerunabhängige Austauschformate überführt werden. Die Grundidee besteht in einer mithilfe von Algorithmen automatisierten Untersuchung, die aus allen verfügbaren CAD-Dateien Informationen sowie Daten über die Eigenschaften der Bauteile sammelt. Die gesamte Vorgehensweise kann in die folgenden drei Bereiche eingeteilt werden.

Am Anfang wird erforscht, welche Informationen, die sich in der Datenbasis befinden, von Relevanz sind und wie diese zugänglich gemacht werden können. Insbesondere werden dafür verschiedene CAD-Dateiformate untersucht, um festzustellen, welche Art von Daten sich automatisch extrahieren lassen.

Im zweiten Schritt werden die gewonnenen Informationen aufbereitet und für den Bauteilvergleich in Form von Signaturen arrangiert.

Die abschließende Informationsweiterverarbeitung ist in die Bereiche Ähnlichkeitssuche und Klassifikation aufgegliedert und schließt den Entwurf sowohl mit einer Bauteilsuchmaschine als auch mit einer Einteilung der Dateien in verschiedene Cluster ab.

Somit entsteht eine Methode, die anhand von Informationsauswertung eine automatische Überführung eines ungeordneten in einen strukturierten Datenpool erlaubt und mit den entwickelten Programmen die Zugänglichkeit zu den Bauteilen erhöht sowie die Handhabung vereinfacht bzw. beschleunigt.

Abstract

For the development of new products in modern industry applications different software-systems are utilized, which create CAD-data as well as other file-formats in large amounts. A big arising challenge is the applicable administration of these data masses in order to ensure completeness and guarantee a simple accessibility.

For the solution of this problem the present work provides a concept that characterizes the possibilities of an automatic sorting of an unordered CAE-database filled with engineering parts. The main focus is on CAD-models, which are usually proprietary, but in the course of an encompassing analysis easily transferable to independent exchange formats. The basic idea is rested upon an algorithmic automatized investigation that gathers information about the properties from all available CAD-data. The whole approach can be arranged in the following three sections.

At the beginning it is explored which of the information stored in the database is relevant and how it can be made accessible. For that reason several CAD-file formats are examined in order to determine which kind of data can be extracted automatically.

In the second step the obtained information is processed and prepared for the upcoming comparison between engineering parts in form of signatures and fingerprints, respectively.

The concluding information finishing is split up into the areas of similarity search and classification to complete the draft in a search engine for components as well as a division into various clusters.

Thus, a method is developed that automatically conveys a disorganized database into a well structured one aided by the evaluation of all collected information. Thereby the developed algorithms allow a faster access to the engineering parts and simplify their handling.

Teil 1: Stand der Forschung und Technik

Kapitel 1. Einleitung

1.1 Ausgangssituation

Der Einzug von computergestützten Konstruktionsmethoden in den ingenieurstechnischen Alltag veränderte, vereinfachte und beschleunigte die Entwicklung von neuen Produkten maßgeblich. Die allgemeinen Vorteile der EDV, wie z.B. eine erleichterte Vervielfältigung, Wiederverwendung und Erweiter- sowie Verwaltbarkeit von Datenbeständen, wirkten sich besonders im Bereich der technischen Entwicklungen positiv auf die Bearbeitungsdauer von gesamten Projekten aus. Das Ziel der meisten technischen Softwareprogramme, die für die Ingenieurswissenschaften entwickelt wurden, war es und wird es sein, die Tätigkeiten, die von den Entwicklern im Zuge eines Projektes ausgeübt werden, am Computer zu realisieren, zu vereinfachen und möglichst zu automatisieren.

Die ersten CAD-Programme (Computer Aided Design), die in der Praxis angewendet wurden, ermöglichten den Benutzern technische Zeichnungen zu erstellen und bereits erarbeitete Sachverhalte wiederzuverwenden, zu kopieren oder zu erweitern [Weis-08]. Die Arbeitsschritte, die früher manuell durchgeführt wurden, konnten somit auf den Computer übertragen werden, was von großem Vorteil war. Im Laufe der Zeit wurde zusätzlich die Software erweitert und verbessert, so dass nicht nur die Erstellung von technischen Zeichnungen ermöglicht wurde, sondern auch die Konstruktion von virtuellen dreidimensionalen Modellen. Somit konnte in den meisten Fällen der mühsame Herstellungsprozess von Prototypen eingespart werden. Außerdem brachte die 3D-Technik Vorteile für die Anschauung des fertigen Bauteiles und den Zusammenbau zu ganzen Baugruppen, was ebenfalls die Kosten massiv senkte. Computergestützte Methoden übertrugen dementsprechend nicht nur die konventionellen Arbeitsschritte in die virtuelle Welt, sondern erschufen auch alternative Prozesse, die mit weniger Aufwand zum Ziel führten.

Im Laufe der Zeit wurden die Programme zur Unterstützung der Ingenieure so komplex, dass nicht mehr nur die technischen Aspekte in der Entwicklung berücksichtigt wurden, sondern die gesamte Umsetzung von Großprojekten - zusammen mit allen Aspekten der Planung, Konstruktion, Produktion und der Vermarktung - computergestützt ablaufen konnte. Dieses allumfassende Konzept der Produktplanung und -entwicklung wird als CAE (Computer Aided Engineering) bezeichnet. Meywerk definiert den Begriff als eine Sammlung von Aufgabenfeldern [Meyw-07]:

> *Berechnung, Simulation u.a.; CAE umfasst in seiner eigentlichen Bedeutung rechnergestützte Ingenieurtätigkeit, also alle mit Rechnern durchzuführenden Ingenieurtätigkeiten;*

Im Zuge dessen soll ebenfalls das PLM (Product Lifecycle Management) erwähnt werden, das bereits bei der Entwicklung eines neuen Produktes nicht nur dessen Planung, Herstellung und Gebrauch berücksichtigt, sondern auch andere Gesichtspunkte wie Marketing, Wartung oder Wiederverwendung der Materialien, Recycling und Umweltverträglichkeit. Entwicklerunterstützende PLM-Systeme werden von Schäppi et al. wie folgt definiert [SAKR-05]:

> *PLM-Systeme sind technische Informationssysteme zur Speicherung, Verwaltung und Bereitstellung von produktbeschreibenden Daten und Dokumenten in allen Phasen des Produktlebenszyklus.*

Da es mehr und mehr wichtig wird Daten, Kennwerte und Wissen im Allgemeinen zum richtigen Zeitpunkt in möglichst einfacher Art und Weise zur Verfügung zu stellen, entstehen gerade in großen Unternehmen neue Herausforderungen bei der Verwaltung, Bereitstellung und auch der Sicherheit von Daten. Die Internationalisierung von großen Unternehmen mit weltweiten Standorten ist nur ein Beispiel der fortschreitenden Globalisierung. Neue Wege der Kommunikation und des Austauschs von Wissen müssen erforscht und praktikabel angewendet werden. Insbesondere die Möglichkeiten des Internets führen zu Innovationen auf vielen Gebieten, die sich zurzeit, z.B. in einem völlig neuen Ausmaß des Sammelns von Daten, bemerkbar machen. Auf der einen Seite stellt dies große Vorteile und neue Anwendungsfelder bereit, was andererseits aber auch zu Problemen hinsichtlich der Verwaltung, Auffindbarkeit oder Distribution führen kann. Folglich müssen geeignete Werkzeuge zur Durchsuchung und Klassifikation (Taxonomie) von Datenbeständen erforscht und zur Verfügung gestellt werden, die es den Anwendern ermöglichen große Bestände zielgerichtet und in kürzester Zeit zu untersuchen und das gesuchte Wissen bzw. die gewünschten Informationen zweckmäßig herauszufiltern. Neben Suchmaschinen, die textbasiert mit Hilfe paralleler Algorithmen große Datenmengen durchsuchen, sind auch umfangreichere Ansätze denkbar, die auch in der Lage sind komplexere Datenbasisentitäten, wie etwa Bauteile, miteinander zu vergleichen und Ähnlichkeiten festzustellen.

An dieser Stelle soll als Beispiel der Begriff 'Industrie 4.0' erwähnt werden, dessen Konzept in Zukunft viele der beschriebenen Anforderungen erfüllen soll, indem nicht nur die Forschung und Entwicklung von neuen Produkten mit der Konstruktion und Fertigung verknüpft wird, sondern auch der gesamte Produktlebenszyklus einschließlich dem Ein- und Verkauf, der Logistik, der Umweltverträglichkeit oder dem Marketing. Neben einer effizienten Bereitstellung von Informationen ist für die Realisierung einer solchen Verflechtung der einzelnen Disziplinen in erster Linie ein Kommunikati-

onssystem von Nöten, das allen Beteiligten einen effektiven Wissensaustausch ermöglicht.

Schon heute existieren Ansätze der computergestützten Informationsadministration beispielsweise auf dem Gebiet der digitalen Lagerverwaltung. Wenn die Mindestmenge eines bestimmten Artikels im Lager unterschritten wird, erscheint im Computersystem entweder eine Meldung oder es bestellt automatisch eine festgelegte Ersatzmenge, ohne dass eine weitere menschliche Interaktion nötig ist.

Nichtsdestotrotz befindet sich die Vernetzung und Bereitstellung von Wissen noch in den Anfängen. In Kapitel 2.6 wird beispielsweise in der Industrie schon heute angewendete Software dargestellt, die neben den gängigen Aufgaben eines PLM-Systems auch Methoden der Klassifikation und des Bauteilvergleichs implementiert.

1.2 Hintergrund und Motivation

Wie in Kapitel 1.1 beschrieben, existieren schon heute marktreife professionelle Programme, die sich den existenten Ansätzen der Datenverwaltung und der Informationsextraktion bedienen und auf die Anforderungen der Entwickler beim Entwurf von neuen Produkten zugeschnitten sind. Dennoch bedarf es auf dem Gebiet der Datenhandhabung und der Subdisziplinen, wie z.B. Datenbasisanalyse, Informationsauffindung und -extraktion, Suchmaschinenentwicklung oder Klassifikation von Informationen, noch Grundlagenforschung.

Eine Herausforderung, die der Gebrauch von umfangreichen CAE-Programmen mit sich bringt, ist die Verwaltung der Daten, die mit Hilfe unterschiedlicher Software erschaffen werden. Zum einen sollte eine einfache Zugreifbarkeit und Erreichbarkeit von alten Beständen ermöglicht werden, allerdings stellt es auch eine große Herausforderung dar, alle jemals erschaffenen Dateien oder Arbeiten so zu sortieren, zu kategorisieren und zu klassifizieren, dass sie jederzeit übersichtlich und leicht zugänglich sind. Problematisch ist insbesondere die Anwendung in großen Unternehmen, bei denen sehr große Datenmengen (evtl. an unterschiedlichen Standorten) anfallen. Gewiss gibt es viele verschiedene Systeme und Ansätze, die Dateien nach verschiedenen Gesichtspunkten, wie z.B. Datum, Projekt, Abteilung oder Kunde, verfügbar halten. Ein alternatives Konzept ist auch ein unsortierter Datenpool, deren Inhalt erst bei einer konkreten Anfrage eines Benutzers gezielt angezeigt wird. In Kapitel 2.3 wird näher auf die Möglichkeiten des Datenbasisaufbaus eingegangen.

Eine erwähnenswerte Technik, bei der große Datenmengen untersucht werden, sind die folgenden zwei bzw. drei Fachbegriffe. Unter KD (Knowledge Discovery) bzw. KDD (Knowledge Discovery in Databases) wird der Prozess verstanden, der relevantes Wissen aus einer Menge von Daten, deren Großteil uninteressant ist, extrahiert. Für die Vorbereitung und Entwicklung sowie für die spätere Anwendung von wissensbasierten Systemen ist dieser Schritt unerlässlich. Ester und Sander definieren den Begriff nach Fayyad, Piatetsky-Shapiro und Smyth [EsSa-00]:

> *Knowledge Discovery in Databases (KDD) [...] ist der Prozess der (semi-) automatischen Extraktion von Wissen aus Datenbanken, das*
> - *gültig (im statistischen Sinne)*
> - *bisher unbekannt und*
> - *potentiell nützlich (für eine gegebene Anwendung) ist.*

KDD wird nicht als Methode angesehen, sondern als ein Gesamtprozess, der das Wissen nicht nur entdeckt und extrahiert, sondern auch andere Informationen, wie z.B. die Art der Abspeicherung, analysiert [FaPS-96]:

> *KDD focuses on the overall process of knowledge discovery from data, including how the data are stored and accessed, how algorithms can be scaled to massive datasets and still run effectively, how results can be interpreted and visualized, and how the overall man-machine interaction can usefully be modeled and supported.*

Teil dieses Gesamtprozesses ist das sogenannte DM (Data Mining), das sich nur auf die Entdeckung von interessantem Wissen aus großen Datenmengen bezieht. Petersohns Definition beinhaltet auch die Etymologie [Pete-05]:

> *In der Umgangssprache steht Data Mining für Datenabbau, Datengewinnung. Analog dem Bergbau, dessen Gegenstand bspw. die Gewinnung von Kohle aus den Erdmassen darstellt, soll unter Data Mining die Gewinnung von interessanten Daten aus den Datenmassen verstanden werden.*

Die internen Techniken, die bei der Umsetzung in Form von Algorithmen während des DM ablaufen, können sehr unterschiedlich sein. Bei der Untersuchung großer Datenbestände, müssen Prioritäten gesetzt werden, damit der Suchprozess nach relevanten Informationen angemessen schnell ablaufen kann.

Die Multidisziplinarität der Programme bringt neben zahlreichen Vorteilen allerdings auch Komplikationen und neue Herausforderungen mit sich. Zum einen besteht das Problem, dass jedwede Software innerhalb von kurzer Zeit weiterentwickelt und aktualisiert wird, so dass die Benutzer bei jeder Erneuerung die Anwendung wiedererlernen müssen. Aufgrund der Komplexität müssen ganze Lehrveranstaltungen und Seminare angeboten werden, um alle Möglichkeiten und Funktionen der CAE-Programme erfassen zu können. Der Wechsel zwischen verschiedenen Versionen eines Programmes oder gar zwischen Software unterschiedlicher Hersteller bringt außerdem die Problematik der Datenaustauschbarkeit mit sich. Während neue Versionen eines Programmes mit älteren Ausgaben meist abwärtskompatibel sind, besteht beim Austausch von Daten verschiedener Hersteller meist das Problem, dass auf die Dateien gar nicht zugegriffen werden kann. Aus diesem Grund wurden programmübergreifende und herstellerunabhängige Austauschformate, wie z.B. STEP (Standard for the Exchange of Product Model Data) oder STL (Standard Triangulation Language), entwickelt, die einen vorgegebenen Aufbau besitzen und dreidimensionale Bauteile in immer gleicher Weise abspeichern. Somit gehen allerdings auch die Konstruktionsschritte, die der Anwender bei der Entwicklung durchgeführt hat, verloren.

In den gängigen CAD-Programmen werden normalerweise alle Funktionen, Einstellungen und Abläufe, die bei der Konstruktion eines virtuellen Bauteiles benutzt werden, im sogenannten Strukturbaum abgespeichert. Kornprobst beschreibt den Strukturbaum so [Korn-07]:

> *Der Strukturbaum, der im Modellbereich angezeigt wird, erweitert sich im Laufe einer Konstruktion fortwährend. Dabei werden alle Konstruktionsschritte, die explizite Geometrie oder Regeln hervorbringen, in chronologischer Abfolge eingeschrieben.*

Dieser führt zu einer sehr guten Übersicht und Nachvollziehbarkeit der Gedankengänge und bringt den großen Vorteil, dass Änderungen, die im Nachhinein hinzugefügt werden sollen, einfach durchführbar sind. Es muss nicht das gesamte Bauteil neu entwickelt werden, wenn ein Detail hinzukommt, geändert oder entfernt werden soll.

Besonders problematisch wird es, wenn die Dateien verschiedener Programme in einer großen Datenbasis miteinander in Einklang gebracht oder sogar von Hand erstellte technische Zeichnungen oder Abbildungen im Nachhinein digitalisiert werden sollen.

Der Umgang mit Datenbeständen bringt außerdem die Automatisierung der typischen Ingenieursaufgaben mit sich. Verständlicherweise müssen große Datenbasen nach bestimmten vordefinierten Gesichtspunkten automatisch durchsucht werden, da eine manuelle Handhabung zu zeitintensiv wäre. Im Zuge dessen geht es also darum möglichst viele Informationen aus allen Dateien automatisiert zu extrahieren, auszuwerten und die gewonnenen Erkenntnisse weiterzuverarbeiten, um die gewünschten Kategorien von Daten zu erschaffen, herauszufiltern oder erneut zugänglich zu machen. Ähnliche Ansätze sind Bestandteil vieler aktueller Forschungen, die die Übertragung ingenieurswissenschaftlicher Aufgaben auf den Computer zum Thema haben.

Neben einer generellen Anwendung der DM- bzw. KDD-Konzepte in der Softwareentwicklung sind neben der eigentlichen Lauffähigkeit noch weitere Eigenschaften, wie z.B. eine benutzerfreundliche Bedienbarkeit, eine übersichtliche grafische Oberfläche (GUI - Graphical User Interface) und insbesondere eine praktikable Ausführgeschwindigkeit, von großer Bedeutung. Um beispielsweise die Leistungsfähigkeit der Algorithmen zu erhöhen, ist eine parallele Verarbeitung auf mehreren Kernen von Vorteil. Zudem sollten einfache Softwaretools heutzutage so entwickelt werden, dass die Anwendung möglichst selbst erklärend ist und somit kein Handbuch mehr gebraucht wird. Ein weiterer Ansatz zur Unterstützung der Benutzer ist die Idee der Hintergrundprogramme (XPS (Expertensysteme), Softwareagenten), die, z.B. bei der Anwendung von relativ komplizierter CAE-Software, die Vorgehensweise des Nutzers überwachen und ggf. Fehler korrigieren, Wissen liefern oder zusätzliche Werkzeuge bieten.

Zusammenfassend ist anzumerken, dass sich einerseits schon professionelle Programme im industriellen Einsatz befinden, die den Zweck verfolgen im Großen und Ganzen das Datenmanagement zu vereinfachen und zu verbessern. Andererseits existieren im Bereich der DM- und KDD-Forschung noch viele Ansätze, deren Implemen-

tierung in Hinsicht auf die industrielle Anwendung sinnvoll wäre. Des Weiteren erlauben weitere technische Fortschritte und immer neue Technologien die Entwicklung von zusätzlichen Konzepten.

1.3 Wissenschaftliche Zielsetzung

Das Ziel dieser Arbeit ist es ein neuartiges Konzept zu entwerfen, das die folgenden drei Schwerpunkte enthält. Ausgehend von einer gegebenen CAD-Datenbasis, wie sie in großen Unternehmen angelegt wird, sollen relevante Informationen zuerst extrahiert, danach aufbereitet und zum Schluss weiterverarbeitet werden.

In der industriellen Praxis existieren zahlreiche Ansätze und Praktiken, die entwickelten Dateien aufzubauen und ganze Datenbasen zu handhaben. Aufgrund der Diversität der verschiedenen Ansätze und der praktikablen Systeme entstehen unzählige Möglichkeiten der Anwendung und vor allem der Vereinfachung in der Entwicklung und Kreation von neuen Projekten. Folglich gilt es zu Beginn die Theorie herauszukristallisieren und mit der angewandten Praxis zu vergleichen. Das Hauptaugenmerk soll dabei vor allem auf dem Aufbau von Dateien und ganzen Datenbasen liegen, aber auch auf den einzelnen Schritten, die bei der Entwicklung und Konstruktion eines neuen Produktes befolgt werden.

Des Weiteren werden die Funktionsweisen sowie die Vor- und Nachteile bekannter Datenstrukturen (Dateitypen) ermittelt, so dass Methodiken entwickelt werden können, die die praktischen Eigenschaften zum Vorteil nutzen und im nächsten Schritt in Form von Computeralgorithmen realisiert werden. Für diese vorbereitenden Schritte sind die Klärungen der Fragen, wie bestimmte Daten oder ganze Datenbasen aufgebaut sind, welche beinhalteten Informationen für die Weiterverarbeitung von Interesse und wie diese herauszufiltern sind, notwendig. Abschließend wird aus den gewonnenen Erkenntnissen eine Software entwickelt, die nicht nur die beschriebenen Schritte realisiert, sondern auch die nützlichen Informationen in praktischer Art und Weise weiterverarbeitet.

Im Großen und Ganzen müssen also die Systematiken der Bauteilkonstruktion sowie der Forschung und Entwicklung erörtert und der daraus resultierende Datenaufbau ermittelt werden. Somit können die Fragen geklärt werden, welche Informationen von Interesse in gegebenen Datenmengen enthalten sind und wie diese extrahiert werden können. Erst danach kann die Auswertung dieser Daten und die Weiterverwendung bzw. Verarbeitung in Angriff genommen werden. Einen wichtigen Bestandteil wird die automatisierte Kategorisierung der vorhandenen Informationen darstellen. Nachdem mit Hilfe des DM die gewünschten Datenbestandteile aufgespürt und extrahiert wurden, können diese beispielsweise für die Entwicklung einer Suchmaschine benutzt werden. Es gilt weiterhin zu klären, nach welchen Gesichtspunkten die Dateien in einem Datenpool sortiert werden können, so dass eine sinnvolle Zugänglichkeit für den Anwender entsteht. Außerdem sollen die Möglichkeiten erforscht werden die eigentliche Datenbasis unberührt zu lassen und nur die vom Benutzer angef-

ragten Dateien in eine zusätzliche Umgebung zu kopieren, wo auf sie extern zugegriffen werden kann. Die Suchkriterien sollen dabei so umfassend und vielfältig wie möglich definiert sein, so dass die gefilterten Informationen, z.B. aus einer einzigen Datei, auf die personalisierte Suche zutreffen.

Der erste Schritt konzentriert sich hauptsächlich auf die Übertragung einer proprietären Datenbasis hin zu herstellerunabhängigen Formaten, die einzeln untersucht werden und deren unterschiedlicher Informationsgehalt erforscht wird. Bestimmte Daten sind am Anfang in das jeweilige Programm einzuspeisen, um diese zu analysieren. Sie können als Input bezeichnet werden und um zu verstehen wie sie von Nutzen sein können, muss als erstes ihr Aufbau und ihre Struktur untersucht werden. Dabei sind insbesondere die Möglichkeiten zu erörtern, die die einzelnen Datenformate (vgl. Kapitel 2.2) im Hinblick auf die nächsten Schritte bieten. Neben Austauschformaten werden aber auch am Beispiel von CATIA V5 (Computer Aided Three-Dimensional Interactive Application) exemplarisch die Kapazitäten von proprietären CAD-Dateien untersucht. Auch zweidimensionale Formate sowie technische Zeichnungen und Fotos werden im Zuge des Reverse Engineering analysiert, so dass möglichst viele Informationen gesammelt und im zweiten Schritt aufbereitet werden können. Auch die dort enthaltenen Daten sollen möglichst automatisch extrahiert werden, um sie, z.B. für die Rekonstruktion von Bauteilen, im nächsten Schritt weiterzuverwenden. Dies wird ein weiteres Themengebiet darstellen. Es behandelt die Frage, inwiefern es möglich ist, aus gegebenen, unter Umständen nicht vollständigen Bauteilinformationen ein dreidimensionales CAD-Modell bzw. eine möglichst präzise Annäherung zu rekonstruier-en. Als Inputinformationen können demnach alle verfügbaren Arten von Fakten über ein Bauteil oder eine Baugruppe, wie z.B. Fotos, technische Zeichnungen oder Prototypen, dienen, die wiederum automatisiert ausgewertet und weiterverarbeitet werden. Schließlich sind die übrig gebliebenen Informationen, die nun größtenteils für die Anwendung relevant sein sollten, auf ihre Konsistenz zu analysieren und auszuwerten.

Für die anschließende Weiterverarbeitung bedarf es einer Aufbereitung aller relevanten gesammelten Informationen. Neben einer Darstellung im einfachen Textformat wird im Speziellen auf die Bauteildarstellung in Form von Signaturen bzw. Fingerprints eingegangen. Darunter werden gebündelte Informationen verstanden, die die wichtigsten Eigenschaften der Teile beinhalten. Eine grundlegende Fragestellung ist demzufolge die Möglichkeit einer eindeutigen Identifikation von Bauteilen durch eine reduzierte Informationsessenz. Die Grundidee ähnelt der eindeutigen Identifikation eines Menschen aufgrund seines Fingerabdruckes oder seiner DNS. Angewandt auf die eindeutige Darstellung bei ingenieurstechnischen Bauteilen in einem Datenpool werden hier im Lauf der Informationsaufbereitung die folgenden zwei Ansätze erforscht.

Zum einen wird auf die Charakterisierung eines Bauteiles durch eine Kategorisierung der wichtigsten Eigenschaften eingegangen. Ähnlich wie bei einem Produktschlüssel entsteht dabei eine Zahlenfolge, die etwas undifferenziert die Beschaffenheit oder die Merkmale der Teile umschreibt und später zur Identifikation und zum Vergleich beiträgt.

Der zweite Ansatz behandelt eine graphenbasierte Darstellung der CAD-Geometrien, bei der die einzelnen Flächen und Flächenverbindungen in einer Graphen- bzw. Matrixform aufbereitet und abgebildet werden. Auch dieser Ansatz ist Teil der Informationsaufbereitung zur finalen Weiterverarbeitung im letzten Schritt.

Dieser zeichnet sich aus durch die Auswertung der gegebenen Informationen, um am Ende zu einer geeigneten Klassifizierung bzw. einem Vergleich zu führen. Dieser letzte Schritt der Informationsweiterverarbeitung konzentriert sich zum einen auf den Bauteil- bzw. Fingerprintvergleich, der in einer Art Suchmaschine zur Auffindung von Ähnlichkeiten führt. Des Weiteren werden die Möglichkeiten der automatischen Klassifikation bzw. der Einteilung in Cluster, die sich ebenfalls der Bauteilsignaturen als Grundlage bedienen, ergründet.

1.4 Dissertationsübersicht

Diese Dissertation lässt sich in zwei große Bereiche einteilen. Der erste Teil beinhaltet die Kapitel 1-3, in denen die klassischen Begriffe der Forschung und Entwicklung sowie der Anwendung und der industriellen Praxis erörtert und dargestellt werden. Der zweite Teil umfasst in den Kapiteln 4-7 die spezifischen Forschungsthemen und die Eigenentwicklungen, die im Zuge der hier dargestellten Untersuchungen erstellt wurden. Im Gegensatz zu den konventionellen Vorgehensweisen, die im ersten Teil beschrieben werden, zeigt der zweite Teil neue Aspekte und Strategien auf, die die im ersten Teil beschriebenen Begriffe der Datenanalyse und Klassifikation von Produkten erweitern bzw. erneuern und verbessern.

Kapitel 1 gibt einen Überblick und definiert die wissenschaftliche Zielsetzung sowie die Ausgangssituation und die Motivation dieser Arbeit.

In Kapitel 2 geht es um die grundlegenden Begriffe der ingenieurstechnischen Praxis in der Industrie. Zur Einführung werden zuerst die wichtigsten industriellen CAE-Systeme sowie die gängigen herstellerunabhängigen Dateiformate charakterisiert. Neben einer Erläuterung der Grundbegriffe zu den Themen des Datenbasisaufbaus und der Produktklassifikation wird außerdem ein Schwerpunkt auf drei in der Anwendung befindliche Programme gelegt, die neben den konventionellen Funktionen von PLM-Systemen auch eingehendere Kapazitäten zum Bauteilvergleich und zur Dateianalyse bieten. Am Ende von Kapitel 2 werden des Weiteren noch drei Beispiele zur Datenorganisation in Großunternehmen veranschaulicht.

Um einen allumfassenden thematischen Überblick über die Fragestellungen dieser Arbeit zu erhalten, konzentriert sich Kapitel 3 auf den aktuellen Stand der Forschung sowie auf die theoretische Basis. Abgesehen von den Begrifflichkeiten, die schon in Kapitel 2 erwähnt werden, sind hier auch die Forschungsansätze, z.B. zur Graphentheorie oder zu allgemeinen Klassifikationsaufgaben, enthalten. Der Unterschied besteht vornehmlich aus marktreifer softwaretechnischer Anwendung und theoretischer Grundlagenforschung.

Der zweite Teil der Arbeit beinhaltet die Eigenentwicklungen und ist in drei Bereiche eingeteilt. In Kapitel 4 wird auf die Informationsextraktion aus unterschiedlichen Dateiformaten eingegangen. Die gesammelten Daten werden unter dem Stichwort 'Informationsaufbereitung' in Kapitel 5 zu Fingerprints und Matrizen geformt, so dass in Kapitel 6 die finale Informationsweiterverarbeitung stattfinden kann. Dies geschieht in Kapitel 6.1 unter den Gesichtspunkten einer Suchmaschine, mit deren Hilfe ähnliche Bauteile identifiziert werden können, und in Kapitel 6.2 mit verschiedenen Klassifizierungsalgorithmen, die die Aufgabe erfüllen, die Bauteile in Gruppen einzuteilen bzw. ihnen Cluster zuzuweisen.

Die Kapitel 4-6 beschreiben im Allgemeinen die Konzepte, die entwickelt wurden, und sind fast ausschließlich in jeweils drei Abschnitte eingeteilt. Zu Anfang wird die Grundidee bzw. der Gedanke einer Methode erläutert. Danach folgt die Implementierung bzw. Umsetzung in der Programmiersprache 'Python' und ggf. dem CAE-Programm 'CATIA V5'. Am Ende werden in der Evaluation die Ergebnisse analysiert und auf ihre Konsistenz geprüft. Zur Erprobung der entwickelten Software wurde eine Gesamtheit von ca. 100 Bauteilen zugrunde gelegt, die in Abbildung 1.1 auszugsweise (nicht maßstabsgetreu) dargestellt werden. Die Teile, die in allen anderen Abbildungen enthalten sind, gehören ebenfalls zu dieser Testmenge.

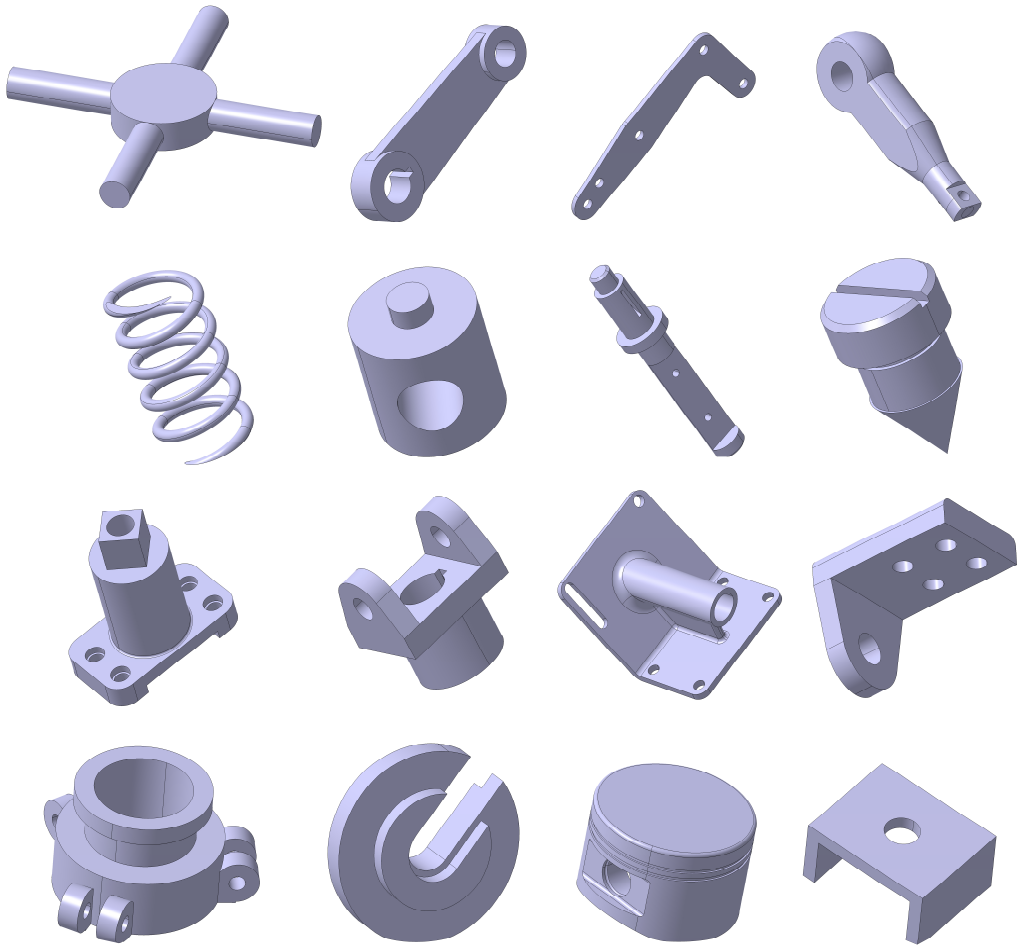

Abbildung 1.1: Auszug der Grundgesamtheit aller Bauteile

Abbildung 1.2 zeigt einen schematischen Überblick über die einzelnen miteinander verknüpften Teilbereiche. Ausgehend von einer gegebenen CAD- bzw. CAE-Datenbasis (rot) werden sechs verschiedene Ansätze zur Informationsextraktion untersucht (gelb). Diese beinhalten sowohl herstellerunabhängige Dateiformate, als auch eine Untersuchung des CATIA V5-eigenen Strukturbaums. Nach der Informationsextraktion werden die Ergebnisse (grün) aufbereitet und zu Fingerprints in Form von Zahlenfolgen und Matrizen (türkis) zusammengefasst. Der letzte Schritt der Informa-

tionsweiterverarbeitung ist in Form von gebildeten Klassen oder Suchmaschinenergebnissen auf der rechten Seite zu erkennen (violett).

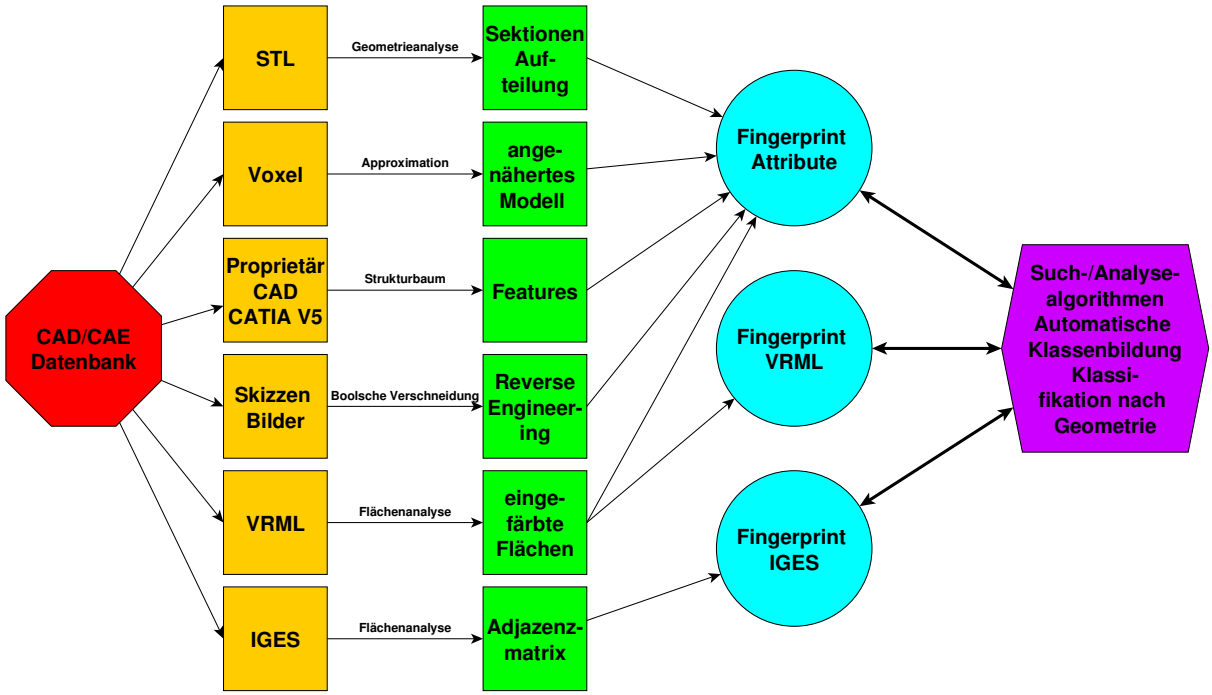

Abbildung 1.2: Schematische Übersicht

Einen noch detaillierteren Überblick liefert Tabelle 1.1. Diese ist ebenfalls in die drei Bereiche eingeteilt und verdeutlicht, welche Datentypen an welcher Stelle verarbeitet wurden. Insgesamt sind die hier aufgelisteten 29 Programme enthalten, die in den Kapiteln 4-6 zuerst konzeptionell erklärt und deren Ergebnisse dann evaluiert werden. Wie in der rechten Spalte zu erkennen ist, wurde einer besseren Übersicht dienend die Sektion der Informationsweiterverarbeitung in die Teilbereiche 'Suchmaschine' und 'Clustering' aufgegliedert.

Im abschließenden Kapitel 7 werden ein Fazit gezogen und mögliche zukünftige Forschungsaufgaben erläutert.

Nach dem Haupttext folgt der Anhang, der zum einen aus einer weiteren tabellenförmigen Programmübersicht mit Ein- und Ausgabedateien besteht, und außerdem einer Sammlung von Flussdiagrammen, die jedes Programm in einem Programmablaufplan visualisieren.

In der gesamten Arbeit sind wörtliche Zitate in kursiver Schrift hervorgehoben und im Literaturverzeichnis mit Seitenzahl vermerkt. Alle Abbildungen und Tabellen wurden selbstständig erstellt. Wissenschaftlichen Veröffentlichungen, Studentenprojekte sowie Bachelor- und Masterarbeiten, die im Zuge dieser Arbeit entstanden sind, sind mit einem Asterisk ('*') gekennzeichnet.

	Informationsextraktion	Informationsaufbereitung	Informationsweiterverarbeitung	
	Analyse	*Fingerprints*	*Suchmaschine*	*Clustering*
STL	stl.py			
Voxel	voxel.py			
Strukturbaum	read_tree.py			
Rekonstruktion	create_stl.py			
	stl_to_bool.py			
	drilling.py			
	pocket.py			
VRML	vrml.py	attributes.py	attributes_euklid.py	attributes_cluster.py
		vrml_matrix.py	attributes_pearson.py	
		simple_vrml_matrix.py	simple_vrml_matrix_compare.py	simple_vrml_cluster.py
			vrml_matrix_compare.py	vrml_cluster_quan.py
			vrml_matrix_statistic.py	vrml_cluster_qual.py
			vrml_matrix_subgraph.py	subgraph_vrml_cluster_quan.py
			vrml_matrix_scaled.py	subgraph_vrml_cluster_qual.py
IGES	iges.py	iges_matrix.py		iges_cluster.py
		simple_iges_matrix.py		simple_iges_cluster.py

Tabelle 1.1: Übersicht aller Programme

Kapitel 2. Industrielle Anwendung

2.1 CAE-Software

In diesem Kapitel soll auf die professionelle CAE-Software eingegangen werden, die sich insbesondere in großen Unternehmen in der Anwendung befindet. Üblicherweise ist es so, dass firmenintern ein Programm festgelegt und mit Zulieferern und Kunden über Austauschformate (vgl. Kapitel 2.2) kommuniziert wird. Bevor in den nächsten Sektionen die wichtigsten Programme verschiedener Hersteller umrissen werden, ist an dieser Stelle anzumerken, dass alle sich im Großen und Ganzen nur noch im Detail - insbesondere der Bedienung - unterscheiden. Der Grundaufbau ist immer in zweidimensionales Skizzieren und dreidimensionales Modellieren eingeteilt. Darüber hinaus gehören zur Standardausrüstung meist noch Module zum technischen Zeichnen, für FEM-Berechnungen (Finite Elemente Methode) und zum Zusammenbau einzelner Teile zu Baugruppen. Dennoch verfügen die meisten Hersteller in ihrer Produktpalette noch über detailliertere Programme - z.B. zur Verwaltung von Daten (PLM), Strömungssimulationen (CFD) oder FEA (Finite Element Analysis). Lediglich die Entwicklung und der Anwendungshintergrund der Programme unterscheiden sich maßgeblich.

Weiterhin ist allen Programmen gemein, dass sie neben der Fähigkeit herstellerunabhängige Austauschformate zu erzeugen, hauptsächlich in proprietäre Dateien abspeichern, die neben der Geometrie der Bauteile auch die Befehle, die zum Erzeugen nötig waren, und die Reihenfolge, in der der Benutzer vorgegangen ist, in Form eines Strukturbaumes beinhalten. Somit sind die Vorgehensweise des Anwenders bzw. die Konstruktionshistorie nachzuvollziehen und eventuelle Änderungen im Nachhinein besser zu bewerkstelligen.

2.1.1 AutoCAD

AutoCAD ist die wichtigste Software zur Erstellung von zweidimensionalen technischen Zeichnungen und wurde bereits 1982 von der Firma Autodesk entwickelt. Die aktuelle Version wird als AutoCAD 2015 bezeichnet und besitzt trotz der wichtiger gewordenen dreidimensionalen Modellierung noch immer große Marktanteile [Onst-14]. Neben einer Anwendung im ingenieurstechnischen Bereich profitieren auch weitere Industriezweige, wie z.B. das Architektur- oder Designwesen, von den vielfältigen Anwendungsmöglichkeiten.

Die zwei gängigsten Dateiformate sind DWG (Drawing) und DXF (Drawing Interchange Format), von denen sich in erster Linie das DXF-Format als standardmäßiges Austauschformat für zweidimensionale Daten auch von anderen Herstellern etabliert hat. Dennoch werden beide Formate an die jeweils aktuelle Version von AutoCAD angepasst, so dass es für die Bearbeitung auch mit Software von Fremdfirmen einer nicht allzu alten Version bedarf.

2.1.2 CATIA

CATIA wurde 1982 von der Firma Dassault Systèmes entwickelt und mittlerweile in der sechsten Version veröffentlicht. Da die Zeitabstände zwischen den Ausgaben relativ lang waren, gibt es für jede Version außerdem zusätzliche 'Releases', in denen die Gelegenheit ergriffen wurde, nur kleine Aktualisierungen vorzunehmen [Schn-16].

Ursprünglich wurde CATIA für die Luft- und Raumfahrtindustrie kreiert, was sich noch heute in der Vielzahl an Modulen für zweidimensionale Freiform- und Oberflächen widerspiegelt. Weiterhin sind neben den trivialen Arbeitsumgebungen für Konstruktion und technische Zeichnungen auch Elemente der FEM-Berechnung, Bauteilvernetzung oder Fachwerksplanung enthalten, was abgehen von der Anwendung bei den beiden größten Passagierflugzeugherstellern Airbus und Boeing auch zu einer flächendeckenden Verbreitung in anderen Industriesparten - speziell der Automobilindustrie - führte. Trotz der aktuellsten Version 6 ist CATIA V5 heute noch am weitesten verbreitet.

2.1.3 NX

Die Software NX wird vom Siemens-Konzern herausgegeben und ist der Tochterfirma 'Siemens PLM Software' zugehörig [WHDK-15]. Diese ging aus der Akquisition der Firma 'Unigraphics Solutions Corporation' hervor, die ihre Eigenentwicklung 'Unigraphics' und die aufgekaufte Software 'I-DEAS' (Integrated Design and Engineering Analysis Software) zusammenführte und zum Hauptprogramm NX verband. Die aktuelle Version ist NX 10. Sie ist historisch gesehen auf die Automobilindustrie zugeschnitten und beinhaltet neben CAE- und CAD-Modulen auch eine CAM-Arbeitsumgebung (Computer Aided Manufacturing), die die Programmierung von CNC-Maschinen (Computerized Numerical Control) erlaubt.

2.1.4 Inventor

Ebenso wie das in Kapitel 2.1.1 erwähnte AutoCAD wird Inventor von der Firma Autodesk entwickelt und ist mittlerweile in der 19. Version verfügbar [Sche-15]. Seit Version 11 sind die Releases nur noch mit den Jahreszahlen gekennzeichnet und mit der Veröffentlichung im April 2015 bei der Version Inventor 2016 angekommen. Obwohl AutoCAD mit Inventor im Besonderen durch die Ähnlichkeiten bei zweidimensionalen Zeichnungen sowie dem Gebrauch der erwähnten Datentypen verbunden ist, wurde Inventor zusätzlich entwickelt, um auf dem Markt für professionelle CAE-Software konkurrenzfähig zu sein. Der Schwerpunkt liegt auf der Anwendung im maschinenbautechnischen Bereich sowie der Erstellung von Präsentationen (Rendern). Weiterhin wird den Anwendern eine Normteilbibliothek zur Verfügung gestellt, aus der Kleinteile bei dem Zusammenbau von Baugruppen automatisch eingefügt werden können.

2.2 Herstellerunabhängige Dateiformate

Die Dateiformate der CAE-Dateien lassen sich in zwei Kategorien einteilen. Zum einen hat jedes CAE-Programm die Möglichkeit die Anwendungen des Benutzers unter dem herstellerspezifischen Format abzuspeichern, welches meistens am umfangreichsten und informationsreichsten ist. Weiterhin existieren die herstellerunabhängigen Austauschformate, auf die in diesem Kapitel näher eingegangen werden soll.

Um die grundsätzliche Problematik zu verdeutlichen wird hier Bezug auf CATIA V5 genommen. Dort existieren eine Reihe von verschiedenen Dateiendungen. Die wichtigsten sind CATPart für die Konstruktion von einzelnen Bauteilen, CATProduct für den Zusammenbau dieser einzelnen Teile zu Baugruppen, CATDrawing für zweidimensionale technische Zeichnungen und CATAnalysis für FEM-Berechnungen. In diesen Datenformaten sind alle Schritte, die der Konstrukteur bei der Erstellung der Datei durchgeführt hat, abgespeichert und im CATIA-eigenen Strukturbaum nachzuvollziehen. Auch bei CAE-Programmen anderer Hersteller verhält es sich mit der Abspeicherung ähnlich. Problematisch gestalten sich die Erweiterungen eines jeden CAE-Programmes, die im Laufe weiterer Versionen oder Veröffentlichungen (Releases) hinzukommen. Die aktuellsten Versionen sind meist abwärtskompatibel, was das Problem der Verwendung alter Dateien löst. Wenn allerdings eine Datei, die mit der neuesten Version eines Programmes erstellt wurde, z.B. beim Austausch von Konstruktionsdaten zwischen zwei Unternehmen, mit einer älteren Version geöffnet werden soll, so ist dies meist nicht möglich, da noch nicht alle Fähigkeiten in der älteren Version implementiert sind.

Generell gestaltet sich der Datenaustausch zwischen verschiedenen Unternehmen, die in der Industrie zusammenarbeiten, als schwierig, da es eine Vielzahl von Softwareherstellern und Programmen gibt, die alle unterschiedliche Fähigkeiten und Funktionen besitzen (vgl. Kapitel 2.1). Aus diesem Grund wurden die hier beschriebenen Austauschformate entwickelt, die unabhängig vom Hersteller die Dateien in immer gleicher Art und Weise abspeichern.

Weiterhin besteht aber auch bei diesen Dateiformaten das Problem, dass sie zwar von den meisten Programmen geöffnet und bearbeitet werden können, aber nur wenige Informationen beinhalten. So können meist die Erstellungsschritte, die normalerweise im Strukturbaum enthalten sind, nicht mehr nachvollzogen und somit auch nicht mehr verändert werden. Auch ist die Art der Abspeicherung größtenteils grundverschieden. Bei den Herstellerformaten werden normalerweise nur die Arbeitsschritte des Konstrukteurs abgespeichert aber nicht die eigentliche Geometrie, die dadurch entstanden ist. Dementsprechend ist der Speicherplatzbedarf der entstehenden Dateien sehr klein, da er kaum von den Abmessungen des Bauteiles abhängt. Die

Austauschformate können die Fähigkeiten eines jeden CAE-Programmes und deren interne Syntax nicht berücksichtigen und müssen deshalb die Form und den Aufbau bzw. die Geometrie der Teile abspeichern. Je nach Komplexität und Größe wird der benötigte Speicher unverhältnismäßig groß.

Auch wenn die hier beschriebenen standardisierten herstellerunabhängigen Austauschformate viele Vorteile bieten, sollte bei einem Datenaustausch zwischen Anwendern mit den gleichen CAE-Systemen immer das systemeigene, gehaltvollere Format bevorzugt werden.

Wenn die Entstehungsgeschichte der verschiedenen CAD-Systeme betrachtet wird, so sind generell zwei verschiedene Modellierungsphilosophien zu erwähnen, die sich prinzipiell unterscheiden. Bei der ersten Vorgehensweise mit dem Namen B-Rep (Boundary Representation) wird ähnlich wie bei der Freiformflächenmodellierung lediglich die Bauteilkontur modelliert, wobei darauf zu achten ist, dass eine geschlossene Hülle entsteht. Die Abspeicherung der Geometrien findet textbasiert in Form von dreidimensionalen Koordinaten statt, die aus Punkten, Kanten und Flächen aufgebaut sind, desweiteren verbunden werden und somit die Gesamtgeometrie definieren. Das Modell wird folglich aus einer Hierarchie von Punkten (nulldimensional), Kanten (eindimensional) und Flächen (zweidimensional) definiert, die in Form von Tabellen oder Dictionaries abgespeichert wird. Abbildung 2.1 zeigt als Beispiel eine Schraube mit Sechskantkopf. Auch die Form des runden Schafts wird durch zwei Zylinderhalbschalen dargestellt.

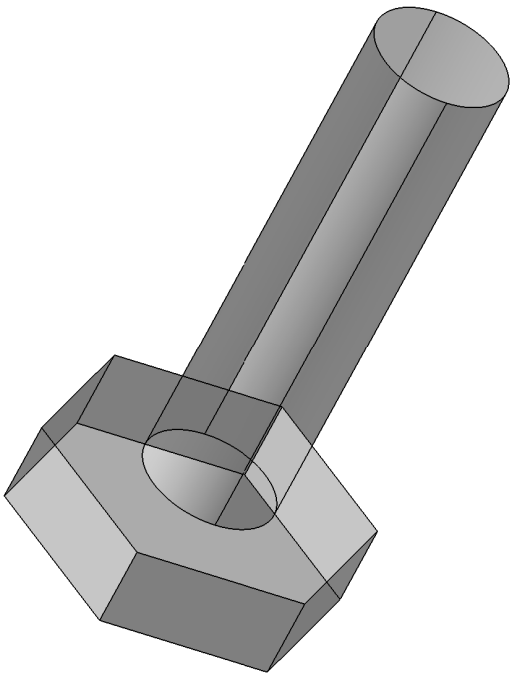

Abbildung 2.1: B-Rep-Darstellung einer Schraube

Die zweite Vorgehensweise wird als CSG (Constructive Solid Geometry) bezeichnet. Dabei wird das gesamte Modell aus Grundkörpern (Bodies) aufgebaut, die das dreidimensionale Pendant zu den geometrischen Primitiven darstellen und mit Hilfe von Booleschen Operationen zusammengefügt, voneinander abgezogen oder verschnitten werden.

Ein anschauliches Beispiel wird in Abbildung 2.2 dargestellt. Im oberen Bereich sind zwei relativ einfache Grundkörper in Form eines extrudierten Sechskants sowie eines Schaftes zu erkennen (a). Durch die Kombination beider Körper mit der Booleschen Operation 'Zusammenfügen' entsteht eine Sechskantschraube (b). Die Differenz zwischen Sechskant und Schaft ergibt eine Mutter (c), während die Schnittmenge eine Art Gewindestift entstehen lässt (d).

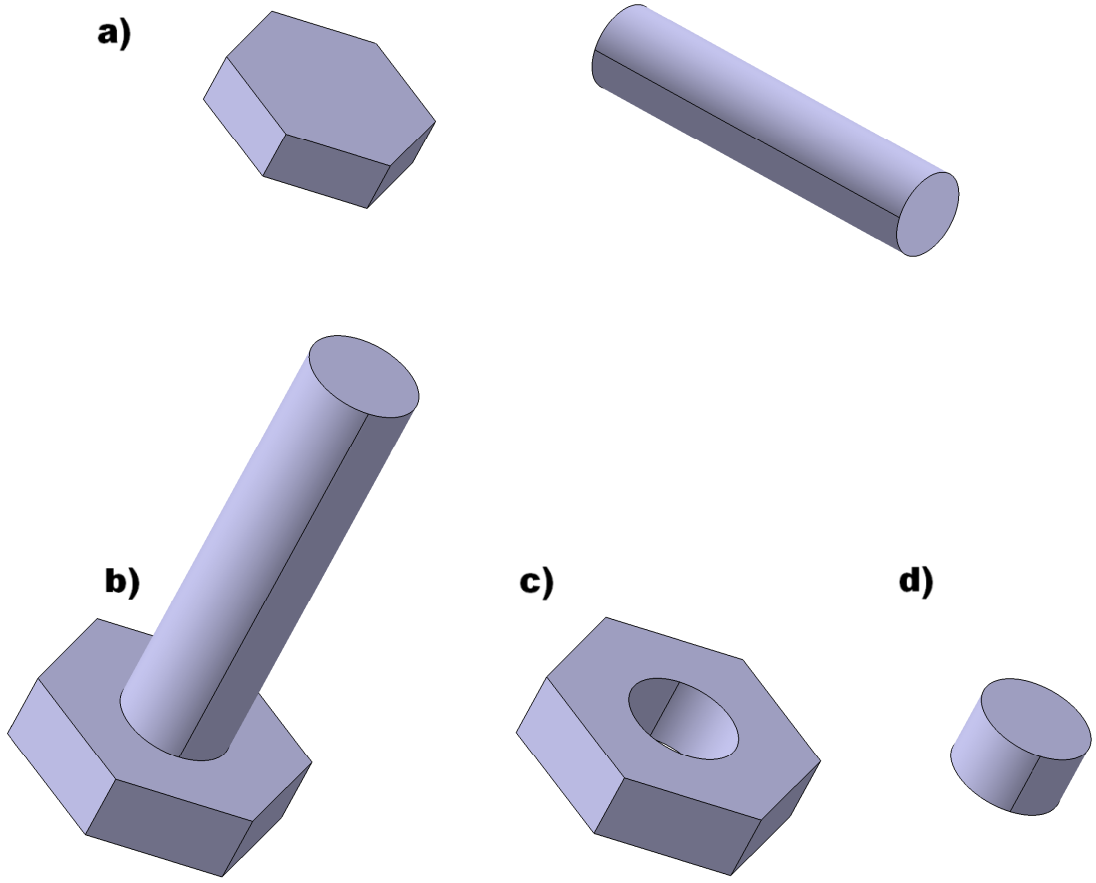

Abbildung 2.2: Boolsche Operationen

Der Vorteil des CSG besteht darin, dass der Konstruktionsprozess durch eine simple Definition jener Grundkörper ablaufen kann und die Modelle erst durch eine geschickte Verknüpfung eine gewisse Komplexität erreichen.

Bei der Betrachtung eines fertigen Modelles ist auf den ersten Blick meist nicht mehr nachzuvollziehen welche Schritte der Konstrukteur durchgeführt hat. Durch die

Vielzahl an Kombinationsmöglichkeiten ist insbesondere die Reihenfolge variabel und lässt so verschiedene Konstruktionsansätze zu. Moderne CAD-Systeme bieten allerdings heutzutage eine Vielzahl weiterer Funktionen, die die Handhabung beim Modellierungsprozess zusätzlich vereinfachen und einen ausschließlichen Gebrauch von Booleschen Operationen überflüssig machen.

2.2.1 STL

Das STL-Format wurde entwickelt, um komplexe dreidimensionale Formen annähernd darzustellen und ist heute das Standardeingabeformat für Rapid-Prototyping-Verfahren [ChLL-10]. Die Grundidee ist es, ein beliebiges Modell am CAD-System zu modellieren, das später mit Hilfe eines 3D-Druckers vorzugsweise aus Kunststoffgranulat schichtweise aufgebaut wird (Stereolithografie). Die meisten 3D-Drucker verarbeiten die Bauteilgeometrien im STL-Format, was vom Benutzer eine Übertragung bzw. Abspeicherung der proprietären Dateien ins STL-Format bedarf. Die Herstellung des ausgedruckten Kunststoffmodells erfolgt aus Granulat, das an den richtigen Stellen von einem Laser kurzzeitig erhitzt wird, sich so verflüssigt und bei der Abkühlung mit der Gesamtfigur verschmilzt. Mit diesem Verfahren wird zuerst am Computer das Modell in viele dünne Schichten aufgeteilt, die vom Laser nacheinander abgearbeitet werden und das Bauteil so stufenweise aufbauen. Ebenso wie bei der Darstellung von Kreisen oder Bögen an Computermonitoren mit der Hilfe von Pixeln, ist es faktisch unmöglich, Rundungen realitätsgetreu zu erschaffen. Ein stufenweiser Aufbau stellt immer nur eine Annäherung an die ideale Form dar, was dazu führt, dass das gedruckte Modell nach der Fertigung ggf. noch nachbearbeitet werden muss.

Eine solche Approximation an die ideale Form spiegelt sich auch in der Gestalt von Geometrien im Quelltext der STL-Dateien wieder. Beliebige Konturen werden durch die Darstellung ihrer Oberflächen in Form von kleinen Dreiecken aufgebaut, die immer aus drei Eckpunkten (Vertices) und ihrem Normalenvektor definiert und in unterschiedlichen Winkeln miteinander verknüpft sind. Die Geometrie eines Bauteiles wird im Quelltext so nur durch die Definition ihrer Oberfläche spezifiziert, welche in viele kleine Dreiecke unterteilt wird, die durch die Abspeicherung ihrer Eckpunkte in X-, Y- und Z-Koordinaten im virtuellen Raum beliebige Formen bilden können. Ähnlich wie bei einer Pixeldarstellung von runden geometrischen Primitiven entsteht so kein ideales, sondern basierend auf dem Feinheitsgrad dieser Dreiecke nur eine Annäherung an das Original. Im Idealfall sollten beim Rapid Prototyping die Oberflächen durch die Dreiecke aber so fein strukturiert sein, dass ein polygonaler Aufbau im fertigen Modell nicht mehr zu erkennen ist.

Die Tatsache, dass ein Eckpunkt Teil von mehreren Dreiecken sein kann und seine Koordinaten im Quelltext der STL-Datei somit mehrfach aufgelistet werden, kann bei großen komplizierten Bauteilen zu unverhältnismäßigen Datenmengen führen.

2.2.2 MSH

Austauschformate für dreidimensionale Geometrien sind auch im Bezug auf FEM-Anwendungen interessant, da eine Vernetzung des zu untersuchenden Bauteiles bei der Vorbereitung zur eigentlichen Berechnung erfolgt und das Modell an dieser Stelle meist automatisch in verschiedenförmige Knoten und Verbindungen eingeteilt wird, die ähnlich wie bei der Oberflächendefinition das Bauteil als Volumenkörper darstellen. Ein solches Standardformat für die Vernetzung von Bauteilen ist das MSH-Format. Es wird hauptsächlich bei der Anwendung von professionellen Vernetzungsprogrammen erstellt und ist das native Datenformat des Programmes Gmsh [GeRe-15]. Es kann des Weiteren auch zur Übermittlung von geometrischen Formen verwendet werden.

Ähnlich wie beim STL-Format sind im Quelltext von MSH-Dateien auch Vertices durch ihre dreidimensionalen Koordinaten im virtuellen Raum definiert. Neben weiteren Informationen, wie z.B. der Versionsnummer oder physikalischen Bezeichnungen, ist jede MSH-Datei hauptsächlich in die beiden Abschnitte '$Nodes' und '$Elements' unterteilt. Im ersten Bereich werden alle Knoten aufgelistet, nummeriert und ihre Position durch die Angabe der X-, Y- und Z-Koordinaten bestimmt. Der zweite Teil greift diese Knoten wieder auf und definiert ihre Zugehörigkeit zu Elementen verschiedenster Form. Je nach Art der FEM-Berechnung und der Form des Netzes ist der Elementtyp ausschlaggebend für plausible Ergebnisse. So wird etwa das in Kapitel 2.2.1 genannte Problem der geometrischen Verrundungen durch eine Verfeinerung des Netzes an den entsprechenden Stellen gelöst.

2.2.3 XML

Das XML-Format (Extensible Markup Language) wurde für die Abspeicherung von Informationen in hierarchischen Strukturen vom W3C (World Wide Web Consortium) entwickelt [Bech-09]. Von einem einzigen Eintrag ausgehend, der beispielsweise als Überschrift der XML-Datei angesehen werden kann oder auch als Wurzel bezeichnet wird, sind alle weiteren Einträge - die sogenannten 'Kinder' - abgezweigt. Diese werden in Form von 'Tags' in der Gestalt von '<tag>...</tag>' festgelegt. Eine XML-Datei kann folglich nur eine einzige Wurzel, allerdings theoretisch unendlich viele Kinder und Kindeskinder haben. Ein Eintrag, der in der Hierarchie direkt oberhalb eines Kindes steht, wird als Elternknoten bezeichnet. Zur zusätzlichen Informationsbeigabe der Knoten dient die Möglichkeit der Attributsdefinition und des Texteintrages. Dies soll am folgenden Beispiel eines Buches verdeutlicht werden:

```
<book author="Boris Pasternak" title="Doktor Schiwago">
  <chapter number="1" title="Der Fünf-Uhr-Schnellzug">
    Sie gingen und gingen und sangen...
  </chapter>
</book>
```

Das besondere an diesem Beispiel ist außerdem der Sachverhalt, dass es 'wohlgeformt' ist. Das bedeutet es besitzt genau einen Wurzelknoten, jeder Eintrag hat einen Tag zum öffnen (*<tag>*) und einen Tag zum schließen (*</tag>*) bzw. einen in sich geschlossenen Tag (*<tag/>* - wird benutzt wenn kein Texteintrag vorliegt), die einzelnen Einträge sind wie bei der mathematischen Klammersetzung nicht in der jeweiligen, sondern nur in der vorherigen Ebene geschachtelt und kein Tag darf Attribute mit dem selben Namen doppelt besitzen.

Informationen oder Datenbanken sind in der Informatik und in der Technik häufig in derartigen hierarchischen Strukturen vorzufinden.

2.2.4 SVG

Das SVG-Format (Scalable Vector Graphics) basiert auf den gleichen Prinzipien wie das in Kapitel 2.2.3 beschriebene XML-Format [Fibi-02]. Die grundlegenden Elemente, wie z. B. die Tags zum Öffnen und Schließen der Einträge, sind ebenso enthalten wie die Wurzel oder die einzelnen Knoten mit ihren Werten und Attributen. Der Unterschied besteht jedoch darin, dass eine SVG-Datei entweder mit einem Texteditor geöffnet werden kann, um den Quelltext angezeigt zu bekommen, oder mit einem Programm, das in der Lage ist den Quelltext zu übersetzen und die Datei in grafischer Form zu visualisieren (wie etwa ein Plugin für einen Internetbrowser).

Der Vorteil dabei ist, dass Grafiken nicht in Form von Pixeln in verschiedenen Farben bestimmt werden, sondern dass von Bildelementen wie Linien, Kreisen oder Ellipsen nur die Parameter zur Definition abgespeichert werden. Eine Linie wird beispielsweise einzig und allein durch die Koordinaten der zwei Endpunkte definiert. Zusätzlich können aber auch Attribute, die bei der Darstellung hilfreich sein können, wie die Strichdicke oder die Farbe angegeben werden. Dadurch, dass jedes Bildelement durch eine immer ähnliche Menge an Zahlen definiert wird, spielt es im Hinblick auf den Speicherplatz keine Rolle, ob der Kreis große oder kleine Maße besitzt.

Bei der Anzeige der Grafiken wird außerdem ein weiterer gravierender Vorteil deutlich. Wenn ein Ausschnitt vergrößert dargestellt oder das ganze Bild herangezoomt werden soll, dann ist dies möglich, ohne dass die Pixel, die für die Darstellung benutzt wurden, vergrößert angezeigt werden (Skalierbarkeit). Ein Nachteil der SVG-Dateien ist allerdings die fehlende Möglichkeit der Darstellung von komplizierteren Bildern oder Fotos. Neben normalen Anzeigemöglichkeiten sind jedoch auch Animationen oder grafische Effekte mit Hilfe des Quelltextes definierbar.

2.2.5 VRML

Beim VRML-Format (Virtual Reality Modeling Language) handelt es sich um ein dreidimensionales Grafikformat, das in der Version 2.0 bereits 1997 entwickelt wurde, um Animationen oder komplexere geometrische Formen mit Texturen und sogar Lichtquellen darstellen zu können [AmNM-97]. Ähnlich wie das STL-Format (vgl. Kapitel 2.2.1) stellt VRML bei dreidimensionalen Objekten ebenfalls nur die Oberfläche zweidimensional dar, so dass ein räumliches Netz mit einer unendlich dünnen Hülle modelliert wird. Darüber hinaus werden alle Flächen in der Visualisierung ebenfalls aus Punkten und kleinen Dreiecken aufgebaut, so dass problematischere Strukturen aus Polygonen zusammengesetzt werden.

Der Quelltext von VRML-Dateien ist ebenfalls strukturiert aufgebaut. Für einfache geometrische Formen sind vorgefertigte Bausteine definiert. Alle weiteren Körper, die auf komplexere Art und Weise miteinander verbunden sind, können mit Punkt-, Linien- oder Flächensets bestimmt werden, die wiederum die dreidimensionalen Koordinaten beinhalten und denen weitere Eigenschaften, wie z.B. eine Farbe, zugewiesen werden können.

VRML-Dateien besitzen die Namenserweiterung WRL (World).

2.2.6 IGES

IGES-Dateien werden unter dem Kürzel IGS abgespeichert und ihr Standard wurde vom NBS (National Bureau of Standards) - heute NIST (National Institute of Standards and Technology) - erstmalig 1980 herausgegeben. Später wurde die Weiterentwicklung und Pflege vom US PRO (United States Product Data Association) übernommen und die aktuellste Version 5.3 zuletzt 1996 herausgegeben [USPD-96]. Die Aktualisierung wurde erst im Jahr 2006 aufgegeben, da andere Datenformate, wie z.B. STEP (vgl. Kapitel 2.2.7), eine immer größere Rolle spielten.

Dennoch sind heutzutage die meisten CAE-Systeme in der Lage, die Bauteile ins IGES-Format zu exportieren sowie IGES-Dateien zu öffnen. Eine Abspeicherung erfolgt durch die Definition der Oberfläche bzw. deren Elemente. So entsteht eine Hülle des Bauteils, die aus zweidimensionalen Flächen im dreidimensionalen Raum die Geometrie definiert. Dennoch ist zu erwähnen, dass die meisten modernen CAD-Systeme in der Lage sind einen nicht parametrischen Volumenkörper aus dieser Hülle zu erzeugen, der im Programm mit Material gefüllt ist und somit auch die entsprechenden physikalischen Eigenschaften besitzt.

Der IGES-Standard wurde vornehmlich für den Informationsaustausch von CAD-Daten geschaffen und der Quelltext einer Datei kann somit in die textliche Darstellung von zwei- und dreidimensionalen geometrischen Primitiven unterteilt werden. Anders als bei den bereits erwähnten STL- und VRML-Dateiformaten wird somit der Speicherplatzbedarf möglichst gering gehalten, was das Format handlicher für die Anwendung in der Industrie macht.

2.2.7 STEP

Das STEP-Format kann historisch gesehen als eine Weiterentwicklung des IGES-Formats angesehen werden, da die PDES-Organisation (Product Data Exchange using STEP) an der Erarbeitung und Verbreitung von beiden Standards beteiligt war [AnTr-00]. Die Besonderheit des STEP-Formats ist die Tatsache, dass neben geometrischen Informationen eines Bauteils auch weitere Produktdaten, wie z.B. Volumeninformationen oder Materialeigenschaften, abgespeichert werden können. Dies bietet die Gelegenheit beim Datenaustausch weitere Informationen, z.B. über die Fertigung, zu übermitteln und ermöglicht eine Bauteilpflege von STEP-Dateien in PDM-Systemen (Product Data Management). Im Gegensatz zum IGES-Format können auch Volumenmodelle definiert werden.

Die aktuelle Version AP 242 wurde vom ISO (International Organization for Standardization) 2014 herausgegeben.

2.2.8 JT

In den letzten Jahren hat sich neben IGES und STEP auch das JT-Dateiformat (Jupiter Tessellation) zum Datenaustausch zwischen einander fremden CAD-Systemen durchgesetzt [Siem-16]. Obwohl es sich dabei genau genommen um kein herstellerunabhängiges Dateiformat handelt, da es von Siemens im Bezug auf NX akquiriert wurde, fungiert es heute de facto als Austauschformat und kann von vielen CAE-Softwaresystemen mit speziellen Plugins importiert und auch exportiert werden.

Speziell für die Weitergabe ingenieurstechnischer Informationen ist das JT-Format gut geeignet, da neben einer geometrischen Darstellung in tesselierter oder exakter Form auch hier eine Unterbringung zusätzlicher Informationen, wie z.B. Toleranzen, Maße oder Fertigungsdaten, möglich ist.

Bereits im Jahre 1998 wurde ein erster Standard definiert und im Laufe der Zeit mit dem Schwerpunkt auf technische Anwendungen stetig weiterentwickelt. Da das JT-Format industriell entwickelt wurde, besteht im Vergleich zum VRML-Format der Nachteil, dass der Quelltext nur mit einer umfangreichen Dokumentation zugänglich ist. Da beispielsweise der Quelltext des VRML-Formats relativ einfach zu interpretieren ist, entstanden mit Hilfe der Entwicklergemeinde viele Anwendungen aus unterschiedlichen fachlichen Richtungen. Obwohl ähnliche Ansätze beim JT-Format noch zu vermissen sind, bietet sich unter der Voraussetzung einer umfassenden Dokumentation eine programmiertechnische Weiterverarbeitung des JT-Formates an. Ein Alleinstellungsmerkmal ist die Unterbringung von Metadaten, auf die eventuell ebenfalls zugegriffen werden kann.

2.3 Datenbanken

In diesem Kapitel soll auf die Grundstrukturen und den Aufbau von Datenbanksystemen eingegangen werden, da im zweiten Teil der Arbeit noch einmal Bezug darauf genommen wird. Es handelt sich dabei um einen Oberbegriff, der sich nach Gabriel und Röhrs in die folgenden drei Unterbegriffe einteilen lässt [GaRö-94]:

1. Die Datenbasis bzw. Datenbank beinhaltet alle Daten bzw. Dateien, die im Zuge des Datenbanksystems verwaltet und administriert werden sollen.
2. Das Datenbankverwaltungssystem übernimmt die Steuerung aller Prozesse, die für das Management notwendig sind.
3. Die Datenbankkommunikationsschnittstelle ist für alle Zugriffe, die von außerhalb der Datenbank kommen, zuständig und gewährleistet den Informationsaustausch mit externen Quellen und Anfragen, wie z.B. den menschlichen Benutzern oder anderen Programmen.

Im Allgemeinen ist jedes Datenbanksystem nach diesem Schema aufgebaut. Neben den Managementprogrammen, die Vorgänge, wie Verschiebungen, Neuerstellungen, Löschungen oder Kopien, übernehmen und automatisieren, kann die eigentliche Datenbasis auch abgespalten und separat betrachtet werden. Im Grunde genommen ist die Festplatte eines jeden Computers schon eine Datenbank, da alle enthaltenen Objekte in Verzeichnissen kategorisiert sind und der Benutzer die typischen Arbeitsschritte, die normalerweise vom Managementsystem übernommen werden, manuell durchführen kann.

An dieser Stelle soll außerdem der Begriff der relationalen Datenbanken erwähnt werden. Nach Meier werden die Einträge, die nach diesem Konzept aufgebaut sind, in Tabellenform abgespeichert [Meie-04]. Diese Tabellen werden Relationen genannt und sind in Zeilen (Tupel bzw. Datensatz) und Spalten (Attribut) aufgebaut. Ähnlich wie bei einer Sachmerkmalsleiste (vgl. Kapitel 2.4) besitzen die Einträge verschiedene Eigenschaften, die in den zugehörigen Zellen definiert werden. Der sogenannte Schlüssel identifiziert jeden Datensatz und ist somit einzigartig. Weiterhin können für eine genauere Darstellung von den Beziehungen der Relationen untereinander diese miteinander verknüpft werden und so komplexere Sachverhalte darstellen bzw. abspeichern.

Von besonderem Interesse sind Datenbanken, deren Basis aus CAD-technischen Formaten oder Produktdateien besteht. Wenn in der Theorie die Datenbasis nur aus einem Format aufgebaut ist, verhält sich das Management, das vom Datenbankverwaltungssystem übernommen wird, verhältnismäßig einfach. In der Praxis ist es allerdings so, dass viele verschiedene Dateiformate, wie z.B. Datenblätter, Dokumentationen, Zusatzinformationen oder CAD-Dateien unterschiedlicher Austauschformate und

vieler Hersteller, enthalten sind. Bei der Produktentwicklung fallen neben den normalen Konstruktionsdateien, in denen die jeweiligen Einzelteile als dreidimensionale Modelle enthalten sind, auch weitere Formate an, die mit dem gleichen CAE-Programm erstellt werden können. Dazu zählen Baugruppen, in denen mehrere Einzelteile zusammengesetzt werden, zweidimensionale technische Zeichnungen, die normgerecht sind oder sich an den firmeninternen Richtlinien orientieren, FEM-Berechnungen, bei denen die Belastungen und Verformungen der Teile oder Baugruppen im späteren verbauten Zustand berechnet werden, oder Animationen bzw. Präsentationen, die ebenfalls mit dem CAD-Programm erstellt werden und Bewegungen oder Kamerafahrten in Form von Videos darstellen können.

Hinzu kommen externe Dateiformate wie Kataloge, Tabellen oder Austauschformate, die Zusatzinformationen über Aufträge oder Kundendaten enthalten. Besonders bei der Berücksichtigung der PLM-Methode fallen neben technischen Entwicklungsdateien auch viele andere Informationen, z.B. über das Marketing oder den Vertrieb, der produzierten Produkte an.

Je nach der Komplexität des Aufbaus steigt auch der Schwierigkeitsgrad einer funktionellen und effektiven Verwaltung. Es muss die Frage geklärt werden, nach welchen Gesichtspunkten eine Datenbank sortiert werden soll, so dass alle Bestandteile zu jedem Zeitpunkt schnell wieder aufzufinden und leicht zugänglich sind.

Besonders wenn die Datenbank manuell verwaltet werden soll, kann eine Fehlstrukturierung je nach Größe zu einer massiven Verlängerung der Bearbeitungszeiten oder im Extremfall zum Verlust von Daten führen. Doch auch die Datenbankverwaltungssysteme bearbeiten nur die Schritte, die normalerweise manuell durchgeführt werden, verringern so den Zeitaufwand und steigern die Effektivität. Eine übersichtliche Strukturierung und eine vorgegebene, definierte Prozessfolge der Grundschritte sind folglich für die menschliche und algorithmische Bearbeitung essentiell.

Es existieren viele verschiedene Ansätze für die Klärung der Frage, nach welchen Vorgaben eine Datenbank aufgebaut sein sollte. Ein wichtiger Bestandteil für die zweckmäßige Strukturierung sind Nummernsysteme (vgl. Kapitel 2.4), die, so wie die Datenbasis selbst, nach vielen verschiedenen Ansätzen, wie z.B. chronologisch, alphabetisch, nach Kundenaufträgen oder nach den Abteilungen innerhalb einer Firma, organisiert werden können. Oft wird eine Kombination diverser Kriterien angewendet. Idealerweise sollten schon beim Aufbau der Zweck und die späteren Anforderungen bekannt sein, so dass das Nummernsystem und die damit einhergehende Datenbank dementsprechend angepasst werden können. So kann etwa ein Verwaltungssystem sowohl zeitlich als auch unterteilt nach den unterschiedlichen Firmenbereichen segmentiert sein.

Großunternehmen, die Firmenstandorte in verschiedenen Städten oder sogar in mehreren Ländern unterhalten, sind heutzutage mit Hilfe der Cloud-Computing-Technologie aber in der Lage ihre Datenbanken zu zentralisieren bzw. zu vereinigen. Wenn verschiedene Quellen oder in diesem Fall Datenbasen unterschiedlichen Inhaltes zusammengeführt werden sollen, spricht man von Informationsintegration. Die Schwierigkeit besteht darin, dass Dateien unterschiedlichen Aufbaus oder Formats miteinander in Einklang gebracht werden müssen, so dass eine einfache Auffindbarkeit und Zugänglichkeit erhalten bleiben.

Wenn dieser Vorgang innerhalb eines Unternehmens geglückt ist, bietet dies für die alltägliche Anwendung und vor allem für den Datenaustausch sehr große Vorteile. Praktisch gesehen können so Unternehmensstandorte, die weit voneinander entfernt sind, zusammenarbeiten und innerhalb kürzester Zeit die erarbeiteten Ergebnisse weltweit untereinander austauschen. Eine Neustrukturierung kann anhand einer aktuellen Kopie erfolgen, die mit den gewünschten Änderungen die alte Version am Ende überschreibt.

Wenn aus einem Datenbanksystem nur Informationen abgefragt werden sollen, ohne dass die Gefahr besteht den Inhalt zu verändern, wird die Vorgehensweise der 'read-only'-Kopie angewendet. Somit können Informationen nur eingesehen werden und das Datenbankverwaltungssystem erstellt innerhalb kürzester Zeit eine Kopie des vom Benutzer angefragten Inhaltes, die dann übermittelt aber nicht verändert werden kann. Bezüglich IT-Sicherheit liefert Eckert eine Definition [Ecke-09]:

> *Um mit der erstellten Kopie aussagekräftige Analysen vornehmen zu können, ist es notwendig, die Kopie in einem read-only Modus zu mounten, um zu verhindern, dass die Untersuchungsergebnisse durch versehentliche Veränderungen an den Datenbeständen unbrauchbar werden.*

Hier soll vor allem der Schwerpunkt auf die Verwaltung von Systemen gelegt werde, die sich in der Industrie in der Anwendung befinden und dementsprechend mit vielen verschiedenen Dateiarten und -formaten gefüllt sind. Es müssen also nicht nur die gewünschten Informationen analysiert, sondern auch per DM zuvor aus den gegebenen Informationsmengen möglichst sauber herausgefiltert werden, um sie dann klassifizieren zu können.

Eine solche technische Sortierung bzw. Erkennung der funktionellen Eigenschaften eines jeden Produktes soll möglichst automatisch ablaufen. Die Kriterien, die dabei untersucht werden, könnten beispielsweise Abmessungen (die Maximalmaße in X-, Y- und Z-Richtung werden als 'Bounding Box' bezeichnet), Maße, Toleranzen, Materialien, Gewicht, Oberfläche, Schwerpunkt, Form oder Funktion sein. Unter automatischem Ablauf eines solchen Klassifikationsalgorithmus wird die unabhängige Untersuchung dieser Produkteigenschaften bezeichnet. Der Entwickler muss also bei der

Konstruktion eines jeden Teiles nicht diese Spezifikationen manuell eingeben, sondern das System soll, erst nachdem der Benutzer seine Such- bzw. Klassifikationsanfrage gestartet hat, all diese technischen Aspekte automatisch untersuchen und erkennen.

An dieser Stelle spielt der Begriff der Features eine wichtige Rolle. Dieser wird von Vajna et al. folgendermaßen definiert [VWBZ-09]:

> *Ein Feature ist ein informationstechnisches Element, das Bereiche von besonderem (technischem) Interesse (nicht ausschließlich Geometrie) von einzelnen Produkten darstellt. Es wird durch die Summe von Eigenschaften eines Produktes beschrieben. Die Beschreibung beinhaltet relevante Eigenschaften selbst, deren Werte sowie deren Beziehungen (Relationen und Constraints).*

Bauteile werden in der virtuellen Konstruktion durch geometrische Features, wie z.B. Extrusionen, Drehungen, Verrundungen oder Bohrungen, aufgebaut. Für den technischen Aufbau einer Datenbasis ist dies von großem Interesse, da die Features ein Klassifikationspotenzial bezüglich der Funktion des spezifischen Teiles und vor allem dessen Form zulassen. Als Beispiel könnten diesbezüglich Schrauben genannt werden, die alle aus ähnlichen Features wie Drehungen und Gewinden aufgebaut sind, somit immer eine ähnliche Form aufweisen und sich auch in ihrer Funktion nicht wesentlich unterscheiden. Dennoch können auch hier die Klassengrenzen noch schärfer gezogen werden und Schrauben weiter, z.B. nach Größe, Norm, Material, Gewindeart oder Schraubenkopf, unterschieden werden.

An diesem einfachen Beispiel wird deutlich, dass schon eine technische Klassifikation von simplen Bauteilen nach vielen verschiedenen Kriterien erfolgen kann und dies sowohl vom Anwender als auch vom Such- bzw. Klassifikationsalgorithmus selber berücksichtigt werden muss.

Anstelle einer Suche kann auch die Datenbank an sich schon in diese technischen Klassen unterteilt sein. Je nach den Anforderungen, die an das Datenbanksystem gestellt werden, ist eine technische Einteilung in funktionelle Kategorien, z.B. für einen Großhändler, sinnvoll, während eine Staffelung nach Erstellungsdatum und unternehmensinterner Abteilung in einem konstruierenden und produzierenden Großbetrieb zweckmäßiger sein kann.

2.4 Nummernsysteme

Gängige Praxis in großen Unternehmen ist es, alle Konstruktionen bzw. Dateien, die von den Mitarbeitern erstellt werden, neben dem Namen auch mit einer Nummer zu versehen, die im Grunde genommen einen Code darstellt, der gewünschte Zusatzinformationen beinhaltet. Neben den üblichen administrativen Informationen, wie z.B. Erstellungsdatum, Abteilung innerhalb des Unternehmens, zugehörige Produktlinie oder Zulieferer, können eben auch Informationen über die spätere Produktklasse oder sogar technische Spezifikationen in einem solchen Nummernsystem untergebracht werden.

Die Grundlage für technische Nummernsysteme sind sogenannte Sachmerkmalsleisten. Wie am Beispiel in Tabelle 2.1 zu erkennen ist, sind diese rasterförmig aufgebaut und beinhalten möglichst detaillierte Informationen, passend zu den einzelnen Spaltenüberschriften. Ein Sonderfall einer Sachmerkmalsleiste stellt beispielsweise das Schriftfeld einer technischen Zeichnung dar. Dort sind ebenfalls in Tabellenform Informationen über den Ersteller, das Halbzeug, das Material, die zugrunde liegende Norm oder die Fertigungsmaschinen enthalten. All diese Zusatzinformationen dienen neben der eigentlichen technischen Zeichnung sowohl den Kommunikationszwecken zwischen verschiedenen Abteilungen eines Unternehmens als auch einer guten Verständlichkeit für Außenstehende.

Sachmerkmalsleiste für Getriebe 18011987					
Kennziffer	A	B	C	D	E
Name	Wellen	Zahnräder	Verkleidung	Konsole	Lager
Beschreibung	Hohlwellen	Stirnräder	Bleche	Bedienelement	Halterung
Anzahl	2	2	5	1	1
Gewicht	10kg	5kg	1kg	2kg	3kg
Fertigung	Gedreht	Gefräst	Gewalzt	Spritzguss	Gegossen

Tabelle 2.1: Sachmerkmalsleiste

So wie der Aufbau von Datenbanksystemen und Produktklassen muss ein neu zu entwickelndes Nummernsystem auch auf die zweckmäßigen Anforderungen ausgelegt werden. Je nach Bedarf kann ein Nummernsystem also auch verschiedene Absichten erfüllen, aber muss nicht zwangsläufig alle technischen, administrativen oder chronologischen Spezifikationen eines Produktes enthalten. Arnold et al. ordnen einem Nummernsystem die Aufgaben der Identifikation, Klassifikation, Information und Kontrolle zu und listen die folgenden Ziele auf, die durch die Anwendung erreicht werden sollen [ADEK-05]:

- *Vereinfachung der Informationsverarbeitung*
- *Vereinfachung der Ablauforganisation*
- *Verbesserung der Kommunikation zwischen Unternehmensbereichen, Kunden und Lieferanten*
- *Verbesserung des Ordnungswesens*
- *Reduzierung der Personalkosten*
- *Reduzierung von Änderungen*
- *Verbesserung technischer Einrichtungen*

Wenn die Klassifikationsnummer eines Produktes z.B. das Erstellungsdatum enthält, werden für die Entschlüsselung dieses Wissens keine bestimmten Mechanismen benötigt und die Aufgabe des Informierens ist schon erfüllt. Bei der Anwendung von Nummernsystemen, die Identifizieren oder Klassifizieren sollen, kann sich die Entschlüsselung als schwierig gestalten und es bedarf Zusatzinformationen über den Aufbau der Nummerung, um alle enthaltenen Auskünfte verstehen zu können.

Bernhardt und Bernhardt teilen die Nummernsysteme in sprechende, halbsprechende und nichtsprechende Teilenummern auf [BeBe-85]. Mit sprechenden Teilenummern sind Bezeichnungen gemeint, deren Bedeutung ohne eine Legende bzw. einen Schlüssel aufgrund ihrer Länge verständlich sind. Halbsprechende Nummern werden auch Verbundnummern genannt und lassen sich teilweise entziffern. In Großunternehmen, die sehr große Datenmengen und somit auch Produkte in ihren Datenbanken speichern, können allerdings nicht alle Artikel anhand ihrer Sachmerkmale oder Spezifikationen unterschieden werden. Um dieses Problem zu lösen, bekommen z.B. nur bestimmte Produktlinien oder Fertigungschargen eine eindeutige Identifikationsnummer. Die Einzelteile auf der untersten Ebene werden bei Bedarf dann noch mit einer Zähleinheit versehen, so dass die Nummer zum Teil sprechend und zum Teil nicht sprechend ist. Wenn die Identifikationsnummer nur aus einer solchen Zähleinheit besteht, wird sie als nichtsprechend bezeichnet.

Ein bekanntes Beispiel für ein umfassendes Nummernsystem ist die ISBN (Internationale Standardbuchnummer). Jedes Buch, das weltweit veröffentlicht wird, wird mit einer eindeutigen ISBN versehen, so dass jede Nummer die entsprechende Veröffentlichung hundertprozentig identifizieren kann und nie doppelt vorkommt. Durch beinhaltete Länder- und Verlagscodes wird automatisch eine weltweite regionale Einteilung unternommen und nur anhand der Nummer kann schon erkannt werden, wo und von welchem Verlag das Buch publiziert wurde. Zur Absicherung ist außerdem eine Prüfziffer enthalten.

2.5 Klassen und Zugehörigkeiten

Um den Ausdruck der Klassifikation erklären zu können, muss als erstes der Begriff der Klasse definiert werden. Eine Klasse ist eine abstrakte Ordnung, in der Objekte bzw. Sachverhalte zu Kategorien gruppiert werden. Grundsätzlich ist zwischen Klassen und Objekten in der Informatik und in der Realität zu unterscheiden. Zuerst soll der Begriff Objekt aus der Mathematik und dem Fachgebiet Informatik erklärt werden. Bezogen auf das Paradigma des objektorientierten Programmierens sind mit den Objekten keine Gegenstände, sondern Programmiereinheiten gekennzeichnet, die durch unterscheidende Eigenschaften voneinander getrennt und in einzelne Kategorien eingeteilt werden können. Im Gegensatz zum prozeduralen Programmieren wird bei der Objektorientierung der Quelltext in einzelne Bestandteile, wie z.B. Klassen und Unterklassen, eingeteilt, in denen Methoden (Funktionen), die funktionelle Quelltextabschnitte darstellen, und Attribute (Eigenschaften) enthalten sind, die zusammen wiederum themenorientierte Bestandteile bilden. Bei der Verwendung dieser Funktionen im Hauptprogramm können dann die Objekte bzw. Funktionspakete separat aufgerufen und beliebig oft benutzt werden. Besonders bei der Umsetzung von großen Projekten ist diese Unterteilung in die einzelnen Bestandteile aus Übersichts- und Wiederverwendungszwecken sehr hilfreich. Zu beachten ist hierbei auch die Möglichkeit der Vererbung von Klasseneigenschaften, was besonders bei der Bildung von Oberklassen von großem Vorteil ist.

Im Bereich der Informatik können die Prinzipien des Klassenaufbaus als Ontologie zusammengefasst werden. Dieser Ausdruck stammt ursprünglich aus der Philosophie und bezeichnet dort entsprechend der Begrifflichkeit der Klassifizierung eine Strukturierung der menschlichen Wahrnehmung der Wirklichkeit. 1993 definierte Gruber Ontologie im Bezug auf die Computerwissenschaften [Grub-93]:

> *An ontology is an explicit specification of a conceptualization.*

Eine Ontologie kann auch eine Darstellungsform von Datenstrukturen sein. Durch Verknüpfungen der einzelnen Bestandteile werden so auch die Beziehungen untereinander dargestellt. Das englische Wort 'conceptualization' ist in diesem Fall mit 'Begriffsbildung' zu übersetzen und bezeichnet die Aufteilung eines Sachverhaltes in die einzelnen Bestandteile. Stuckenschmidt [Stuc-11] erklärt Ontologie im Bezug auf Informatik sinnbildlicher:

> *Um die potentielle Bedeutung von Ontologien in der Informationsverarbeitung zu verstehen, muss man sich klarmachen, was die grundlegenden Probleme der Informationsverarbeitung sind: nämlich bestimmte Ausschnitte der realen Welt in eine geeignete Darstellungsform zu überführen,*

> *die mit Hilfe des Computers manipuliert werden kann, um mit dieser Veränderungen in der realen Welt abzubilden.*

Im Alltag spielt die Einteilung in Klassen ebenfalls eine wichtige Rolle. Aus evolutionärer Sicht liegt es in der Natur des Menschen Gegenstände und Sachverhalte zu klassifizieren und zu kategorisieren. Im Gegensatz zum oben beschriebenen Klassenbegriff in der Informatik ist die Vorgehensweise bei der Klassenbildung meist umgekehrt. Wenn z.B. verschiedene Gegenstände manuell klassifiziert werden sollen, werden die Attribute bzw. die Eigenschaften eines jeden Objektes analysiert und auf Ähnlichkeiten untereinander untersucht. Durch die gefundenen Ähnlichkeiten entstehen Klassengrenzen, die bei der endgültigen Sortierung gezogen werden. Die Klasse entsteht also erst, wenn die Objekte schon untersucht wurden. In der objektorientierten Informatik läuft diese Prozedur gewissermaßen rückwärts ab. Die Klassen und Klassengrenzen müssen zu Beginn im Quelltext genau definiert werden. Die Objekte werden dann aus diesen Klassen und ihren enthaltenen Attributen abgeleitet. Sie entstehen also erst im Nachhinein.

Ein gutes Beispiel für die Klassifikation in der realen Welt stellen politische Systeme und die Staatstheorie mit der Einteilung in Länder, Provinzen, Regionen oder Bezirke dar, die angefangen bei politischen Zusammenschlüssen über Staatszugehörigkeiten bis hin zur Identifikation des einzelnen Individuums mit den jeweiligen Nationalitäten, Ethnien, Kulturkreisen oder sogar Vereinen reichen. Auch hier ist die Idee der Vererbung von Eigenschaften, die für mehrere Klassen gelten nachvollziehbar. Hier soll das Hauptaugenmerk allerdings auf die Aufteilung von Produkten in spezifizierte Kategorien, wie z.B. Fahrzeugklassen, gelegt werden.

Der nächste Schritt nach der Definition von Klassen ist die eigentliche Abgrenzung und Zuordnung des Inhalts. Grabowski, Lossack und Weißkopf [GrLW-02] spalten den Begriff der Klassifikation in zwei Bereiche auf:

> *Die Klassifikation umfasst die Begriffe Klassifizierung (Bildung von Gruppen oder Klassen, Aufbau einer Klassenstruktur) sowie Klassieren (Zuordnung von Objekten in vordefinierte Klassen).*

Bei der Anwendung in der Praxis ist diese Einteilung gut nachvollziehbar, wenn ein derartiges Ordnungssystem aufgebaut werden soll. Als erstes müssen demnach die Merkmale bzw. Attribute einer jeden Klasse definiert werden, so dass erst dann die eigentliche Klassierung - also die Einordnung der Objekte in die Klassen beginnen kann. Es gilt zu beachten, dass mit der hier angegebenen Definition nicht die Klassifikation in der Informatik gemeint ist. Dort gibt es zwar auch vordefinierte Klassengrenzen, allerdings werden dabei die Objekte aus den Klassen erzeugt.

Die hier beschriebenen Klassen sollten so aufgebaut sein, dass die Objekte zu mehreren Gruppen zusammengeschlossen aber nicht durch die Attribute einer Klasse voll-

ständig identifiziert werden. Wenn also mehrere Klassen existieren, die jeweils nur ein Objekt enthalten, dann sollten die Klassengrenzen vielleicht in anderer Art und Weise gezogen werden. Pahl et al. [PBFG-05] definieren bei der Auflistung von Anforderungen an Nummernsysteme genau diesen Sachverhalt. Sie unterscheiden deutlich zwischen Identifizieren und Klassifizieren:

> *- Identifizieren, d.h. eindeutiges und unverwechselbares Erkennen eines Gegenstandes anhand von Merkmalen ermöglichen.*
>
> *- Klassifizieren, d.h. Ordnen von Sachen und Sachverhalten nach festgelegten Merkmalen ermöglichen. Eine Klassifizierung ist nur eine Beschreibung ausgewählter Eigenschaften. Dieselbe Klassifikations-Nr. stellt also die Gleichheit von Gegenständen in Bezug auf diese Eigenschaften fest, nicht aber eine Identität.*

Alle Klassen eines Systems sollten so angelegt werden, dass kein Objekt durch die Attribute einer Klasse vollständig identifiziert wird. Aber auch das genaue Gegenteil - nämlich dass ein Objekt keiner Klasse zugeordnet werden kann - sollte vermieden werden. Die Schwierigkeit bei der Definition der Klassen und der Festlegung der Attribute besteht also darin schon zu Beginn die Eigenschaften aller Objekte zu überblicken und dementsprechend die Klassengrenzen zu ziehen. Wenn ein funktionell gegliedertes System, wie z.B. eine Datenbank, geschaffen werden soll, ist die Aufteilung zu Beginn gemäß der späteren Funktion sehr wichtig. Vossen definiert einen solchen zweckmäßigen Datenbankentwurf folgendermaßen [Voss-08]:

> *Die Aufgabe des Datenbankentwurfs ist der Entwurf der logischen und physischen Struktur einer Datenbank so, dass die Informationsbedürfnisse der Benutzer in einer Organisation für bestimmte Anwendungen adäquat befriedigt werden können.*

Bei der Klassifikation und den Darstellungsformen dieser Klassifikation sind die Objekte aufgrund von nicht eindeutigen Klassengrenzen nicht immer exakt zuordenbar. Wenn die Klassengrenzen ineinander übergehen oder nicht genügend Merkmale für eine eindeutige Zuordnung getroffen werden können, kann die Fuzzylogik bei der Entscheidungsfindung hilfreich sein. Im Gegensatz zur konventionellen Logik, die wie in einem binären System nur zwei Zustände unterscheidet (z.B. Signal oder kein Signal), wird bei der Anwendung der Fuzzylogik zwischen mehreren Zuständen unterschieden (z.B. heiß, warm, kühl und kalt). Mit dieser Vorgehensweise können die Objekte somit eindeutiger differenziert und ihren entsprechenden Klassen zugeordnet werden (vgl. Kapitel 6.2).

2.6 Produktklassifikation

Die Grundlagen der Klassifikationstheorie können auf die Kategorisierung von Produkten, wie es z.B. in der industriellen Anwendung geschieht, übertragen werden und sollen an dieser Stelle aufgrund der Klassifikationsansätze in Kapitel 6.2 eingehender beschrieben werden. Um die Systematik der Einteilung in Kategorien zu veranschaulichen, sollen als Beispiel die Grundlagen des elektronischen Handels dienen. Schon im normalen Lebensalltag fällt auf, dass alle Arten von käuflichen Produkten z.B. im Supermarkt oder auf Einkaufsplattformen im Internet in Abteilungen bzw. Produktkategorien eingeteilt sind. Dies hat aus nachvollziehbaren Gründen den Zweck der einfacheren und schnelleren Auffindbarkeit von gesuchten Artikeln. Die Aufteilung von Ladenlokalen in verschiedene Abteilungen hat außerdem Vorteile für die Ausbildung des Personals und die Einrichtung der benötigten Bedarfsvoraussetzungen, wie z.B. Kühlhäuser.

Auch in großen Handelshäusern oder bei Zwischenhändlern stellt sich die Frage nach welchen Kriterien die Waren kategorisiert werden sollen. Es existieren verschiedene Ansätze, wie z.B. die Aufteilung nach Preis, Funktion, Hersteller oder Erscheinungsdatum. Je nach den Eigenschaften eines Produktes kann es vorkommen, dass eine eindeutige Kategorisierung nicht möglich ist, da der Artikel mehrere Funktionen hat oder die Klassen- bzw. Abteilungsgrenzen nicht eindeutig definiert sind. Als Beispiel könnte etwa eine Eieruhr genannt werden, die entweder zum Haushaltsbedarf oder eben in die Uhrenabteilung klassifiziert werden könnte. Ein weiteres anschauliches Beispiel im Bezug auf die Automobilindustrie ist die Neuentstehung der Fahrzeugklasse der SUVs (Sports Utility Vehicle). Diese kombiniert die Eigenschaften von Geländewagen mit dem Fahrkomfort für den Stadtverkehr und dem Laderaum eines Vans.

Für eine einfache Durchführbarkeit des elektronischen Handels wurden sogenannte Produktstandards entwickelt, die auf verschiedene Berufsbranchen spezialisiert sind und alle verfügbaren Artikel enthalten. Auch hier werden verschiedene Kriterien, wie z.B. Funktion oder Fähigkeit, für eine effektive Sortierung angewendet. Im Bereich des Maschinenbaus handelt es sich hierbei um einen Produktstandard namens 'eCl@ss' [Ecla-15]. Ebenso wie die Darstellung von Hierarchieebenen mit Hilfe von Bäumen aus der mathematischen Graphentheorie (vgl. Kapitel 3.9), sind die Produkte, die im System enthalten sind, erst in verschiedene Untergruppen und Gruppen aufgeteilt, die auf den höheren Ebenen zu Klassen zusammengefasst und so nach immer gröberen Merkmalen sortiert werden. Sachmerkmalsleisten (vgl. Kapitel 2.4), in denen die Eigenschaften eines jeden Produktes stichpunktartig in Tabellenform aufgeführt sind, helfen bei der Feineinteilung innerhalb der untersten Ebenen.

Weitere Produktstandards sind z.B. ETIM (Elektrotechnisches Informationsmodell) für die Kategorisierung von elektrotechnischen Waren oder UNSPSC (United Nations Standard Products and Services Code), das ein branchenübergreifendes Warengruppensystem bildet. Es gilt zu beachten, dass sowohl in eCl@ss als auch in UNSPSC nicht nur Waren und Artikel, sondern auch Dienstleistungen beinhaltet sind und bei der Klassifikation berücksichtigt werden.

Ähnlich der in Kapitel 2.3 beschriebenen Datenbanksysteme, wird besonders für die Verwaltung von großen Datenmengen, die eines Inhaltüberblickes bedürfen, das Konzept des sogenannten 'Data-Warehouse' angewendet. Kemper und Eickler definieren den Begriff umfassend [KeEi-06]:

> *Darunter versteht man ein dediziertes Datenbanksystem, in dem die für Decision-Support-Anwendungen notwendigen Daten eines Unternehmens in konsolidierter Form gesammelt werden.*

Mit Decision-Support sind hier Softwaresysteme gemeint, die den Anwender bei der Entscheidungsfindung unterstützen, indem sie beispielsweise Zusatzinformationen bereitstellen oder visualisieren.

Der Begriff Operational Data Store gleicht dem des Data-Warehouse, allerdings dürfen in einem solchen System die Daten nicht nur eingesehen, sondern auch verändert werden. Schütte et al. beschreiben dies so [ScRH-01]:

> *In einem Operational Data Store werden operative Daten sehr zeitnah zusammengeführt, so dass eine neue Architekturschicht entsteht, deren Daten an Informationsobjekten orientiert, aktuell, änderbar, detailliert, integriert und vor allem in Echtzeit zugänglich sind.*

Beide Ansätze werden besonders in der Warenwirtschaft und dem oben beschriebenen elektronischen Handel angewendet, da eine schnelle Aktualisierung der Bestände und der Änderungen wichtig ist. Besonders Onlineshops greifen faktisch in Echtzeit auf die ihnen zugrunde liegenden Datenbanken zu und zeigen dem Benutzer z.B. die Anzahl der Verpackungseinheiten, die sich noch im Lager befinden, an.

2.7 Methoden zur Klassifikation

In diesem Kapitel gilt es die Verbindung zwischen der klassischen Konstruktionssystematik und der Klassifikation von den dabei entstehenden Produkten herzustellen. Im Hinblick auf die Eigenentwicklungen im zweiten Teil der Arbeit ist dies ein großer Bestandteil in der ingenieurstechnischen Entwicklung sowie der computergestützten Verarbeitung der anfallenden Daten. Zur Ideenfindung beim Entwerfen von neuen Produkten bietet sich die Möglichkeit eines morphologischen Kastens bzw. einer morphologischen Matrix an. Darunter wird eine Arbeitsweise verstanden, bei der alle möglichen Entwürfe einer bestimmten Kategorie in einer Art Tabelle aufgelistet werden, so dass sie bei der späteren Bewertung in übersichtlicher Form gestaltet sind und einfach kombiniert werden können.

Pepels definiert einen morphologischen Kasten wie folgt [Pepe-06]:

> *Die Morphologie ist die Aufgliederung eines Problems hinsichtlich aller Parameter und die Suche nach neuen Kombinationen vorhandener Teillösungen [...]. Das Problem wird dabei in seine Bestandteile zerlegt, die grafisch in einem Kasten untereinander angeordnet werden. Neben jedes Problemelement werden dann möglichst viele Lösungsmöglichkeiten geschrieben, deren Kombination verschiedene Lösungen des Gesamtproblems ergibt.*

Hieraus wird ersichtlich, dass eine gewisse Klassifikation des Produktes schon durchgeführt wird bzw. im morphologischen Kasten ersichtlich ist. Wenn in einem Unternehmen schon die Ideenfindung einer solchen Methodik gut dokumentiert wird, dann könnten diese Unterlagen auch später zur Bestimmung der Produktklassen dienen. Auch können nicht nur die technischen, sondern auch die funktionellen oder formgebenden Spezifikationen enthalten sein, was weiterhin für die Bewertung und spätere Einteilung anhand unterschiedlicher Gesichtspunkte wichtig sein kann.

An dieser Stelle soll noch einmal der Begriff der Features aufgegriffen werden. So wie ein Bauteil beim virtuellen Aufbau im CAD-Programm in verschiedene Bereiche eingeteilt wird, könnte auch der morphologische Kasten in jene Kategorien gegliedert werden. Eine solche Einteilung in herstellungsspezifische oder funktionelle Bereiche eines Bauteiles könnte sich also durch die gesamte Entwicklung und Produktion eines Produktes ziehen, wenn die Features schon beim Entwurf im morphologischen Kasten untergebracht sind, das Bauteil dann bei der virtuellen Konstruktion im CAD-Programm in die Features unterteilt wird, die die einzelnen Verfahrensschritte der Maschinen in der Produktion darstellen, und am Ende für die datenbanktechnische Verwaltung benutzt werden.

Eine ähnliche Vorgehensweise zur Lösung von Konstruktionsproblemen ist das sogenannte TRIZ. Dies ist eine russische Abkürzung und kann laut Orloff [Orlo-06] im Deutschen mit 'Theorie des erfinderischen Problemlösens' übersetzt werden. Den Kern dieser Technik bilden 40 Lösungsmöglichkeiten für technische Probleme, die bei der Entwicklung der Methode anhand der Analyse vieler verschiedener ingenieurstechnischer Patente extrahiert wurden und somit im Bezug auf die Produktklassifikation auch als Anhaltspunkt für die Einteilung in Kategorien dienen können.

Eine weitere interessante Technik, die sich mit der Zerlegung von Produkten und einer folgenden Untersuchung ihrer Teilbereiche beschäftigt ist die FMEA (Failure Mode and Effects Analysis). Die deutsche Übersetzung lautet nach Tietjen und Müller 'Fehler-Möglichkeits- und Einflussanalyse' [TiMü-03]. Die Ergebnisse einer solchen Separation von Produkten vor der eigentlichen Produktion können ebenfalls - ähnlich wie bei einer morphologischen Matrix oder bei TRIZ - für die Produktklassifikation von großem Nutzen sein.

2.8 PDM-Software zur Bauteilsuche und Klassifikation

Bei den in diesem Kapitel vorgestellten Programmen handelt es sich um professionelle PDM- bzw. ERP-Systeme (Enterprise Ressource Planning), die dafür benutzt werden große Datenbestände zu verwalten. Neben einer Angabe von Informationen, die mit jedem Eintrag in der Datenbasis verknüpft sind, haben alle Programme das Ziel die Suche von Dateien und insbesondere von Bauteilen und Baugruppen zu beschleunigen. Ein wichtiger Aspekt bei der Auffindung ist die sogenannte Ähnlichkeitssuche, bei der ein Referenzteil vom Benutzer vorgegeben wird und alle in der Datenbasis befindlichen Bauteile mit dieser Referenz verglichen werden. Sollte eine Übereinstimmung bestehen, so wird das aktuelle Teil als Treffer gewertet und dem Anwender in den Ergebnissen - meist verbunden mit einem Ähnlichkeitsfaktor - angezeigt.

Wie schon in Kapitel 2.3 beschrieben, werden Bauteile in den industriellen CAE-Systemen für gewöhnlich aus einzelnen Features aufgebaut, deren Spezifikationen im Strukturbaum chronologisch abgespeichert werden und von dort eine Fülle an Informationen liefern. Die hier erwähnten Programme werten neben der geometrischen Erscheinung insbesondere diese Spezifikationen aus und benutzen sie für die Klassifikation nach den zuvor definierten Benutzervorgaben. Allen Programmen ist also gemein, dass sie von jeder in der Datenbasis befindlichen Datei so viele Informationen wie möglich sammeln, um den Anwendern den Zugang zu vereinfachen und die Suche zu beschleunigen. Diese gesammelten Informationen werden in Form einer Metadatenbank abgespeichert, von wo aus sie bei eventuellen Suchanfragen innerhalb kürzester Zeit wieder abgerufen werden können. Ein ähnliches Prinzip wird auch bei Internetsuchmaschinen angewendet (vgl. Kapitel 2.9.1). Auch dort wird bei einer Suchanfrage nicht das gesamte Internet durchsucht, sondern hauptsächlich auf die Einträge der Metadatenbank zugegriffen, auf der lediglich die Schlüsselinformationen liegen.

Im Zuge dieser Arbeit sind zwei studentische Projekte entstanden, die die Analyse und die Evaluation von Similia und Exalead zum Schwerpunkt haben. Özmen untersucht die Ergebnisse, die Exalead speziell bei der Ähnlichkeitssuche anzeigt und evaluiert die Unterschiede der Resultate bei einfachen, durchschnittlichen und komplexen Geometrien [*Özm-15]. Des Weiteren analysiert er den Strukturbaum mit Hilfe des in Kapitel 4.4 vorgestellten Programmes und vergleicht die Ergebnisse mit denen von Exalead.

Wagner konzentriert sich auf die Auswertung von Similia und Exalead und untersucht außerdem herstellerunabhängige Austauschformate [*Wag-15].

2.8.1 Similia

Similia ist ein PDM-System des Herstellers Simuform und umfasst eine Vielzahl an Modulen, die sich hauptsächlich auf den Bauteilvergleich, aber auch auf andere Anwendungen, wie z.B. Server Management, Teileverwaltung oder Integration von neuen Dateien ins Gesamtverzeichnis, konzentrieren [Simu-14]. Das vollständige System ist in Form einer SQL-Datenbank (Structured Query Language) aufgebaut, was eine schnelle Bearbeitung ermöglicht und den Vorteil der lizenzfreien Verbreitung bietet.

Von besonderem Interesse für die Thematiken im zweiten Teil dieser Arbeit sind die Module 'Part Overlay', 'Contour Search', 'Segment Search' und 'Dublicate Search'. Das 'Part Overlay' bietet die Möglichkeit zwei Bauteile oder Baugruppen ähnlich wie bei einer booleschen Operation übereinander zu platzieren, um so schnellstmöglich eventuelle Differenzen identifizieren zu können. Ähnlich verhält es sich beim 'Contour Search'. Dieses Modul wurde auf die Anforderungen der Fertigung zugeschnitten und erfüllt den Zweck ein passendes Rohteil bzw. Halbzeug zur Weiterverarbeitung und Fertigstellung zu finden, das die nötigen Schritte und den Arbeitsaufwand bei der Herstellung minimiert.

Das Modul 'Segment Search' erweitert die Möglichkeiten der normalen Suche nach Bauteilen und ganzen Baugruppen, die in den meisten PDM-Systemen standardmäßig gegeben sind. Es ermöglicht es dem Benutzer einen bestimmten Bereich eines Bauteiles auszuwählen und nur nach dieser Substruktur bzw. nach diesem Teilbereich im gesamten CAD-Datenbestand zu suchen. Dabei ist es denkbar, dass ein Bereich eines Produktes in einem anderen noch einmal ähnlich oder genau so vorkommt, so dass dies bemerkt wird und von den Anwendern nicht neu ausgearbeitet werden muss.

Das 'Dublicate Search' wird insbesondere bei großen Unternehmen benötigt, die ihre Datenbasis aufräumen bzw. bereinigen wollen. Mit Hilfe dieses Moduls werden doppelte Einträge gefunden und können dann entfernt oder aufbereitet werden, so dass am Ende ein reduzierter Datenpool entsteht, der dennoch keine Informationsverluste zu verzeichnen hat.

2.8.2 Exalead One Part

Die Kurzbezeichnung für dieses PDM-Programm ist 'Exalead' und wurde ebenso wie CATIA von der Firma Dassault Systèmes entwickelt [Dass-15]. Der Aufbau der Nutzeroberfläche gestaltet sich ähnlich dem einer Internetsuchmaschine. In einem Textfeld kann der Anwender Begriffe oder verallgemeinerte Informationen eingeben und ggf. auswählen nach welchen Dateitypen gesucht werden soll. Nachdem ein Datensatz für die Verarbeitung in Exalead indiziert wurde, sind alle Datentypen, die sich im Gesamtverzeichnis befinden, auch in der Suchleiste auswählbar. Dabei spielt es - so wie bei den anderen Programmen auch - keine Rolle von welchem Hersteller die Daten stammen bzw. welche Dateitypen untersucht werden sollen.

Im Ergebnisfenster werden nach Eingabe der Suchbegriffe alle Dateien sortiert nach der Relevanz auflistet. Wenn die Resultate nicht zufriedenstellend sein sollten oder auf Grund zu vieler Dateien immer noch zu unübersichtlich sind, dann kann die Suche durch eine Präzisierung der Parameter spezifiziert werden. Neben textlichen Schlagwörtern werden dabei Zusatzinformation, wie z.B. Erstellungs- und Änderungsdatum, Bauteilfarben, Featuregehalt, Abmessungen, Material oder weitere physikalische Eigenschaften, aus allen Dateien im Index automatisch herausgelesen und können bei der Suche einbezogen werden.

Beim direkten Formenvergleich zweier Bauteile kann auch hier eine Referenz ausgewählt werden, die mit ähnlichen Teilen verglichen wird und deren eventuellen Gleichwertigkeiten dem Nutzer in einem einzigen prozentualen Zahlenwert (100% Übereinstimmung bei identischen Teilen) angezeigt werden.

2.8.3 Simus Classmate

Die Fähigkeiten der Software Simus Classmate tragen laut der Entwicklerfirma Lino zum DPM (Daten Prozess Management) in Unternehmen mit großen Datenbeständen bei [Lino-15]. Damit ist eine ganzheitliche Betrachtung aller Daten eines Produktes gemeint, was unter anderem auch die Qualitätssicherung oder die Klassifikation des Produktes mit einschließt.

Das Programm ist aus den folgenden vier Paketen zusammengesetzt, die alle spezielle Anwendungsfälle umfassen: 'Simus Classmate-CAD', 'Simus Classmate-Data', 'Simus Classmate-Finder' und 'Simus Classmate-Plan'. Neben dem CAD-Modul, das den Kern bildet und sowohl für die Analyse, als auch für die Klassifikation von Bauteilen und Baugruppen verantwortlich ist, stellt das Data-Modul das Pendant zum 'Dublicate Search' von Similia dar. Der 'Simus Classmate-Finder' kann als Suchmaschine umschrieben werden und analysiert außerdem die in den Bauteilen enthaltenen Features. Ein Alleinstellungsmerkmal von Simus Classmate ist das Plan-Modul, das eine Art Projektmanagement ermöglicht und an der Erstellung von Arbeitsplänen und Kalkulationen beteiligt ist. Eine weitere Besonderheit von Simus Classmate ist die Synchronisationsmöglichkeit mit SAP-Verwaltungssystemen.

2.8.4 Geolus Search

Das Programm Geolus Search wurde so wie NX von der Firma Siemens PLM entwickelt und ähnelt sich durch die Anwendung in einem Internetbrowser stark mit Exalead One Part [Siem-15]. In einer Suchleiste können Begriffe eingegeben werden, die ggf. in der gesamten Datenbasis vorhanden sind und alle Dateien, die einen Treffer ergeben haben, werden dem Benutzer dann angezeigt. Ebenso wie bei Exalead, kann bei der geometrischen Ähnlichkeitssuche ein Toleranzbereich definiert werden, der bei der Suche alle Teile automatisch herausfiltert, die nicht in diese Toleranzgrenzen passen.

Es ist jedoch zu erwähnen, dass Geolus Search stark mit der Produktpalette von Siemens PLM verbunden ist, zu der z.B. auch die CAE-Software Solid Edge gehört.

2.8.5 Cadenas

Das Programm Cadenas ist ein Softwarepaket, das aus den drei Bestandteilen PARTsolutions, PURCHINEERING und GEOsearch zusammensetzbar ist [Cade-15]. Die gleichnamige Entwicklerfirma Cadenas stellt mit diesen drei Paketen strategisches Teilemanagement, eine Verbindung zwischen Einkauf und Engineering sowie eine Suchmaschine, die ggf. mehrere Standorte miteinander vereinen kann, zur Verfügung.

Weiterhin entwickelt das Unternehmen auch speziell zugeschnittene Software nach Kundenaufträgen.

2.9 Datenorganisation in Großunternehmen

In diesem Kapitel soll das Management und die Strategien im Umgang mit großen Datenmengen am Beispiel von drei Großkonzernen dargelegt werden. Aufgrund der unterschiedlichen Anforderungen soll verdeutlicht werden, dass jede Situation im Umgang mit großen Datenmengen Unterschiede aufweist und eines speziell angepassten Konzeptes bedarf.

2.9.1 Google

Die Systematik, die das Unternehmen Google Inc. bei der Datenverwaltung verfolgt, veranschaulicht sowohl die in diesem Kapitel beschriebenen Möglichkeiten der Klassifikation als auch den Aufbau von und den Umgang mit Datenbanken. Zu Anfang soll auf die Mechanismen eingegangen werden, die intern bei jeder Suchanfrage, die von Internetnutzern auf der ganzen Welt gestartet werden, automatisch ablaufen. Doch es ist anzumerken, dass die genauen Algorithmen ein von Google gut gehütetes Geschäftsgeheimnis sind und der Öffentlichkeit und somit der Konkurrenz nicht zugänglich gemacht werden. Dennoch ist über die Funktionalität und den Aufbau der Rechnersysteme einiges bekannt.

Jede der vielen Suchanfragen, die jede Sekunde aus aller Welt gestartet werden, wird an die von Google betriebenen Rechenzentren weitergeleitet und dort von vielen Computern parallel bearbeitet. In diesen Rechenzentren, von denen es mindestens 13 Stück gibt, durchsuchen Computer, die zu Clustern und größeren Einheiten zusammengefasst sind, die Datenbasis von Google [Grei-06]. Bei einer Suchanfrage wird nicht etwa in Sekundenbruchteilen das gesamte Internet durchsucht, sondern nur dieser interne Datenpool. Im Jahr 2009 gab das Unternehmen an, dass sie mehr als eine Billionen Indices und somit Webseiten enthalte, auf die bei der Suche zugegriffen werden kann.

Dieser Index wurde jedoch im Laufe der Zeit wirklich durch die gesamte Durchsuchung des Internets gefüllt. Laut Hill erfolgt eine solche ca. ein Mal pro Monat und dauert länger als eine Woche [Hill-06]. Unternehmensintern wird dieser Prozess 'Deep Crawl' genannt und ist von dem sogenannten 'Fresh Crawl' zu unterscheiden. Der 'Fresh Crawl' wird ungefähr ein Mal am Tag durchgeführt und untersucht insbesondere die am häufigsten aktualisierten Webseiten, um die neuesten Inhalte, wie z.B. Nachrichten, für jede Suche relevant zu machen.

Die Geschwindigkeit, mit der die Ergebnisse für eine jede Anfrage geliefert werden können, ist hauptsächlich der parallelen Verarbeitung zu verdanken. Zum einen benutzt Google in den Rechenzentren sehr günstige Standardcomputer, die bei Fehlfunktion schnell ersetzt werden können, und zum anderen werden bei jeder Anfrage so viele Computer beteiligt, dass eventuelle Ausfälle einzelner Geräte oder ganzer Cluster nicht weiter ins Gewicht fallen.

Verglichen mit Kapitel 2.3 können alle beteiligten Computer in den Rechenzentren als eine große Datenbankkommunikationsschnittstelle angesehen werden, die dem Benutzer eine 'read-only'-Kopie ausgibt. Das Datenbankverwaltungssystem ist mit den oben genannten Crawlern zu vergleichen, die befugt sind die Datenbasis zu verändern und zu aktualisieren.

Für eine Aufnahme der gefundenen Informationen in den Index der Datenbank müssen laut Erlhofer die Daten jedoch zuerst durch Datennormalisierung, Datenanalyse und Generierung einer durchsuchbaren Datenstruktur (Index) aufbereitet werden [Erlh-11]. Der Index ist mit dem Stichwortverzeichnis eines Buches zu vergleichen. Eine Suchanfrage überprüft den Google-Index und öffnet bei einer Übereinstimmung dessen Inhalt, der auf die gesuchten Webseiten verweist, und zeigt die Ergebnisse dem Benutzer an. Bezüglich einer Klassifikation erfolgt hier keine Gruppierung in bestimmte Kategorien, sondern die Datenbasis folgt lediglich einer alphabetischen Ordnung.

Die hier erläuterten Prinzipien, die das Unternehmen Google durch den speziellen Aufbau und der Benutzung verfolgt, zeigen, dass eine aufwendige Klassifikation in Kategorien aufgrund der schnellen Rechenleistung nicht zwangsläufig nötig ist. Durch die Verwendung so vieler parallel geschalteter Rechner, kann die Datenbank nach nur einem Gesichtspunkt, nämlich einer alphabetischen Sortierung, gegliedert sein und der Inhalt ist dennoch in Sekundenschnelle abrufbereit.

2.9.2 Carl Zeiss

Die Ausführungen in diesem Kapitel, die vor allem die Anwendung von Nummernsystemen in der Praxis schildern sollen, beziehen sich auf die Unternehmensrichtlinien der Carl Zeiss AG für die Erstellung von Dateien mit der Software Pro/ENGINEER [Carl-10]. Alle Entwickler und Konstrukteure, die bei ihrer Arbeit CAD-Dateien erstellen, müssen sich an diese Richtlinien halten, damit die Daten später vollautomatisch in die Datenbank, die von SAP PLM verwaltet wird, eingegliedert werden können (Seite 1).

Damit dies reibungslos möglich ist, werden schon zu Beginn der virtuellen Entwicklung eines neuen Bauteiles in der 3D-Umgebung die Dateien nicht neu erstellt, sondern es müssen bestimmte Startmodelle verwendet werden (Seite 2). In diesen sind etwa die Längen- und Masseeinheiten schon voreingestellt.

Des Weiteren existiert eine Normteildatenbank, deren Inhalt z.B. bei dem Zusammenbau von Teilen zu Baugruppen verwendet werden muss. Auch die Erstellung von zweidimensionalen technischen Zeichnungen ist bezüglich des Zeichnungsrahmens, der Beschriftungen, der Bemaßung und der Toleranzen streng reguliert (Seite 3).

Wenn eine Datei fertig erstellt wurde, muss ihr ein Schlüssel zugewiesen werden. Dieser besteht aus einer Nummer, die die Datei, wenn sie später in die Datenbank aufgenommen wird, eindeutig identifiziert. Ein Teil dieses Schlüssels bildet dann auch den Namen der Datei. Die Nummer kann in die folgenden vier Bestandteile unterteilt werden (Seite 5):

1. *Dokumentennummer (Materialnummer/lfd.Nr. des Dokumentes)*
2. *Dokumentenart*
3. *Nummer des Teildokumentes*
4. *Version*

Zu Beginn steht die Dokumentennummer, die wiederum in Materialnummer und laufende Nummer (von 01 hochgezählt) separiert werden kann, welche durch einen Schrägstrich voneinander getrennt werden. Der zweite Bestandteil der Gesamtnummer ist die Dokumentenart, die entweder als FUM (Fertigungsunterlage materialabhängig) für zweidimensionale Zeichnungen oder FU3 (Fertigungsunterlage materialabhängig (3D)) für 3D-Modelle ausgeführt wird. Als nächstes folgt die dreistellige Nummer des Teildokumentes. Anhand dieser kann der Teilestamm der Datei nachvollzogen werden. Beispielsweise erhalten Teilefamilien, Katalogteile, Hilfsmodelle, Normteile oder Zubehörteile jeweils eine eigene Teildokumentennummer (Seiten 8-9). Den Abschluss der Gesamtnummer bildet die Version der Datei. Somit kann die Erstellungschronologie auch ohne genaue Datumsangaben nachvollzogen werden.

Zu dem Schlüssel einer jeden Datei kommen außerdem sogenannte Metadaten im PLM-System hinzu, die für die Einsortierung in die Datenbasis wichtig sind und beispielsweise Informationen wie Zeichnungsnummer, Bearbeiter oder Material enthalten (Seite 6).

Anhand dieser konstruktiven Vorgaben wird ersichtlich wie wichtig eine gute Vorbereitung zur Einspeisung von Informationen in die Datenbasis für ein übersichtliches strukturiertes Datenbanksystem ist.

Nur durch derartige Konventionen kann eine Datenkonsistenz erhalten und die eigentlichen Zwecke einer Datenbank, nämlich die adäquate Abspeicherung und eine schnelle Abrufbereitschaft, bewahrt werden.

2.9.3 Mercedes-Benz

Dieses Kapitel bezieht sich auf das Produktionssystem von teilautonomen Fertigungsinseln der Daimler AG. Um die notwendige Klassifikation, auf die in Kapitel 2.6 eingegangen wurde, zu verdeutlichen, werden hier insbesondere die Mechanismen, die bei der Produktionsplanung umgesetzt werden, dargestellt. So wird unterstrichen, dass eine modulare Betrachtung der einzelnen Fahrzeugbestandteile bzw. eine Voreinteilung, die bereits während der Planungsphase entsteht, auch für die spätere Fabrikation von großem Vorteil ist.

Das grundlegende Konzept ist eine Abkehr der Fließbandmontage hin zu modularen Fertigungsgruppen, in denen die Mitarbeiter auf bestimmte Teilgebiete der Montage bzw. auf Teilbausätze des fertigen Fahrzeuges spezialisiert sind. Hier wird deutlich, dass die gesamte Auslegung der Produktion in diese Klassen erfolgen muss und ebenfalls die Ausbildung der Arbeiter auf die diversen Funktionssysteme der Gesamtprodukte gewichtet werden sollte. Ausgehend von dem Produktionssystem der Volvo Group in der Fabrik Uddevalla, in dem die Mitarbeiter bis zu 16 Monaten eingearbeitet wurden und vereinzelt bezüglich der Montage gesamte Automobile alleine zusammenbauen konnten, setzte Mercedes-Benz eine Mischlösung der sogenannten teilautonomen Fertigungsinseln um [Ulic-05].

Zölch, Weber und Leder bezeichnen dieses System als Boxen-Fertigung [ZöWe-99]. Charakteristisch ist neben der Montage von Unterbaugruppen auch die Eigenorganisation, bei der die Mitarbeiter sich neben einem reibungslosen Prozessablauf auch um die Einarbeitung neuer Mitarbeiter, die Fristensetzung und das Qualitätsmanagement kümmern müssen. Schumann und Gerst gelangen in einer Studie zum Thema Arbeitspolitik am Fallbeispiel der zwei Montagegruppen 'Motorenmontage' und 'Endmontage Inneneinbau' zu dem Ergebnis, dass die Arbeit in kleinen Gruppen zur Motivation und zum Zusammenhalt beiträgt [ScGe-96].

Clarke konzentriert sich auf die Standardisierung der Prozessabläufe des Mercedes-Benz Produktionssystemes und gelangt in einer Studie z.B. auf das Ergebnis, dass das Produktionssystem die Kooperation innerhalb und zwischen den einzelnen Produktionsteams stärkt [Clar-05]. Es bleibt zu erwähnen, dass eine Klassifikation in Teilbereiche nicht nur für die Konstruktion von Vorteil sein, sondern bestimmte Fertigungsstrategien ebenfalls positiv beeinflussen kann.

Kapitel 3. Stand der Forschung

3.1 Knowledge Based Engineering

Während in Kapitel 2 der Schwerpunkt auf die industrielle Anwendung und grundlegende Strukturierung von Daten sowie deren Klassifikation gelegt wurde, konzentriert sich dieses Kapitel auf die zu Grunde liegenden wissenschaftlichen Ansätze zu allen behandelten Themen.

Die Herangehensweisen, die in diesem dritten Kapitel sowie in den Eigenentwicklungen im zweiten Teil der Arbeit beschrieben werden, können zu den beiden Oberbegriffen 'wissensbasierte Systeme' bzw. KBE (Knowledge Based Engineering) zusammengefasst werden. Kurbel unterscheidet wissensbasierte von konventioneller Software im Bezug auf die Aufteilung von Wissen und definiert wissensbasierte Systeme wie folgt [Kurb-92]:

> *Ein wissensbasiertes System ist ein Softwaresystem, bei dem das Fachwissen über ein Anwendungsgebiet ("Domain Knowledge") explizit und unabhängig vom allgemeinen Problemlösungswissen dargestellt wird.*

Er unterteilt ein solches System in eine Wissensbasis, in der das Fachwissen enthalten ist, und in eine Problemlösungskomponente, die für bestimmte Problemfälle die spezifischen Lösungsmöglichkeiten bereithält. In einem konventionellen Softwaresystem hingegen sind beide Bestandteile fest miteinander verbunden und können nur durch den Programmierer, nicht aber durch den Anwender erweitert werden.

Im Folgenden werden die Funktionsweisen sowie einige Anwendungsmöglichkeiten des KBE vorgestellt. Im Deutschen wird der Begriff meist mit 'Wissensbasierte Konstruktion' übersetzt, was genau genommen die allumfassende Bezeichnung der Ingenieurstätigkeiten nicht ganz wiedergibt. Cooper und LaRocca definieren KBE folgendermaßen [CoLa-07]:

> *KBE can [...] be defined as the use of dedicated software language tools (i.e. KBE systems) in order to capture and reuse product and process engineering knowledge in a convenient and maintainable fashion.*

Des Weiteren legen sie dar, dass das Ziel bzw. der Effekt von KBE eine Kostensenkung und Zeiteinsparung bei der Produktentwicklung ist, was insbesondere durch die Automatisierung von sich wiederholenden, nicht-kreativen Gestaltungaufgaben und der multidisziplinären Integration in die Konzept- und Entwicklungsphase erreicht werden kann.

Grob gesagt sollen durch die Entwicklung von Software-Werkzeugen, die die Tätigkeiten der Anwender vereinfachen, ganze Prozesse beschleunigt werden. Aufgaben,

die routinemäßig ablaufen, sollen durch die Speicherung und Wiederverwendung von Expertenwissen automatisiert werden, so dass die Anwender nicht zwangsläufig Experten sein müssen, um das Programm bedienen zu können.

Bei der Entwicklung von KBE-Software bzw. von wissensbasierten Systemen muss zu Beginn immer die Frage geklärt werden, was unter Expertenwissen zu verstehen ist und welches Wissen integriert werden soll. Aus nachvollziehbaren Gründen müssen abhängig von dem zu entwickelnden Programm Schwerpunkte gesetzt und Wissensbereiche, die beinhaltet sein sollen, definiert werden. Außerdem muss bestimmt werden, welche Fähigkeiten die Software besitzen soll und wie diese erreicht werden können. Aus diesem Grund soll hier als erstes auf die verschiedenen Wissensarten eingegangen werden.

Es existiert eine Vielzahl von Definitionen des Begriffes Wissen, die sich nach Fachgebiet bzw. wissenschaftlicher Disziplin unterscheiden, aber nie ganz eindeutig sind. Pocsai erklärt beispielsweise die Begriffe Daten, Informationen und Wissen nach Rude anhand ihrer Verwendung in den Bereichen allgemeines Wissen, Semiotik, Betriebswirtschaftslehre, Technik und künstliche Intelligenz [Pocs-00]. Es wird deutlich, dass für die Verwendung von Daten und Informationen Wissen notwendig ist, da ein reines Vorhandensein bei der Anwendung der Daten oder Informationen nicht ausreicht. Dafür ist Wissen nötig, das für die Verarbeitung bzw. Benutzung der Daten sorgt.

Weiterhin listet Pocsai nach Gammack, Welbank und Rude eine Vielzahl von Wissensarten auf, von denen nur einige hier erörtert werden sollen:

Er beginnt mit der Unterscheidung zwischen deklarativen (expliziten) und prozeduralen (impliziten) Wissen. Deklaratives Wissen bezeichnet er auch als Beschreibungswissen, womit Eigenschaften erklärt werden können. Prozedurales Wissen hingegen ist für die Folge von Prozessen notwendig und kann die Vorgehensweise von Abläufen beinhalten. Weiterhin wird zwischen terminologischem Wissen (Fachbegriffe), Faktenwissen (Wahrheiten), kausalem Wissen (logische Schlussfolgerungen), typologischem Wissen (Klassifikation), Regelwissen (Herleitungen) und kontextabhängigem Wissen (Situationen) unterschieden.

Eine weitere Wissenskategorie ist das unbestimmte Wissen, das meist unvollständig ist, aber dennoch bei der Entscheidungsfindung hilfreich sein kann. Laut Pocsai nach Rude und Specht gehören heuristisches Wissen (Empirik), vages Wissen (Indizien), temporales Wissen (vorübergehend), unsicheres Wissen (Wahrscheinlichkeiten), ungenaues Wissen (Oberbegriffe), unvollständiges Wissen (Interpretation) und Metawissen (Verweise) in diese Kategorie. Für die Entwicklung von XPS sind diese Wissensarten jedoch relativ ungeeignet.

Kim und Kim untersuchen in einer Veröffentlichung die Unterschiede zwischen prozeduralem und kausalem Wissen und überprüfen in einer praktischen Anwendung im CAD-Bereich die Bewertung und Evaluierung der Ergebnisse [KiKi-11] [KiKi-10]. Sie kommen zu dem Schluss, dass prozedurales Wissen auch deklaratives und kontextuales Wissen beinhaltet und sich durch die Anwendung von kausalem Wissen die folgenden Vorteile ergeben:

> *(1) more decision alternatives; (2) predictive reasoning to advise design decision; (3) diagnostic reasoning to acquire design faults in current product design; (4) dynamic knowledge allowance to give flexible product design; and (5) knowledge integration to keep the product design knowledge in a team, department, and company.*

Wie schon erwähnt, muss für die Entwicklung eines XPS entschieden werden, welche Wissensarten untergebracht werden, in welcher Art und Weise das Wissen dort repräsentiert wird und wie es verwaltet bzw. zugänglich gemacht werden kann. Unter Wissensrepräsentation versteht man die Art der Darstellung bzw. in diesem Fall die Art der Abspeicherung des Wissens. Für einige Beispiele der Wissensrepräsentation soll an dieser Stelle auf Schulze eingegangen werden, der in seinem Bericht aus der Wirtschaftinformatik weitere Möglichkeiten aufzählt [Schu-01]. Dabei teilt er die verschiedenen Optionen in die Kategorien logikbasierte Ansätze, Produktionsregelsysteme, Semantische Netze, Frames und weitere Ansätze zur Wissensrepräsentation ein.

Mit den logikbasierten Ansätzen sind insbesondere die mathematischen Begriffe gemeint, die in Form der Mengenlehre und Informatik schon in Kapitel 2.5 kurz angesprochen wurden. Schulze bezieht sich im mathematischen Bereich auf den Begriff der Inferenz bzw. der logischen Schlussfolgerung, die sich in die Kategorie des kausalen Wissens zuordnen lässt. In erster Linie geht er auf die Syntax (Summe aller beinhalteten Zeichen) und die Semantik (Summe aller beinhalteten Bedeutungen) der Prädikatenlogik ein, die es erlaubt Sachverhalte darzustellen und zu verknüpfen. Da diese Technik in vielen verschiedenen Zweigen der Wissenschaft angewendet werden kann, stellt die Prädikatenlogik eine wichtige grundlegende Form der Wissensrepräsentation dar.

Schulze geht weiterhin auf sogenannte Produktionsregelsysteme ein, die Wissen in Form einer Datenbasis und einer Vielzahl von Regeln abspeichern und mit Hilfe von Parameter-Wert-Paaren repräsentieren. Ähnlich einer Inferenz können so Verhältnisse oder Relationen verdeutlicht werden.

Um Wissen in grafischer Form zu visualisieren bezieht sich Schulze auf semantische Netze. Diese sind vergleichbar mit Mindmaps und können auch als Graphen (vgl. Kapitel 3.9) angesehen werden, deren Knoten die Sachverhalte darstellen und

die Kanten die Beziehungen untereinander. Wie in Abbildung 3.1 zu sehen, können ebenfalls hierarchische Strukturen realisiert werden (vgl. Kapitel 2.2.3).

Vergleichbar mit den beschriebenen semantischen Netzen und den Mindmaps sind die sogenannten Frames, die aus Slots und Fillern aufgebaut sind. Slots beschreiben dabei die Vielzahl der Möglichkeiten bzw. Eigenschaften, die ein Objekt besitzen kann und die Filler stellen die konkreten Werte oder Einträge dar.

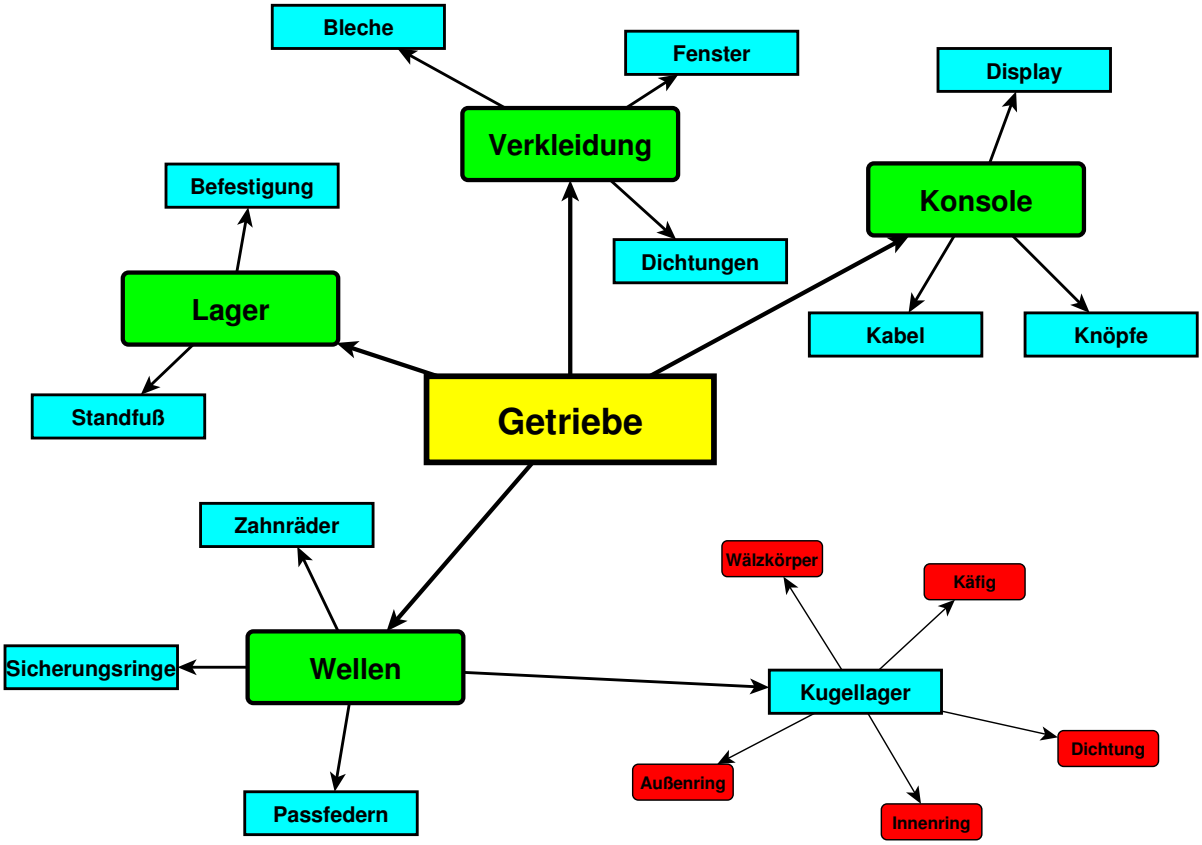

Abbildung 3.1: Mindmap eines Getriebes

In der Kategorie der sonstigen Ansätze zur Wissensrepräsentation geht Schulze noch auf neuronale Netze, deren Ziel es ist ein menschliches Gehirn zu imitieren und die im Bereich der künstlichen Intelligenz eingesetzt werden, und Fuzzy-Systeme ein, die im Gegensatz zur konventionellen Logik nicht nur mit zwei, sondern mit mehreren möglichen Zuständen arbeiten und so gewisse Wahrscheinlichkeiten als Ergebnis errechnen.

Neben den hier aufgelisteten Möglichkeiten der Wissensrepräsentation handelt ein weiterer Forschungszweig, der wichtig für die Entwicklung von wissensbasierten Systemen ist, vom Wissensmanagement bzw. dem Umgang mit dem Wissen, das durch die Repräsentationsarten zuerst einmal nur abgespeichert wurde. Milton bezeichnet dies als 'Knowledge Technologies' und listet die folgenden Fähigkeiten auf, die bei der Weiterverarbeitung realisierbar sein sollten [Milt-08]:

- *Identifying what knowledge is important to an organisation;*
- *Deciding what knowledge needs to be captured to provide an appropriate solution to a real-world problem;*
- *Capturing and integrating knowledge from expert practitioners and existing repositories;*
- *Representing and storing knowledge in ways that provide ease of access, navigation, understanding, maintenance and reuse;*
- *Embedding knowledge in computer systems to provide significant and definable benefits to an organisation.*

Van der Elst und van Tooren entwickeln eine eigene Methode auf der Grundlage dieser Fähigkeiten und bezeichnen die Verknüpfung von KBE und KM (Knowledge Management) als KE (Knowledge Engineering) [ElTo-08]. Sie entwickeln ein Konzept mit dem Namen DEE (Design and Engineering Engine), das ähnlich wie ein ganzes KBE-System die Aufgaben der Anwender automatisieren und so beschleunigen soll:

> *The design analysis and optimization process of complex products can be supported by automation of repetitive and non-creative engineering tasks.*

Ihr Konzept besteht aus sechs Phasen, die von der Planung bis zur Umsetzung das System einteilen: Zu Beginn muss der Prozess analysiert werden (1. Process Analysis), um herauszufinden welche routinemäßigen Schritte verbessert oder automatisiert werden können. Es folgt die Wissensakquisition (2. Knowledge Acquisition) bzw. Sammlung von relevantem Expertenwissen, das später untergebracht werden soll. Dieses wird dann aufgearbeitet und strukturiert (3. Knowledge Structuring) um es für den nächsten Schritt, der Weiterverarbeitung bzw. Anwendung (4. Knowledge Application), vorzubereiten. In den letzten beiden Phasen wird das gesamte Konzept implementiert und umgesetzt (5. Integration of the KBE Modules) und muss danach nur noch gewartet werden (6. Support, Maintenance and Training). Sie kommen in einem Test der Implementierung zu dem Ergebnis, dass die Prozesszeit um 80% reduziert werden kann.

Ähnlich wie van der Elst und van Tooren beschreibt auch Sandberg den Prozess des Wissensmanagements im Zuge der Entwicklung von ganzen KBE-Systemen [Sand-03]. Er sieht KBE als eine Subdisziplin der sogenannten KBS (Knowledge Based Systems), die er auch als XPS (vgl. Kapitel 3.2) bezeichnet und stimmt mit den Ausführungen von van der Elst und van Tooren überein, dass die Ingenieure, die mit XPS arbeiten, immer noch die kreative Arbeit übernehmen, die Software aber die routinemäßigen Prozesse automatisieren soll.

Verhagen et al. untersuchen in ihrer Veröffentlichung die Auswirkungen und Fortschritte, die sich durch ein geschicktes Wissensmanagement und die Anwendung von

KBE-Systemen in den letzten Jahren ergeben haben [VBDC-11]. Sie kommen zu dem Ergebnis, dass zwar die Vorbereitungs- bzw. die Planungszeit bei großen Projekten länger dauert als bei konventionellen Methoden, allerdings kann die Gesamtzeit der Durchführung eines Projektes gesenkt werden, da die Umsetzung eines ausführlich erarbeiteten Planes weniger Zeit in Anspruch nimmt.

Diese Ergebnisse zeigen, dass bei der Vorbereitung zur Entwicklung von KBE-Systemen schon viele wichtige Grundlagen durch geeignete Wissensrepräsentationen und ein funktionelles Wissensmanagement geschaffen werden sollten, damit sich die Umsetzung und Implementierung des Konzeptes einfacher gestaltet.

Des Weiteren soll geklärt werden, welche Möglichkeiten der Wissensakquisition bzw. der Sammlung und Bereitstellung von Wissen bei der Entwicklung von Softwaresystemen existieren und wie die Anforderungen an ein solches System noch vor der Implementierung definiert werden können. Wie schon oben erwähnt, bestehen wissensbasierte Systeme aus einer Wissensbasis und einer zweiten Komponente, die die Lösungsansätze für bestimmte Probleme beinhaltet. Bei jeder Neuentwicklung muss festgesetzt werden, wie die Wissensbasis aufgebaut wird, damit sie den Anforderungen, die an das Programm gestellt werden, gerecht werden kann.

Ein Fachgebiet bei der Softwareentwicklung, das sich mit der Bestimmung der Anforderungen beschäftigt, wird als RE (Requirements Engineering) bezeichnet. Bezogen auf die Neuentwicklung von wissensbasierten Systemen müssen also bevor die Implementierungs- bzw. Programmierungsphase startet die Erwartungen und die Voraussetzungen für die Umsetzung eines solchen Programmes herausgefunden werden. Ein gutes Beispiel für die Gesetzmäßigkeiten des RE bietet die Dissertation von Jiang [Jian-05]. Diese beschreibt die genaue Definition der Anforderungen an zu entwickelnde Produkte oder Softwareprogramme. Die Ziele und Teilziele sollen also vor der eigentlichen Entwicklung festgelegt und definiert werden, um den gesamten Entstehungsprozess besser planen zu können und so die Kosten zu senken. Jiang entwickelt in Zuge dieser Dissertation ein Rahmenkonzept namens FRERE ("A Framework for Requirements Engineering Process Development"). Zuerst erforscht er, welche Anforderungen ein solches Konzept erfüllen soll. Nach der Entwicklung für spezielle Anwendungsfälle wird das Konzept erprobt und validiert.

Die Implementierung der Methode ist in Form eines Computerprogrammes realisiert, in das der Anwender alle wichtigen Eigenschaften des Projektes eingeben kann. Diese sind beispielsweise Projektumfang, Wichtigkeit der Wiederverwendung der Informationen, Grad an 'Outsourcing' oder Kundenverfügbarkeit. Der Benutzer wird daraufhin von dem Programm bei der Entscheidungsfindung unterstützt. Abschließend führt Jiang eine Fallstudie in einem speziellen Anwendungsfall durch.

Nach Gonzales beschreibt Jiang die Wissensakquisition als Teil der Vorbereitung bei der Entwicklung wissensbasierter Systeme:

> *Knowledge acquisition is the process of extracting, structuring and organizing knowledge from a certain source, such as human experts, technical documents and literature where the salient knowledge can be identified. The most commonly used techniques in this process are interviews, questionnaires, and contextual inquiries.*

Ein anschauliches Beispiel für die Anwendung der Wissensakquisition und die Erschaffung einer Wissensbasis zeigt die Veröffentlichung von Park [Park-03]. Dort stellt er eine Methode vor, die Produktwissen für den Fertigungsprozess sammelt und so aufbereitet, dass die eindeutigen Fertigungsschritte an den jeweiligen Maschinen - insbesondere der CNC-Bearbeitung - genau definiert sind. Am Beispiel eines einzelnen Bauteils, das gefertigt werden soll, müssen zum einen die Informationen über die Geometrie als 3D-Modell vorliegen und für die eigentliche Herstellung die Möglichkeiten, die die vorhandenen Maschinen bieten, ebenfalls bekannt sein.

Die Software soll im Zuge der Anwendung eine Datenbank (Wissensbasis) aufbauen, die die verschiedenen Lösungsmöglichkeiten bzw. Vorgehensweisen bei der Fertigung beinhaltet. Park nennt vier verschiedene Wissenselemente bei der Prozessplanung, die zwischen Fakten (Facts), Bedingungen (Constraints), Denkweise (Way of Thinking) und Regeln (Rules) unterscheiden. Am Beispiel einer Senkbohrung mit mehreren Ebenen zeigt Park die möglichen Vorgehensweisen bei der Umsetzung am Bauteil.

Wie schon erwähnt existieren zahlreiche Methoden das Expertenwissen für die Umsetzung in einem wissensbasierten System zu sammeln. Ratchev, Hirani und Bonney stellen eine wissensbasierte Methodologie für die Kreation von flexiblen Fertigungszellen vor, die jederzeit wieder konfiguriert bzw. an die aktuellen Bedingungen angepasst werden können [RaHB-07]. Die Wissensakquisition geschieht in diesem Fall durch den Abgleich der Benutzeranforderungen mit dem existierenden Wissen der Zulieferer bezüglich Designregeln und -prinzipien, Modulen von verschiedenen Anbietern, neuen Technologien und eigenen sowie fremden Produkten. Das vorhandene Wissen über das Gesamtprojekt, das Design, die Herstellung und die Kosten kann so gesammelt, sortiert und für das Design sowie die Konfiguration von Fertigungszellen benutzt werden.

Verglichen mit diesen Ansätzen zum Aufbau einer Wissensbasis untersuchen Cochrane et al. die Wiederverwendung von Wissen in Entscheidungsunterstützungssystemen (Decision Support Systems) [CYCH-08]. Sie gehen dabei der Frage nach inwiefern es möglich ist Prozesswissen, das für die Herstellung von Produkten nötig ist,

wiederzuverwenden. Sie entwickeln dabei eine Prozedur, die in drei Teile eingeteilt ist:

1. Die Trennung von Informationen und Wissen
2. Die Trennung von Produktwissen vom Wissen über den Herstellungsprozess
3. Die korrekte Klassifikation des Herstellungswissens

Am Beispiel eines Flugzeugtriebwerkes werden die Ansätze beispielhaft verdeutlicht.

Die hier angesprochenen Veröffentlichungen zeigen, dass viele unterschiedliche Möglichkeiten der Wissensakquisition existieren und sich das Wissen, das gesammelt und später im System untergebracht wird, maßgeblich unterscheiden kann. Je nachdem welche Anforderungen erfüllt werden sollen, kann beispielsweise Konstruktions-, Marketing- oder auch Fertigungswissen im System umgesetzt werden.

Sowohl die in diesem Kapitel angesprochenen wissensbasierten Systeme, als auch die im folgenden Kapitel beschriebenen XPS bilden beispielhafte Grundlagen für die Eigenentwicklungen im zweiten Teil der Arbeit. Insbesondere die Algorithmen in den Kapitel 5 und 6 können als wissensbasiert angesehen werden, da für ihre Umsetzung zuerst eine Informationsextraktion aus der gegebenen Datenbank durchgeführt und mit der Erstellung der Signaturen eine Wissensbasis angelegt wird.

3.2 Expertensysteme

Um die Grundzüge von XPS zu erklären und um Beispiele aufzuzeigen wie solche XPS aussehen könnten, sollen zu Beginn dieses Kapitels einige Veröffentlichungen vorgestellt werden, die ihren Entwurf zum Thema haben.

Der Anspruch der Habilitationsschrift von Rude liegt darin die klassischen Methoden des Konstruierens in Computerprogrammen umzusetzen [Rude-98]. Das Ziel besteht darin den Entwicklern von neuen Produkten eine Hilfe zur Verfügung zu stellen, die beim Konstruktionsprozess allgegenwärtig ist.

Basierend auf diesem Anspruch lassen sich mehrere Teilziele formulieren, die - sollten Sie realisiert werden - teilweise die Produktentwicklung automatisieren könnten (Seite 2). Ziel wird es sein, dem Anwender während der Konstruktion Informationen aus einer Daten- und einer Methodenbank mittels eines Dialogsystems zur Verfügung zu stellen. Dabei sollten immer mehr Informationen (z.B. DIN-Normen, Materialkennwerte, Toleranzen) angezeigt bzw. zugänglich gemacht werden, als vom Benutzer eingegeben wurden. Der gesamte Entwicklungsprozess, der früher auf konventionelle Weise in Form von Teamarbeit gelöst wurde, soll mit Hilfe der verfügbaren Methoden und Informationen, die vom CAE-System zur Verfügung gestellt werden, auch von einer Einzelperson durchgeführt werden können.

Allerdings ist zu erwähnen, dass auf diese Weise Teamarbeit nicht überflüssig gemacht werden, sondern weiterhin möglich und auch erstrebenswert sein soll. Das System dürfte zusätzlich in der Lage sein den Organisationsaufwand, der mit dem nötigen Informationsaustausch (systemübergreifende Datenformate, die vielleicht sogar automatisch erstellt werden könnten, sind dabei von Nöten) zwischen den teilnehmenden Konstrukteuren einhergeht, zu minimieren. Rude nennt 'Concurrent Engineering' und 'Simultaneous Engineering' als zwei Methoden, die durch eine maschinentechnische Automatisierung massiv vereinfacht werden könnten.

In Zukunft ist es folglich erstrebenswert eine Art von künstlicher Intelligenz zu erschaffen, die möglichst in allen Bereichen der Entwicklung nicht nur den Konstruktionsprozess, sondern auch die Verwaltung, sowie die Vor- und Nachbereitung der Schaffensprozesse begleitet. Diese soll nicht mehr nur wie bisher vorgehen (das heißt Informationen aus Datenbanken abrufen), sondern auch Erfahrungswerte aus früheren Projekten - ebenso wie ein erfahrener Konstrukteur - berücksichtigen. Ein wichtiger Teil der künftigen Forschung wird sicherlich die Frage behandeln, wie solche Informationen gespeichert und vor allem abrufbar gehalten werden können.

In Kapitel 11 beschreibt Rude ausgehend vom Stand der Technik zur Zeit der Veröffentlichung der Habilitationsschrift 1998 vor allem die Möglichkeiten, die die Vernetzung der Computer in Form des Internets bieten. Aus heutiger Sicht ist anzumer-

ken, dass die formulierten Teilziele wie beispielsweise der Aufbau umfangreicher Wissensspeicher (Seite 339), sowie netzbasierte Wissensbereitstellung und Wissenserwerb (Seite 341) in vollem Umfang (z.B. in Internetforen in allen Fachrichtungen, die mit Hilfe von Suchmaschinen leicht zugänglich sind) erreicht wurden.

Allerdings kommt Rude zu dem Schluss, dass der Stand der Konstruktionsmethodik (noch) nicht ausreicht, um eine sinnvolle Implementierung in Form von Software umzusetzen, die in der Lage ist, eigenständig zu planen und wie oben beschrieben auch Erfahrungswerte zu berücksichtigen. Vielmehr müsste eine neue, eigenständige Konstruktionstheorie für den Rechner entwickelt werden, die sich grundsätzlich von der für den Menschen geschaffene unterscheidet. Der Grund ist, dass ein Computer für die Ausführung von komplexen Sachverhalten, wie zum Beispiel das Arbeiten mit Vermutungen oder das Revidieren von Annahmen (Seite 348) einen großen Umfang von Informationen benötigt, die für einen Menschen selbstverständlich sind, aber einer Maschine zusätzlich vermittelt werden müssen. Diese Tätigkeiten besitzen laut Rude große Bedeutung für das ingenieurmäßige Vorgehen (Seite 348).

Die oben beschriebenen Probleme lassen sich nur arbeitsteilig mit Fachleuten aus der Informatik, den Ingenieurswissenschaften und Forschern aus dem Bereich der künstlichen Intelligenz lösen. Nicht nur bei der Entwicklung von komplexen wissensbasierten Systemen, sondern auch bei der späteren praktischen Anwendung in den Unternehmen soll die Vernetzung und die interdisziplinäre Teamarbeit der unterschiedlichen Arbeitsbereiche eine tragende Rolle spielen.

Ein weiteres Beispiel für ein XPS im Bezug auf die Produktentwicklung ist die Arbeit von Strohmeier [Stro-06]. Er formuliert zu Beginn die These, dass Wissen - insbesondere Expertenwissen in der heutigen Zeit ein wertvolles, immaterielles Wertschöpfungsgut ist und es einer besonderen Art von Management (KM) bedarf, um dieses zu verwalten (Seite 5). Laut Strohmeier sollte in Zukunft Wissen als Ressource betrachtet werden, mit der bewusst gewirtschaftet und die verwaltet werden soll (Seite 5). Gemeint ist, dass der virtuelle Produktentwicklungsprozess (CAE) so organisiert werden muss, dass wissensbasiertes Konstruieren ermöglicht wird.

Strohmeier behandelt in seiner Dissertation den Ansatz, dass für die standardmäßigen Konstruktionsprogramme, die beispielsweise in einem Unternehmen benutzt werden, neuartige Softwaremodule entwickelt und eingebunden werden könnten, die während des laufenden Entwicklungsprozesses Informationen liefern oder auf Fehler hinweisen. Ziel ist es ein sogenanntes XPS zu schaffen, das auf allen Gebieten die Kenntnisse liefern kann, die ein erfahrener Ingenieur persönlich jahrelang gesammelt hat. Diese bestehen insbesondere aus strategischem Wissen (Seite 16), das bei der Lösung von Problemen benötigt wird.

Der große Vorteil dabei ist, dass ein menschlicher Experte meistens nur auf ein einziges Fachgebiet spezialisiert ist, wohingegen das Programm so umfangreich wie möglich sein sollte.

Die vier folgenden Komponenten sollte ein solches XPS laut Strohmeier zwangsläufig beinhalten (Seite 24):

1. Wissensbasis (eine Datenbank, auf der die Informationen nur abgelegt sind)
2. Wissensverarbeitungskomponente (für Schlussfolgerungen und Argumentationen)
3. Wissenserwerbskomponente (zur Veränderung der Informationen)
4. Erklärungskomponente (Hilfesystem für Benutzerfragen)

Die Entwicklung einer derartigen Software wird in Zukunft weitere Forschung auf dem Gebiet der ingenieurswissenschaftlichen künstlichen Intelligenz erfordern. Eine VKI (Verteilte Künstliche Intelligenz) ist ebenfalls denkbar und erstrebenswert. Besonders für umfangreiche Berechnungen (z.B. im FEM-Bereich) stellt eine Mischung aus mehreren Systemen, die mit Hilfe paralleler Algorithmen die Rechenaufgaben separat lösen, eine sinnvolle Alternative dar.

Laut Strohmeier liegt Expertenwissen nicht deterministisch, sondern in einer unscharfen Form vor (Seite 25), weshalb in Zukunft künstliche neuronale Netze bei der maschinentechnischen Umsetzung von begründeten Schlussfolgerungen (Inferenz), die programmiertechnisch relativ leicht umzusetzen sind, und auch intuitiven Schlussfolgerungen hilfreich sein könnten. Diese sind zwar unbegründet, aber, wie die menschliche Herangehensweise an ein Problem bestätigt, ebenso effektiv.

Schon jetzt ist vorhersagbar, dass die Einteilung in die oben genannten vier Gebiete prädestiniert für die Entwicklung von einzelnen Modulen ist, die nach und nach in das System eingebunden werden können und die Software so schrittweise vervollständigen. Strohmeier beginnt in Kapitel 3.4 bei der Erklärung des Aufbaus solcher Module mit der Definition von kleinstmöglichen Wissenseinheiten (Seite 49):

> *Die Gesamtheit aller speziell notwendigen Softwareartefakte, die an der Lösung einer bestimmten (geometrisch) konstruktiven Aufgabe beteiligt sind, bildet eine Wissenseinheit des KBE.*

Zusammengefasst zu Teileinheiten bilden diese dann später die Module, deren Informationen sich durch Kopplungen und Referenzen bei der Implementierung in das Hauptprogramm teilweise überschneiden können. Strohmeier bezeichnet Module als lauffähige Programme, die Wissen beinhalten und kleine Bestandteile des gesamten Konstruktionssystems darstellen (Seite 65).

Vorteile eines modularen Aufbaus von Software lassen sich in der Weiterentwicklung, dem Zusammenbau und der Vermarktung finden. Kunden könnten beispielsweise ausschließlich die für sie notwendigen Anwendungen erstehen und müssen diese

dann nur noch zusammensetzen (eine globale Programmiersprache wird dafür notwendig sein). Auch die Überprüfung und eine individuell zugeschnittene Weiterentwicklung sind bei Programmen in einer solchen gestückelten Form viel einfacher durchführbar.

Strohmeier beschreibt in Kapitel 6 abschließend, dass die Vorgehensweise der modularen Softwarepakete, die er in seiner Dissertation aufzeigt, idealerweise in einigen Jahren zu einem Wissenspool standardisierter Module werden könnte, der alle wünschenswerten Funktionen enthält und so beliebig je nach Bedarf zusammengesetzt werden kann (Seite 106).

Ein Ansatz für die Wiederverwendung von Expertenwissen (Knowledge Reuse), das mit der Unterstützung von Assistenzsystemen in einer Datenbasis gespeichert wird, beschreiben auch Kulon, Broomhead und Mynors [KuBM-06]. Sie zeichnen in ihrer Veröffentlichung den Grundriss einer neuartigen Systemarchitektur, die wissensbasiertes Konstruieren erlaubt. Am Beispiel eines design-ingenieurstechnischen Entwicklungsprozess für Schmiedeteile werden die Vorteile eines Systems dargelegt, das über eine Internetanbindung den Benutzern nicht nur einen Austausch von Geometrie-, sondern auch von Zusatzinformationen wie z.B. Entwicklungs-, Analyse- und Fertigungswissen ermöglicht. Wenn dann nach und nach eine Art Datenbank bzw. ein Variantenpool von einzelnen Produkten entstanden ist, kann jeder Benutzer die vorliegenden Bauteile durch eine Änderung der Parameter bewerten, weiterentwickeln oder als Designgrundlage benutzen. So kann die langwierige Designarbeit, die am Anfang einer jeden Neukonstruktion steht, minimiert bzw. übersprungen werden.

Des Weiteren werden nicht nur die 'von Hand' geschaffenen Produktvarianten auf der Datenbank bereitgestellt, sondern auch Designregeln, Normen und Materialkennwerte sollen dem Benutzer im laufenden Entwicklungsprozess angezeigt werden, damit direkt im ersten Entwicklungsschritt ein möglichst fehlerfreies Produkt generiert werden kann. Neben der theoretischen Systemarchitektur wird in dem Artikel außerdem auch auf die Serverarchitektur, sowie auf die Gestaltung der Benutzeroberfläche eingegangen. Die Autoren schlagen eine webbrowserbasierte Benutzeroberfläche vor, die mit Hilfe eines Strukturbaumes die Beziehungen der einzelnen Konstruktionsbestandteile untereinander darstellt, damit Fehler besser erkannt und an der richtigen Stelle verbessert werden können.

Baxter et al. liefern ebenfalls einen Ansatz für die Wiederverwendung von Expertenwissen [BGCH-07]. Sie schlagen eine Methode vor, die eine Vorgehensweise bei der Wiederverwendung von Designwissen behandelt. Dabei sollen nicht nur geometrische Daten gespeichert, sondern umfangreichere Informationen benutzt werden. Unterschieden wird zwischen Prozesswissen, Produktwissen und Auftragswissen. Ziel ist es ein Prozess- und ein Produktmodell zu erstellen und durch Verwendung der richtigen

Parameter beides miteinander zu kombinieren, so dass es zugeschnitten auf eine konkrete Aufgabe bestmöglich genutzt werden kann. Außerdem wird vorgeschlagen eine ähnliche Vorgehensweise in ein KBE-System einzugliedern, was vor allem den Vorteil bringt, dass bei neuen Designaufgaben, die aber älteren, schon ausgearbeiteten ähneln, das bereits erstellte Wissen wieder zu verwenden.

Bezogen auf die Entwicklung einer künstlichen Intelligenz stellt Pokojski folgende Idee vor [Poko-06]. Es handelt sich dabei um ein Begleitprogramm für CAD-Konstrukteure, das auf Basis der wissensbasierten Konstruktion und einem IPS (Intelligent Personal Assistant) das Wissen, das der Benutzer bei der Erstellung eines neuen Bauteiles anwendet, abspeichert und bei späteren Prozessen wieder zur Verfügung stellt. Mit einer solchen Art von begleitender künstlicher Intelligenz sollen die ganze Zeit über die Arbeitsschritte aller Anwender abgespeichert und nachvollzogen werden, so dass die gesammelten Informationen auch für andere Benutzer später wieder zur Verfügung gestellt werden können und diese davon profitieren.

Auch Boicu bezieht sich in seiner Dissertation auf eine künstliche Intelligenz [Boic-03]. Er entwirft ein Assistenzsystem, das den Benutzer bei der Entwicklung von ingenieurstechnischen Aufgaben unterstützt. Dieses Assistenzsystem soll vor allem dann zur Anwendung kommen, wenn nicht alle Informationen vorhanden sind (Incomplete Knowledge). Des Weiteren soll es lernfähig sein bzw. eine Art von künstlicher Intelligenz besitzen, die in der Lage ist, Vorgänge abzuspeichern und wiederzuverwenden.

Boicu erklärt zu Beginn wie Assistenzsysteme allgemein aufgebaut sind und verdeutlicht die logischen Zusammenhänge und Schlussfolgerungen mit Strukturbäumen. Weiterhin wird verdeutlicht, wie das Wissen, das in einem Datenpool gesammelt wird, bewertet und gewichtet wird. Wenn es wiederverwendet werden soll, dann ist ein System nötig, das die richtigen Wissensstücke automatisch aus der Datenbasis heraussucht und dem Benutzer anzeigt.

Pugliese, Colombo und Spurio benutzen für die Verwendung ähnlicher Ansätze die Begriffe DA (Design Automation) und PDM [PuCS-07]. Als Voraussetzung für eine funktionierende Automatisierung des Konstruktions- bzw. Designprozesses sehen sie:

1. Knowledge Acquisition
2. Knowledge Collection
3. Knowledge Formalization

Unter der Formalisierung von Wissen werden die Abspeicherung und dessen Systematik verstanden. An dieser Stelle zitieren die Autoren die drei Projekte MOKA (Methodology for Knowledge Based Engineering Applications), DEKLARE (Design Knowledge Acquisition and Redesign Environment) und KOMPRESSA (Knowledge-Oriented Methodology for the Planning and Rapid Engineering of Small-Scale Applications), die sich alle mit diesem Thema beschäftigen. Im Zuge ihres eigenen Ansatzes

wurde ein KBE-System entwickelt, das während des Entwicklungsprozesses einen Datenspeicher (Data Repository) auf bereits vorhandene Komponenten oder Produkte überprüft und dem Entwickler zur Verfügung stellt. Gleichzeitig wird dieser Speicher automatisch mit den neu entstandenen Informationen gefüllt.

Ein anschauliches Beispiel für ein solches wissensbasiertes System liefert die Dissertation von Wunsch [Wuns-05]. Die Idee ist die kundenbezogene Individualisierung von Schuhen, dessen Grundlagen allerdings in Zukunft auch auf andere Produkte übertragen werden sollen. Mit Hilfe von Fotos und der Anwendung von 3D-Scannern sollen bereits beim Verkaufsgespräch Daten über die spezifischen Anforderungen gesammelt und gespeichert werden, so dass die eigentliche Entwicklung und Produktion bzw. die Wartezeiten verkürzt werden.

Um die Optimierung eines derartigen Produktentwicklungsprozesses kümmern sich sowohl Deng als auch Gaag [Deng-07] [Gaag-10]. Beide gehen davon aus, dass durch die Analyse des Wissensmanagements, das bei der Neukonstruktion angewendet wird, Schlüsse für die zukünftige Anwendung von Informationseinheiten gezogen werden können, die für spätere Entwicklungen von Vorteil sein können. Daraus lassen sich die Anforderungen an ein KBE-System definieren, das für die geforderten Funktionalitäten ausgelegt ist. Gaag nimmt Bezug auf Ontologien (vgl. Kapitel 2.5), die bei der funktionsorientierten Lösungssuche angewendet werden sollen und setzt als Forschungsziel eine Art Suchmaschine, die für jede Problemstellung oder Herausforderung die richtige Lösung finden kann.

Auf die Optimierung der Prozesse und der damit einhergehenden Senkung der Kosten konzentriert sich auch die Dissertation von Sarder [Sard-06]. Er entwickelt dort eine Methode, die er DKAP (Design Knowledge Acquisition Process) nennt. Das Produkt- und Prozessdesign von Neuentwicklungen kann von der Anwendung einer Ontologie vereinfacht werden, was sich vor allem auf die Kosten auswirkt. Außerdem sollen Hersteller, die neue Produkte auf den Markt bringen wollen, diese schneller fertigstellen und zur Marktreife bringen. Ontologie bedeutet in diesem Fall, dass bestimmte Bereiche von Wissen definiert und deren Beziehungen untereinander ebenfalls bestimmt werden. Mit Stichwörtern wie z.B. 'Knowledge Sharing' oder 'Enterprise Modelling' verdeutlicht Sarder die Möglichkeiten eines modularen Aufbaus, der die Wiederverwendung von bereits entwickelten Produkten oder vorhandenem Wissen begünstigt.

Sure, Ehrig und Studer beschreiben in ihrer Veröffentlichung genau die von Sarder angewandte Zweckmäßigkeit von Ontologien und konzentrieren sich dabei auf die Darstellung und Verknüpfung von Wissen [SuES-06].

3.3 Model Based Definition

Die Grundidee des MBD (Model Based Definition) nach Alemanni, Destefanis und Vezzetti ist es alle Informationen, die für den gesamten Zyklus eines Produktes (PLM) notwendig sind (vgl. Kapitel 3.5), in einem CAD-Modell unterzubringen [AlDV-10]. Das bedeutet, nicht nur die Informationen über die Geometrie eines Bauteiles, sondern auch alle Daten, die für die Fertigung benötigt werden, wie z.B. Toleranzen, Materialkennwerte, Maschinenanweisungen oder Oberflächenbeschaffenheit, werden im CAD-Modell abgespeichert. Diese Methode hat den Vorteil, dass beim Datentransfer nicht viele verschiedene Dateien bzw. Pläne ausgetauscht, sondern bei der Anwendung nur die interessanten Daten aus dem Modell ausgelesen werden müssen. Auch eine Analyse im Zuge der Eigenentwicklungen im zweiten Teil der Arbeit wäre von großem Interesse.

Ein weiterer wichtiger Bestandteil des MBD ist außerdem die Prozess- bzw. Herstellungsplanung der Produkte. Um Zeit und somit Fertigungskosten zu sparen, werden die Fertigungs- und Montageschritte, die bei der Produktion einzelner Teile notwendig sind, analysiert und sowohl die Reihenfolge als auch die Mechanik eventuell zum Einsatz kommender Roboter optimiert.

Ein anschauliches Beispiel liefert die Veröffentlichung von Chu und Gadh [ChGa-96]. Sie beschäftigen sich mit der Frage, inwiefern eine Klassifizierung einzelner Bauteile in Features den Fertigungsprozess vereinfachen kann. Die Vorgehensweise bei der Fertigung und die Frage in welcher Reihenfolge die Features in das Rohteil eingearbeitet werden, werden hierbei erörtert. Es erfolgt die Unterteilung in STAD-Features (Single Tool Axis Direction) und MTAD-Features (Multiple Tool Axis Direction). Als STAD werden Features bezeichnet, bei denen das Fertigungswerkzeug sich nur aus einer einzigen Richtung annähern kann. Bohrungen, die nicht durchgängig sind, sind ein einleuchtendes Beispiel. MTAD-Features lassen sich von mehreren Richtungen fertigen. So kann eine Durchgangsbohrung z.B. von der einen oder der anderen Seite umgesetzt werden.

Des Weiteren muss bei der Einarbeitung der Features in das Halbzeug die Reihenfolge so geplant werden, dass die Bearbeitungszeit möglichst kurz und die Arbeitsschritte, die sich durch umspannen oder Neuausrichtungen des Bauteils ergeben, minimiert werden. Beispielsweise könnte das Material, das für die Erschaffung eines Features entfernt werden muss, zuvor zum Einspannen bei der Fertigung eines anderen Features benutzt werden.

Peifu und Weifeng gehen einem ähnlichen Ansatz zur Fertigungsoptimierung nach [PeWe-99]. Sie stellen ein intelligentes CAD-System vor, das die Vorgänge bei kaltverformenden Gesenkschmiedeprozessen in der Automobilindustrie analysiert und den

Anwender bei der Auslegung der Produktionsvorgänge unterstützt. Die Autoren teilen die Voraussetzungen für die Entwicklung von Herstellungsprozessen in die folgenden drei Gebiete ein:

1. Faktenwissen: Alle Informationen, die vor Beginn des Entwicklungsprozesses bekannt sind, werden als Fakten bezeichnet. Dazu zählen auch die Vorgaben, die beim Endprodukt erfüllt sein müssen.
2. Prozeduren: Hierzu zählen alle Methoden, die für Berechnungen oder Entwicklungen schon bekannt sind, wie etwa numerische Methoden oder andere Algorithmen.
3. Heuristik: Bei komplizierteren Schmiedeprozessen können bestimmte Vorgänge durch Heuristik vereinfacht werden. Wenn beispielsweise eine Baugruppe aus Schmiedeteilen aufgebaut werden soll, dann kann schon während der Entwicklung ein modulares Design erzielt werden, indem die Teile baukastenförmig entwickelt werden.

Das intelligente CAD-System, das hier vorgestellt wird, beinhaltet außerdem Vorgehensweisen aus der Fuzzy-Logik und ist aufgrund der Anwendung neuronaler Netzwerke lernfähig.

Zhang et al. gehen in ihrem Ansatz noch einen Schritt weiter und verbinden die Fertigungsschritte mit der 3D-Modellierung im CAD-Programm [ZSFH-10]. Sie beschreiben eine Methode zur Fertigung von ingenieurstechnischen Bauteilen. Ein Computerprogramm wird vorgestellt, in das zu Beginn mehrere technische Zeichnungen eingespeist werden müssen, in denen jeweils die einzelnen Geometrien nach einem Fertigungsschritt (Umsetzung eines Features) beinhaltet sind. Die Autoren veranschaulichen ihre Methode beispielhaft an einem Bauteil (Welle), das schrittweise vom Rohmaterial bis zum fertigen Teil aufgebaut wird. Das Computerprogramm benötigt Informationen über das Halbzeug, das später im Fertigungsprozess als Ausgangsteil dient und jeweils eine zweidimensionale Zeichnung in Form einer DXF-Datei pro Fertigungsschritt, die die genaue Geometrie exakt definiert.

Bei Bauteilen, die relativ viele Details enthalten, sind folglich verschiedene Prozessschritte und ggf. auch mehrere Maschinen nötig. Eine DXF-Datei beschreibt also einen Fertigungsschritt an einer Maschine. Für filigrane Bauteile sind dementsprechend viele technische Zeichnungen nötig, die immer weiter ins Detail gehen (vgl. Kapitel 3.9).

Neben den technischen Zeichnungen wird außerdem eine weitere Datei gebraucht, die eine Art Anleitung für die Herstellung beinhaltet. Normalerweise übermittelt der Konstrukteur dem Fertiger so die einzelnen Arbeitsschritte in ihrer richtigen Reihenfolge. In dieser Anleitung sind außerdem Randbedingungen und Zusatzinformationen wie z.B. Fertigungsdauer, Vorschubgeschwindigkeit, Materialkennwerte, Oberflächen-

beschaffenheiten, Toleranzen o.ä. definiert. In der hier präsentierten Vorgehensweise ist diese Zusatzdatei ebenfalls notwendig, allerdings wird ihr Inhalt automatisch von der Software interpretiert.

Die gesamte Vorgehensweise kann nicht nur an den entsprechenden Fertigungsmaschinen angewendet werden, sondern die Autoren verdeutlichen auch die Prozessschritte mit der CAD-Software NX. Hier ist das Prinzip ähnlich. Der Benutzer muss auch zu Beginn die Inputinformationen überliefern und das Programm rekonstruiert das Bauteil dann virtuell in der 3D-Umgebung von NX. Der Benutzer kann also dabei zusehen, wie jeder Fertigungsschritt einzeln ausgehend vom Rohmaterial umgesetzt wird und erhält am Ende das fertige 3D-Modell mit allen Fertigungsdetails als NX-Datei.

In Zukunft soll die hier präsentierte Methode noch verbessert werden, so dass kompliziertere Bauteile mit weiteren Fertigungsschritten verarbeitet werden können.

Im Großen und Ganzen lässt sich MBD zu einer wissensbasierten Methode zusammenfassen, die auf die Produktion ausgelegt ist, während sich KBE bzw. wissensbasierte Konstruktion auf die Entwicklung von neuen Produkten konzentriert.

3.4 Featureerkennung

Um die Verknüpfung zwischen wissensbasierten Systemen und den Features der Bauteile bzw. des Strukturbaumes herzustellen, liegt eine Analyse nahe. Da der gesamte Strukturbaum ein jedes Bauteil eindeutig definiert, lassen sich mit den so gegebenen Informationen viele verschiedene Anwendungen entwickeln, von denen einige hier vorgestellt werden sollen. Das Buch von Shah und Mäntylä gibt einen ersten Überblick über die featurebasierte Konstruktion verbunden mit den Herstellungstechniken im CAM [ShMä-95].

Lupa konzentriert sich in seiner Dissertation auf die Optimierung von Produkten anhand einer Analyse der Features [Lupa-09]. Er beschäftigt sich mit den Möglichkeiten der automatisierten und wissensbasierten Erstellung von Komponenten und Baugruppen innerhalb von featurebasierten und parametrischen CAD-Systemen (Seite 3). Wissensbasierte Erstellung bedeutet in diesem Fall, dass das Konstruktionssystem Wissensanteile in Form von Modulen dem Anwender zur richtigen Zeit bereitstellen soll.

Lupa beschreibt, wie sich allgemein die Vorgehensweise bei der Erschaffung eines neuen Produktes aus konstruktionstechnischer Sicht in den letzten Jahren - insbesondere aufgrund von VPE (Virtuelle Produktentwicklung) - geändert hat. Abweichend von der klassischen Konstruktionsmethodik, die nach wie vor das Grundgerüst bildet, haben sich rechnergestützte Konstruktionsarten etabliert, von denen vor allem auf die featurebasierte Modellierung eingegangen werden soll.

Nahezu alle heutigen CAD-Programme unterstützen die Featuretechnologie. Allerdings sind damit meist Formfeatures, wie z.B. Bohrungen, Nuten oder Aufsätze, gemeint, bei denen vom Benutzer nur die entsprechenden Parameter angegeben werden müssen und anhand dieser Informationen alles im CAD-System (zugänglich z.B. durch den Strukturbaum) abgespeichert wird. Ein Feature muss allerdings keine Geometrie enthalten (Seite 16). Laut Lupa ist es lediglich ein informationstechnisches Element, das digital weiterverarbeitet werden kann (Seite 16). Wünschenswert wäre folglich ein VPE-System, das die Featuretechnologie auch schon zu Beginn, in der Anfangsphase der Produktentwicklung anwendet, in der zwar ähnliche, gut automatisierbare Schritte zur Planung unternommen werden, allerdings noch keine Geometrie definiert wird.

Problematisch wird es beim Datenaustausch zwischen derartigen Systemen. Selbst wenn eine Software entwickelt wird, die in allen Phasen der Produktentwicklung den Benutzer featurebasiert unterstützt, wird es beim Übergang und Austausch von einem CAD-System zum anderen erhebliche Kompatibilitätsprobleme geben. Meistens werden hauptsächlich die Geometriefeatures, wie erwähnt, anhand ihrer Parameter

abgespeichert und können aufgrund der unterschiedlichen Programmierstruktur der Systeme nicht übertragen werden. Die in Kapitel 2.2 angesprochenen genormten Standardformate, wie z.B. STEP oder XML, sind entwickelt worden, um Objekte auf den gängigen CAD-Systemen einzusehen. Die Features können allerdings wegen der oben beschriebenen parametrischen Abspeicherung nicht übertragen werden und gehen verloren. Laut Lupa existiert derzeit keine Möglichkeit, um parametrische und featurebasierte CAD-Modelle zwischen verschiedenen CAD-Systemen auszutauschen (Seite 26).

Die Voraussetzungen, die in Zukunft für ein ideales CAD-System existieren müssen, sind nach Lupa ein umfangreiches Datenverwaltungssystem (Seite 19) mit systemübergreifenden Schnittstellen und ein PDM (Seite 26), das, mit Hilfe von künstlicher Intelligenz (Seite 29), Wissen in das virtuelle Produkt integriert (Seite 28). Hierbei soll nicht nur bewusstes (bzw. explizites) Wissen, sondern auch Erfahrungswissen (implizit) verarbeitet und zur Verfügung gestellt werden. Weiterhin ist ein Konfigurationssystem notwendig, das die gesamte Verwaltung übernimmt. Die technischen Anforderungen sind folgende (Seite 65, 66):

1. Es muss mit dem eigentlichen CAD-System bidirektional (Seite 65) kommunizieren können
2. Es muss ebenso das PDM-System unmittelbar aktualisieren
3. Es benötigt eine GUI zur Kommunikation mit dem Anwender
4. Es muss den Zugriff auf die Modellierungselemente des CAD-Systems gewährleisten
5. Die Gestaltungsdaten müssen getrennt von Wissensdaten gespeichert werden, um Redundanz zu vermeiden

Zusammenfassend erläutert Lupa, dass viele professionelle CAD-Systeme sich zu Produktentwicklungswerkzeugen entwickelt haben, wobei die Potentiale der verschiedenen Teilwerkzeuge in der Praxis aber noch nicht vollständig ausgeschöpft werden (Seite 109). Am Beispiel eines Verdichterläufers zeigt er die Vorgehensweise im Bezug auf Baugruppen und legt so den modularen Aufbau von umfassenderen VPE-Systemen dar.

Ziel ist es eine hohe äußere Varianz (aus Sicht des Kunden) zu erreichen und dennoch eine möglichst geringe innere Varianz (aus Sicht des Unternehmens) zu sichern (Seite 53). Bezogen auf das Beispiel von Verdichtern bedeutet dies, dass z.B. die Datenbank eines Unternehmens viele verschiedene Gestaltungsmodule beinhaltet, die nach den gegebenen Kundenwünschen nur noch zusammengesetzt werden müssen. Es sind sozusagen vorgefertigte kleine Teile und Baugruppen gespeichert, die dann zu immer größeren Baugruppen zusammengefügt werden. Im Falle eines Verdichters sind das z.B. verschiedene Wellen, Laufräder oder Gehäuse, die sich in Form, Bauweise

oder Größe unterscheiden. Immer, wenn an einem vorhandenen Bauteil oder einer Baugruppe eine Änderung oder gar eine Neukonstruktion vorgenommen wird, wird auch dieses 'neue' Bauteil auf der Datenbank abgelegt, um die Variantenvielfalt und die Auswahlmöglichkeiten immer weiter zu vergrößern.

Der Kunde spezifiziert zu Beginn dem Unternehmen seine Anforderungen - beispielsweise im Pflichten- und Lastenheft (Seite 99). Diese Informationen werden ins Konfigurationssystem eingegeben, das ein erstes Konzept entwirft und eine Angebotserstellung vornimmt (Kostenvoranschlag). Wenn alle Anforderungen genügend detailliert sind, werden automatisch zusammen mit dem Datenbankverwaltungssystem die Gestaltungsmodule ausgewählt und im CAD-Modellaufbau zu übergeordneten Baugruppen zusammengefügt.

Diese Vorgehensweise ist natürlich für alle möglichen Unternehmen denkbar, die sich auf bestimmte Baugruppen oder Maschinen spezialisiert haben. Vielleicht wäre es sogar möglich ein solches Konzept bei Automobilherstellern, deren Baugruppen aus viel mehr Teilen bestehen, anzuwenden. Im Folgenden werden einige Veröffentlichungen vorgestellt, die sich insbesondere mit der Produktionsplanung befassen. Hier wird deutlich, dass die Unterteilung von Produkten in Features und Featuregruppen gewissermaßen für die Reihenfolge der Fertigung von großer Bedeutung ist.

Babic, Nesic und Miljkovic untersuchen beispielsweise Methoden zur automatisierten Feature-Erkennung in Bauteilen namens AFR (Automated Feature Recognition) [BaNM-07]. Sie beschreiben die drei folgenden grundlegenden Probleme, die bei jedem Prozess behandelt werden müssen:

1. Extraktion der geometrischen Primitive (Basisformen wie Linien, Kreise oder Bögen) aus dem CAD-Modell
2. Definition einer passenden Teilerepräsentation für Formfeatureidentifizierung
3. Feature-Muster Erkennung

Im Hinblick auf die CNC-Fertigung der Bauteile, soll der ganze Prozess der automatischen Erkennung von Features in zweidimensionalen Zeichnungen und die Umwandlung in Maschinenanweisungen bzw. in einen Prozessplan automatisch ablaufen. Die Erkennung der Features wird von den Autoren in vier Aufgaben eingeteilt:

1. Erkennung der Oberflächen
2. Definition der Basen
3. Erkennung der Begrenzungen
4. Formextrahierung

Auch Iyer et al. beschäftigen sich mit der Detektion von Features [IJLK-04]. Die Autoren geben einen Überblick über Formerkennungsmethoden, die in der Vergangenheit entwickelt wurden und die in der Zukunft eine wichtige Rolle spielen könn-

ten. Sie setzen sich mit der Frage auseinander wie eine Form angezeigt oder informationstechnisch angegeben werden kann und beschreiben die folgenden Ansätze:

1. Featurebasierte Techniken im Allgemeinen
2. Feature-Erkennung im Bezug auf die Fertigung
3. Graphenbasierte Techniken
4. Histogrammbasierte Techniken
5. Formerschaffung in Form von Voxeldarstellung
6. B-REP bzw. Freiformflächen und Drahtgittermodelle
7. 3D-Objekterkennung

Ma, Huang und Wu gehen davon aus, dass während des Konstruktionsprozesses von neuen Produkten sich zum Teil bestimmte Designaufgaben und -tätigkeiten immer wiederholen [MaHW-09]. Folglich könnten diese sehr gut automatisiert oder für die Anwendung zur Verfügung gestellt werden. Um ihre Theorie zu untermauern zeigen die Autoren eine Feature-Extraktion bzw. Mustererkennung an einigen Bauteilen, die als 'B-REP Solid Models' vorliegen. Sie kommen zu dem Schluss, dass die Designmustererkennung eine Art neuer Forschungszweig für CAD-Anwendungen ist und noch viele Möglichkeiten, wie z.B. personalisierte Datenbanken oder automatisierte Wissensaufdeckung, zukunftsträchtige Alternativen sind.

Auch die Veröffentlichung von Verma und Rajotia ist eine rückblickende Zusammenfassung von Feature-Erkennungsmethoden [VeRa-10]. Auch dort wird der Fokus auf die Fertigungs- und Prozessplanung gelegt und der Frage nachgegangen, in welcher Reihenfolge es am sinnvollsten ist die unterschiedlichen Features in das Rohteil bzw. Halbzeug einzuarbeiten.

Ähnlich gehen Dipper, Xu und Klemm vor [DiXK-11]. Sie stellen eine Methode vor, bei der CAD-Modelle in Form von STEP-Dateien eingelesen und die Interaktionen zwischen den beinhalteten Features bestimmt werden. Dieser Prozess läuft automatisch ab. Im nächsten Schritt wird das ursprüngliche CAD-Modell untersucht und die Ergebnisse aus dem vorherigen Schritt damit verbunden. Für die Prozessplanung, bei der die Fertigungsreihenfolge der einzelnen Features bestimmt wird, soll diese Vorgehensweise von großer Hilfe sein, da einige Features, wie z.B. Durchgangsbohrungen, die durch eine Nut unterbrochen werden, in einem anstatt in zwei Schritten gefertigt werden können.

Die folgenden drei Veröffentlichungen beschäftigen sich nicht ausschließlich mit der Produktion, die durch die Featureanalyse vereinfacht werden kann, sondern auch mit den Möglichkeiten, die sich durch die Anwendung spezialisierter Software im Bezug auf die Einfachheit und die Dauer von Berechnungsprogrammen bieten. An dieser Stelle soll etwa ein Programm zur Vereinfachung von FEM-Berechnungen von White,

Saigal und Owen vorgestellt werden [WhSO-05]. In ihrem Artikel schlagen sie ein Programm vor, das bei der Vernetzung von CAD-Modellen mit hexagonalen Gitterstrukturen die Komplexität, die bei der Berechnung nötig ist, vorhersagt. Zu Beginn wird erklärt, dass der Rechenaufwand stark von den Feinheiten des Bauteiles abhängt. Wenn das Modell relativ groß ist, aber viele Details bzw. Features beinhaltet, so muss das Netz normalerweise an den entsprechenden Stellen verfeinert werden, so dass es den Einflüssen auf die Stabilität an diesen Punkten gerecht werden kann. Aus diesem Grund wird die Möglichkeit geboten das CAD-Modell vor der Vernetzung zu vereinfachen und kleine Features nicht zu berücksichtigen.

Ein weiteres Problem, das bei der Vernetzung von Details auftritt, sind Lücken und daraus entstehende Fehler. Manchmal muss also das Bauteil zuerst repariert werden, wobei ein Programm benutzt wird, dass kleine Lücken (vor allem bei Konstruktionen mit Freiformflächen und Oberflächendesign) erkennt und automatisch schließt. Erst dann kann mit der Vernetzung und der darauffolgenden Berechnung begonnen werden.

Die Komplexität eines Bauteils bzw. dessen Netzes lässt sich anhand vieler verschiedener Variablen festmachen. Das hier vorgeschlagene Programm konzentriert sich allerdings nur auf die Größe der Elemente, die Topologie und die Geometrie. Diese Produkteigenschaften werden folglich automatisch untersucht und ausgewertet. In Folge dessen wird beispielsweise nach Winkeln, kleinen Kanten, kleinen zusammenhängenden Flächen und auch nach unpraktischen Stellen gesucht, die dann gezählt und bei der endgültigen Berechnung der Komplexität des Netzes berücksichtigt werden.

Am Ende kann dann die Zeit, die die Software für die Vernetzung benötigt, annähernd vorhergesagt werden und der Anwender kann entscheiden, ob das aktuelle Netz benutzt wird, oder noch Änderungen vorgenommen werden sollen.

Quadros und Owen beschreiben in ihrer Veröffentlichung eine automatisierte Vorgehensweise zur Featureerkennung und -entfernung bei CAD-Modellen, die ebenfalls auf die Vernetzung der Teile ausgerichtet ist [QuOw-12]. Ihr Programm hat die Aufgabe komplexe Bauteile zu vereinfachen, indem ein Modell mit Hilfe einer Software vernetzt wird und alle Features, die kleiner als ein vom Anwender bestimmter Schwellwert sind, entfernt werden. Wenn beispielsweise ein würfelförmiges Bauteil eingelesen wird, das mehrere Fasen, Bohrungen oder Verrundungen besitzt, so sollte das Programm nach dem Durchlauf nur einen Würfel auslesen, da alle kleineren Features entfernt wurden. Zusätzlich wird dem Benutzer noch erlaubt für ihn wichtige Features oder gar Regionen eines Teiles einzufrieren. Wenn er also eine kleine Bohrung auch noch nach der Vernetzung und Vereinfachung betrachten will, so hat er die

Möglichkeit diese auszuwählen, um so das Programm davon abzuhalten sie bei der Featureentfernung zu löschen.

Der große Vorteil ist, dass alles automatisch abläuft und so auch große Datenbestände vollautomatisiert verarbeitet werden können. Wenn ein Anwender all diese Arbeitsschritte manuell durchführen müsste, würde es sehr viel länger dauern, da alle Teile einzeln geöffnet, betrachtet und bewertet werden müssten. Bei der späteren Weiterverarbeitung eines vereinfachten Netzes entsteht außerdem der Vorteil, dass z.B. für FEM-Berechnungen ein relativ grobes Netz zur Verfügung steht, das zwar die Grundform des Bauteils im Großen und Ganzen widerspiegelt, aber bei Bohrungen oder Aussparungen nicht unnötig fein ist. In dieser Weise wird natürlich die Rechenzeit bei Verformungsberechnungen stark reduziert - vorausgesetzt, dass aus mechanischer Sicht kleine Löcher oder Materialaussparungen vernachlässigt werden können.

Die Arbeitsschritte, die das Programm durchläuft, können in verschiedene Stufen eingeteilt werden. Von dem ursprünglichen CAD-Modell muss als erstes zur Vorbereitung ein diskretisiertes Modell abgeleitet werden. Im nächsten Schritt wird dieses gescannt, um so alle Features zu identifizieren, die dann unterdrückt werden sollen. Die Autoren beschreiben verschiedene Softwaremodule, die bei der Featureerkennung die geometrischen Faktoren messen und auswerten. Je nachdem welche Eingangsparameter der Benutzer zu Beginn definiert hat, werden alle Details, die kleiner als der Grenzwert sind, als Features erkannt und können im nächsten Schritt, bei der Vernetzung des bereinigten Modells, außer Acht gelassen werden. Am Ende gibt das Programm ein fertig vernetztes Modell aus, bei dem alle Features, die kleiner als die Schwellwerte sind, entfernt wurden.

Ein letzter Ansatz im Bezug auf die Featureerkennung kommt von Linghao, Dongming und Hang [LiDH-11]. Sie beschreiben in ihrem Paper eine Vorgehensweise zur Unterscheidung von zwei ähnlichen CAD-Dateien, die mit CATIA erstellt wurden. Das Szenario ist folgendes:

Wenn ein Konstrukteur ein Bauteil eines anderen Konstrukteurs als Grundlage für eine Variantenkonstruktion benutzt bzw. nur einige Verbesserungen vornimmt, sollen am Ende dennoch beide Dateien weiterhin dauerhaft abgespeichert werden. Problematisch wird es später bei der Unterscheidung beider Varianten. Mit bloßem Auge ist schwer nachvollziehbar welche Änderungen am Original unternommen worden sind und welche Verbesserungen diese mit sich bringen. Aus diesem Grund wurde von den Autoren ein Automatisierungsmechanismus entwickelt, der zwei CATPart-Dateien miteinander vergleicht und die Unterschiede anzeigt. Dafür wird als erstes automatisch von beiden Bauteilen eine Strukturbaumdatei im XML-Format erstellt, in der alle Konstruktionsschritte, die bei der Erstellung unternommen worden sind, abgespeichert werden. Die einzelnen Elemente, aus denen die Bauteile aufgebaut sind,

können durch die Angabe bestimmter Parameter vollständig bestimmt werden. Eine Variantenkonstruktion könnte folglich auch aus nur einer Änderung eines bestimmten Parameters bestehen.

Im nächsten Schritt werden beide Dateien miteinander verglichen. Dank des ebenenbasierten Aufbaus der Dateien im XML-Format, ist auch dieser Schritt mit Hilfe eines Computeralgorithmus möglich. Die Features, die die beiden Dateien unterscheiden, bestehen aus Änderungen, Beseitigungen und Neuerstellungen und werden dem Benutzer nach dem Vergleich angezeigt. Mit dieser Methode können also alle Unterschiede zwischen zwei Bauteilen innerhalb kürzester Zeit erkenntlich gemacht und die Dateien müssen nicht manuell untersucht werden.

Generell ist anzumerken, dass die meisten professionellen CAD-Systeme eigene Funktionen zur Erkennung von Features zur Verfügung stellen (vgl. Kapitel 2.1). Dies ist insbesondere von großer Hilfe, wenn systemfremde Dateien, wie z.B. die Austauschformate STEP oder IGES, aufbereitet werden sollen, so dass diese mit den softwareeigenen Features versehen und somit auch verändert werden können. Dennoch ist anzumerken, dass zu diesem Zeitpunkt kein Programm eine vollautomatisierte Erkennung liefert und somit der Anwender semiautomatisch die Bauteildateien bearbeiten muss.

Für eine Änderung der Geometrien von herstellerunabhängigen Formaten besteht allerdings zusätzlich noch die Möglichkeit einer sogenannten Direktbearbeitung. Dabei wird eine eindeutige Definition der Features umgangen, indem vom Benutzer Elemente, wie z.B. einzelne Flächen, im Bauteil angewählt werden, deren Eigenschaften, wie z.B. der Neigungswinkel oder die Größe, dann manuell zu verändern sind. Diese Vorgehensweise, bei der nicht direkt die Features identifiziert werden, aber dennoch bestimmte Bauteileigenschaften zugänglich sind, kann ebenfalls für eine folgende Eigenschaftsanalyse bzw. Ähnlichkeitssuche verwendet werden.

3.5 Product Lifecycle Management

Mit dem Ziel der Optimierung des Produktentwicklungsprozesses wurden in den letzten Jahren die klassischen Konstruktionsmethodiken in Frage gestellt bzw. überarbeitet. Die grundlegende Idee ist eine Einbeziehung des Lebenszyklus in die Entwicklungsphase eines neuen Produktes. Beispielsweise soll ein neuer Artikel schon so entworfen werden, dass sich die Fertigung möglichst kosteneffektiv gestalten lässt, was sich z.B. durch die Arbeitsteilung mit Fremdfirmen und den Zukauf von Teilen realisieren lässt. Außerdem soll besonderer Wert auf die Recyclingmöglichkeiten und die Trennbarkeit der verwendeten Materialien am Ende des Produktlebens gelegt werden.

Dieses neuartige Konzept wird als PLM oder auch als IPE (Integrierte Produktentwicklung) bezeichnet und ist Bestandteil des CAE. Aus nachvollziehbaren Gründen bieten die Ansätze des KBE zahlreiche Möglichkeiten die Anforderungen des PLM umzusetzen und sind auch für die Informationsextraktion in Kapitel 4 von Interesse. Im Folgenden sollen einige Beispiele vorgestellt werden, die beide Theorien miteinander verbinden.

Ähnlich wie in Kapitel 3.2 geht es in der Dissertation von Summers um die Arbeitsteilung und den Aufbau von Wissensdatenbanken [Summ-04]. Er entwickelt eine Methode für die Lösung verschiedener Designprobleme. Wenn eine Konstruktionsaufgabe während des Entwicklungsprozesses eines neuen Produktes gelöst werden soll, dann kann diese Problemstellung in das hier entwickelte System eingegeben werden. Der Autor nennt diese Art von Entwicklung CBD (Case Based Design). Alle Anforderungen und Randbedingungen müssen bei der Problemlösung betrachtet werden. Das Programm, das Summers entwickelt hat, analysiert die Informationen und vergleicht diese mit Problemlösungsvorschlägen, die in einer Datenbasis gespeichert sind. Im Idealfall ist die Datenbasis so umfangreich, dass für jedes neue Designproblem ein ähnlicher Fall herangezogen werden kann und dieser bei der Lösung des aktuellen Problems hilft. Der Schwerpunkt liegt folglich auf der Wiederverwendung und der Zugänglichkeit von Designwissen.

Einer ähnlichen Frage geht auch Yan in seiner Dissertation nach [YanW-04]. Er prüft inwieweit es möglich ist Designwissen zu sammeln und für kollaborative Zwecke wiederzuverwenden. Mit Designwissen ist das Know-How gemeint, das beispielsweise bei der Konstruktion von Bauteilen oder gesamten Baugruppen von den Ingenieuren angewendet wird.

Ein wichtiger Gesichtspunkt der Arbeit sind dabei die Austauschmöglichkeiten über verschiedene CAD-Systeme hinweg. Der Autor stellt dabei die Vor- und Nachteile der Dateiformate STEP und IGES vor. Da aber besonders die Möglichkeiten des

Internets genutzt werden sollen, benutzt Yan für die Entwicklung eines solchen Designsystems die Formatiersprache PML (Product Markup Language). Mit dieser können zwar nicht gesamte Bauteile abgespeichert werden, wohl aber spezielle Designaspekte. Die Dateien sind in dieser Form wie gerichtete Graphen aufgebaut und stellen so die geometrischen Randbedingungen z.B. in einer Baugruppe dar (vgl. Kapitel 3.9). Im Verlauf der Dissertation wendet der Autor seine Methode beispielhaft an einem Schmiedewerkzeug an. Auch bei einzelnen Bauteilen zeigt er, dass Designwissen bei der Implementierung von Features ebenfalls eine Rolle spielt. Bestimmte Features eines Bauteiles können folglich ebenso in einer PML-Datei untergebracht werden.

Bezogen auf kollaboratives Arbeiten fokussieren sich Zhao und Eskil beide auf die Arbeitsteilung von räumlich getrennten Teilnehmern. Zhao entwickelt in seiner Dissertation eine systematische Methode für gemeinsames Ingenieursdesign [Zhao-02]. Anwendung findet diese Vorgehensweise bei großen Projekten, die in verschiedene Sektionen eingeteilt werden müssen und so parallel zur gleichen Zeit erarbeitet werden. In der Praxis geschieht dies meist in verschiedenen Unternehmen. Zhao untersucht für die Entwicklung einer solchen Methode zuerst die Anforderungen, die in der Praxis erfüllt werden sollten. Hauptsächlich wird der Frage nachgegangen in welcher Art und Weise die Produkte designt bzw. entwickelt werden sollten, damit es später einfach ist die Arbeit aufzuteilen und separat zu erledigen.

Eskil erforscht die Möglichkeiten, die das Internet im Bezug auf ingenieurstechnische Kooperationen bietet [Eski-05]. Am Beispiel einer Brennstoffzelle verdeutlicht er seine Vorschläge und legt eine Möglichkeit der Zusammenarbeit räumlich getrennter Unternehmen dar.

Tay und Gu konzentrieren sich auf das Produktdesign, das mit PLM verknüpft werden kann [TayG-03]. Sie behandeln in ihrer Veröffentlichung eine Methode für evolutionäre Produktentwicklung auf ingenieurswissenschaftlicher Basis. Evolutionär bedeutet in diesem Fall, dass der Entwicklungsprozess in drei Phasen aufgeteilt wird. In der ersten - der Informationsgewinnungsphase - werden aus verschiedenen Quellen so viele Daten wie möglich gesammelt. Allerdings beginnt diese Phase noch vor der eigentlichen Konstruktion des neuen Bauteiles. Als Informationsquellen werden zum einen schon existierende Produkte untersucht und außerdem vorhandene Patente analysiert. Die Autoren gehen davon aus, dass Probleme, die vielleicht bei der eigenen Entwicklung entstehen, durch die Betrachtung ähnlicher Produkte gelöst werden können, da bereits produzierte Bauteile schon funktionsfähig sind und vielleicht sogar genau das gleiche Problem schon bei deren Entwicklung auftrat. Selbst wenn dies nicht der Fall ist, kann der Entwickler sich inspirieren lassen und findet so eventuell andere Wege bei seiner Produktenwicklung.

Die zweite Phase des evolutionären Produktdesigns ist die des Informationsmanagements. Die in der Analysephase gesammelten Informationen müssen z.B. in einer Datenbank gespeichert und abrufbar gehalten werden. Eine suchmaschinenartige Funktion wäre außerdem für die Zugänglichkeit hilfreich. An letzter Stelle steht die Informationswiederverwendungsphase. Selbstverständlich sollen die gefundenen Daten, wie oben beschrieben, an der richtigen Stelle wieder eingesetzt werden.

Am Beispiel eines häuslichen elektrostatischen Luftreinigers wird die gesamte Vorgehensweise in dem Artikel noch einmal durchgeführt. Nach dem Auseinanderbau und der Analyse wird ein neues Designkonzept mit verbesserten bzw. veränderten Eigenschaften vorgestellt. In diesem Fall konnte so viel Zeit, die beim Grundaufbau gespart wurde, in die Neukonzeption investiert werden.

Hsu et al. behandeln einen ähnlichen Ansatz [HCTT-10]. Sie analysieren mit Hilfe eines KBE-Assistenzsystems ein neu entwickeltes Produkt um so die optimale Reihenfolge der einzelnen Produktionsschritte herauszufinden. Auch dabei handelt es sich um eine Art von PLM, da nicht nur eine kostengünstige Produktion im Vordergrund steht, sondern auch zweitrangige Aspekte wie z.B. Montierbarkeit oder Umwelt in Betracht gezogen werden sollen. Die Autoren setzen für den Entwicklungsprozess auch Schwerpunkte auf Größe, Gewicht, Symmetrie oder Formfeatures der Produkte. Dies nennen sie DFA (Design for Assembly).

In der Dissertation von Burchardt werden einige von den hier vorgestellten Ansätzen wieder aufgegriffen [Burc-01]. Er entwirft außerdem ein erweitertes Konzept für die oben schon erwähnte integrierte Produktentwicklung, das Details über ein schwerpunktbezogenes Studium mit speziell darauf ausgelegten Vorlesungen und Übungen beinhaltet.

3.6 Knowledge Discovery und Data Mining

Um das in Kapitel 3.1 angesprochene Sammeln von Wissen vollständig darzustellen wird hier insbesondere auf die Entdeckung bzw. Aufdeckung von Wissen in großen Datenmengen durch KD bzw. DM eingegangen.

Die internen Techniken, die bei der Umsetzung in Form von Algorithmen während des DM ablaufen, können sehr unterschiedlich sein. Bei der Untersuchung großer Datenbestände, müssen Prioritäten gesetzt werden, damit der Suchprozess nach relevanten Informationen angemessen schnell ablaufen kann. Hilderman und Hamilton bezeichnen Klassifikation, Assoziation (Zuordnung), Clusterbildung (Gruppierung) und Korrelation (Verknüpfung) als die vier wichtigsten Techniken, die bei der Extraktion von Daten angewendet werden [HiHa-99]. Um festzustellen, ob die gefundenen Daten von Interesse sind, muss ihre Bedeutsamkeit bewertet werden. Dafür stellen Hilderman und Hamilton 17 verschiedene Maßnahmen sowie deren Vor- und Nachteile vor, die die Wichtigkeit bzw. die Relevanz der Daten analysieren (Interestingness Measures).

Im Gegensatz zu diesem Ansatz bezeichnen Frawley, Piatetsky-Shapiro und Matheus die drei Ziele des KDD als 'Summarization' (Zusammenfassung zu Klassen anhand der Analyse von charakteristischen und gemeinsamen Merkmalen), 'Discrimination' (ausreichende Unterscheidung der Eigenschaften, so dass klare Klassengrenzen gezogen werden können) und 'Comparison' (klare Beschreibung der Eigenschaften, so dass ein Vergleich einfach durchgeführt werden kann) [FrPM-92]. Sie verdeutlichen noch einmal, dass es für die Aufdeckung verschiedenartiger Informationen, die bei der Erfüllung dieser drei Voraussetzungen nötig sind, auch unterschiedliche Algorithmen bedarf bzw. dass die Faktoren die Auslegung der Algorithmen beeinflussen. Die zwei Veröffentlichungen von Romero und Ventura sowie von Köksal, Batmaz und Testik stellen zwei anschauliche Beispiele für genau diesen Sachverhalt dar [RoVe-06] [KöBT-11].

Romero und Ventura verbinden die Methoden des DM mit der Entwicklung von Ausbildungssoftware (Educational Systems). Wenn Schüler oder Studenten bestimmte Lernsysteme benutzen (E-Learning), dann sollen ihre Verhaltensweisen und alle Daten sowie Informationen in einer Datenbank abgespeichert werden. Die Ausbilder sollen Zugriff auf diese Daten haben und mit Hilfe des DM besonders interessante Informationen herausfiltern. So kann der Lehr- und Lernprozess immer wieder abgestimmt, korrigiert und verbessert werden.

Köksal, Batmaz und Testik untersuchen in ihrer Veröffentlichung die Möglichkeiten, die die Informationsgewinnung aus DM-Techniken im Bezug auf die Fertigungsoptimierung bieten und vergleichen die Vor- und Nachteile von verschiedenen wissen-

schaftlichen Arbeiten, die sich mit diesem Thema befassen. Der Schwerpunkt wird auf die Verbesserung der Qualität gelegt, die sich durch die Sammlung und Untersuchung möglichst vieler Informationen ergibt.

Im folgenden Abschnitt soll insbesondere auf den Aufbau von denjenigen Datenbanken eingegangen werden, die durch KD und DM untersucht werden. Bei der Entwicklung von wissensbasierten Systemen entsteht, wie schon in Kapitel 3.1 über Wissensakquisition beschrieben, immer die Frage wo das Expertenwissen herkommt und wie es zur Verfügung gestellt wird. Die folgenden Veröffentlichungen gehen sowohl von einem schrittweisen Aufbau einer Datenbank oder auch von einer festgelegten Datenbasis aus, die von Beginn an durch die Entwickler in das System implementiert wird. Bei beiden Fällen gelten jedoch die grundlegenden Regeln der Wissensaufdeckung bei der Untersuchung der Daten auf in der speziellen Situation relevante Informationen.

Die Veröffentlichung von Winkelmann bietet einen Ansatz für die Wissensaufdeckung in einer von Beginn an vollständigen Datenbasis, die die Rahmenbedingungen für eine Designdatenbank darlegt [Wink-10]. Ingenieure, die Produkte von Grund auf neu entwerfen, sollen von einem neuartigen objektorientierten Computerprogramm unterstützt werden, das Informationen, wie z.B. Normen und Richtlinien oder Standardelemente, wie beispielsweise Schrauben, im laufenden Prozess zur Verfügung stellt. Es soll eine softwaretechnische Verknüpfung zwischen einem Wissenskatalog und einer Arbeitsoberfläche erstellt werden, wobei der Konstrukteur zu jeder Zeit relevante Informationen abrufen kann. Die Wissensaufdeckung bzw. die Verarbeitung der gefundenen Informationen könnte sich an dieser Stelle als besonders problematisch herausstellen, da die Untersuchung und die Anzeige der Ergebnisse während des laufenden Konstruktionsprozesses geschehen müssen.

Hren hingegen beschreibt einen Ansatz für den schrittweisen Aufbau einer Datenbank, die das Expertenwissen enthält und somit als wissensbasiertes System bezeichnet werden kann [Hren-09]. Es handelt sich dabei um ein webbasiertes Programmierwerkzeug, das den Austausch von virtuellen Prototypen innerhalb von CAD-Systemen vorsieht. Aufgrund der einfach anwendbaren und praktischen Technologie, die das Internet zur Verfügung stellt, ist es so möglich von Neuentwicklungen nicht nur die Geometrie auszutauschen, sondern auch Konstruktionswissen, das zur Problemlösung beiträgt. Weiterhin soll eine XML-basierte Konfigurationsdatei, in der alle Produktvarianten gespeichert sind, einfache Änderungen erlauben. In dieser Weise muss nicht bei jeder Verbesserung oder Neukonstruktion eine neue Produktdatei abgespeichert werden und der benötigte Speicher bleibt so gering wie möglich. Zwischen- und Austauschformate, sollen während des gesamten Prozesses vermieden

werden, da bei der Konvertierung von CAD-Daten und bei der Benutzung von Zwischenformaten immer Informationen verloren gehen.

Für die Anwender wird es möglich sein auf fremde virtuelle Prototypen mit Hilfe einer Datenbasis zuzugreifen, um diese für eigene Interessen zu nutzen und außerdem Neukonstruktionen zur Bewertung und Evaluation für andere Benutzer zur Verfügung zu stellen. Ein umfassender Variantenreichtum bei geringem Verwaltungsaufwand wird so gewährleistet. Es liegt nahe, dass für diesen Ansatz der Algorithmus für die Wissensaufdeckung relativ spezialisiert sein muss, da das Wissen, das auf der Datenbasis abgespeichert ist schon relativ klar strukturiert ist und dementsprechend abgerufen werden kann. Die Internetverknüpfung bietet jedoch einen interessanten Ansatz zur parallelen Verarbeitung der Informationen, was auch zu einer Verteilung der Daten bzw. einer verstreuten Datenbasis führt.

Hancq, Walters und Beuth gehen ebenfalls von einer Datenbank aus, die erst durch die Anwendung schrittweise erstellt werden soll [HaWB-00]. Es handelt sich um eine objektorientierte Software, die bei der Vorhersage von Materialverhalten vor allem auf Ermüdungsaspekte im Zuge von Belastungen spezialisiert ist. Ein solches Computerprogramm soll es dem Anwender erlauben nicht nur einfache Aufgaben wie z.B. Analysen von konstanten Belastungen durchzuführen, sondern auch die Lebensdauer von Bauteilen vorherzusagen, die unter Temperatureinflüssen Mehrfachlasten mit unterschiedlichen Amplituden ausgesetzt sind.

Der objektorientierte Aspekt erlaubt anderen Anwendern zusätzlich das Programm auf Modulbasis noch auszubauen. Die Datenbank, die im Zuge dessen erstellt wird, soll alle wichtigen Kennwerte der gängigen Materialien enthalten, so dass der Benutzer bei der Dateneingabe automatisch unterstützt wird.

Eine Methodologie der Wissensaufdeckung und der Anwendung des DM im Bereich der Klassifikation stellen Moon, Simpson und Kumara vor [MoSK-10]. Sie benutzen die Fuzzy-Logik für die Clusterbildung bei der Untersuchung von Produktgruppen. Durch die Aufdeckung und die Untersuchung von den vorhandenen Produktdaten soll eine geeignete Datenbasis generiert werden, die für das Design von sich in Planung befindlichen Produktlinien hilfreich sein soll.

3.7 Reverse Engineering

Mit dem Begriff Reverse Engineering ist eine rückwärts gerichtete Ingenieurstätigkeit gemeint, bei der der Versuch unternommen wird mit wenig bzw. unvollständigen Informationen ein fertiges Projekt so realitätsgetreu wie möglich zu rekonstruieren. Da für eine solche Rekonstruktion auch Informationen aus bestimmten Datenbeständen extrahiert werden müssen, sind die hier beschriebenen Ansätze im Hinblick auf die Eigenentwicklung in Kapitel 4.4 von großem Interesse.

Als Beispiel soll an dieser Stelle eine Sammlung von zweidimensionalen technischen Zeichnungen genannt werden, die in einem Unternehmen eventuell noch nicht als CAD-Modell vorliegen und möglichst automatisiert erstellt werden sollen. Nagasamy und Langrana stellen fest, dass bei einer solchen Rekonstruktion anhand von Bildern, Fotos oder Zeichnungen meistens die folgenden vier Schritte durchgeführt werden [NaLa-91]:

1. Abbildungsanalyse und Transformation von zwei- zu dreidimensionalen Vertices
2. Erstellung von Kantenverbindungen zwischen diesen dreidimensionalen Knoten
3. Zweidimensionale Flächenerstellung im dreidimensionalen Raum
4. Verbund der Flächen zu dreidimensionalen Objekten

In ihrer Veröffentlichung wenden Nagasamy und Langrana ebenfalls diesen Grundablauf an und benutzen dabei eine wissensbasierte Umgebung.

Ein wichtiges Werkzeug, das für die Abbildungsanalyse vonnöten ist, ist die automatische Objekterkennung in Bildern oder Zeichnungen. So wie im ersten Punkt der Aufzählung erwähnt, müssen Formen und Konturen nicht nur erkannt und analysiert, sondern auch interpretiert werden, damit die Transformation der zweidimensionalen Abbildung in den dreidimensionalen Raum geschehen kann. Als Beispiel für die Detektion von Kreisen, die auf technischen Zeichnungen und in Skizzierprogrammen ein geometrisches Primitiv bilden, soll an dieser Stelle die Veröffentlichung von Cuevas und Gonzáles genannt werden, die auch den Fall von ineinander geschobenen Kreisen berücksichtigt [CuGo-12].

Eine solche automatische Erkennung von einfachen skizzierten Formen liefert auch die Methode von Shtof et al. [SAGS-13]. Sie trennen die einzelnen Schritte, die das Programm von der Skizze zum Modell durchläuft in zwei Aufgabengebiete. Die sogenannten semantischen Aufgaben übernimmt der menschliche Anwender des Programmes. Dieser liefert eine Skizze und definiert einfache geometrische Figuren, die darin enthalten sind. Die Aufgabe der Software ist es die gegebenen Inputparameter zu analysieren und zu einem dreidimensionalen Modell zu verbinden. Das Programm

übernimmt dabei gewissermaßen den Zusammenbau der einfachen Figuren zu einem komplexeren Modell, das am Ende in Form von zweidimensionalen Flächen im dreidimensionalen Raum (Freiformflächen) vorliegt.

Eine sehr ähnliche Aufgabenstellung mit jedoch vollkommen unterschiedlicher Vorgehensweise beinhaltet der Ansatz von Takeuchi, Watanabe und Yamakawa [TaWY-11]. Auch in diesem liefert der Anwender des Programmes eine bzw. mehrere Skizzen und muss nach der Eingabe noch weitere Zusatzinformationen definieren. Das Programm setzt aus den Skizzen einen rechteckigen Block zusammen, der die skizzierten Silhouetten enthält und von dem vom Benutzer noch Material an den richtigen Stellen entfernt werden muss. Ähnlich wie bei einer Skulptur, deren Form aus einem einzigen Block durch Materialabtragung erschaffen wird, muss der Anwender die Stellen markieren, die abgetragen werden sollen. Am Ende entsteht ein dreidimensionales Modell, das noch immer die Silhouetten der anfangs erstellten Skizzen beinhaltet.

Eine weitere Methode zur automatischen Skizzenerkennung beschreiben Roth-Koch und Westkaemper [KoWe-10]. Im Gegensatz zu den anderen genannten Ansätzen konzentrieren sie sich dabei auf die digitalisierte VPE mit dem Hintergrund, dass aus den Skizzen ein prototypenhaftes CAD-Modell entstehen soll, das bei der technischen Ausarbeitung zum fertigen Produkt von großer Hilfe sein kann. Die Autoren gehen dabei insbesondere auf die Objekt- bzw. Konturenerkennung auf den zweidimensionalen Skizzen ein, wobei eine Interpretation, Visualisierung, Berechnung und Auswertung stattfinden muss, um die Übertragung in den dreidimensionalen Raum zu ermöglichen. Die fortführende Übertragung des entstehenden dreidimensionalen Objektes hin zu einem brauchbaren CAD-Modell erfolgt mit Hilfe einer Oberflächenanalyse und Definition von Punktwolken, die durch Interpolation von vielen Punkten auf der Oberfläche die endgültige Form des CAD-Prototypen annähern.

Der vorgeschlagene Ablauf von Jang und Ho hat ein ähnliches Ziel [JaHo-10]. Von einem Objekt, das dreidimensional digitalisiert werden soll, werden nicht etwa Skizzen als Eingabeparameter benutzt, sondern es müssen aus mehreren Kameraperspektiven Bilder aufgenommen werden, die bei der virtuellen Rekonstruktion miteinander in Einklang gebracht werden, um so das ggf. sehr komplexe Objekt so exakt wie möglich anzunähern. An einem Beispiel wird die Effizienz von zwei gegebenen Algorithmen getestet. Der erste rekonstruiert das Objekt anhand einer Silhouettenidentifikation, die sozusagen nur die 'Schatten' der aufgenommenen Bilder analysiert und aufgrund der Aufnahme aus mehreren Perspektiven das Objekt in dreidimensionaler Form nachgestalten kann.

Der zweite untersuchte Algorithmus wird mit dem Wort Voxelfärbung beschrieben. Ein Voxel ist das dreidimensionale Equivalent eines Pixels und besitzt immer

eine würfelförmige Erscheinung. Durch die Anordnung von vielen Voxeln, die zusammenhängen, können dreidimensionale Figuren mit einer gestuften Oberfläche angenähert werden. Allerdings gilt ähnlich wie bei den Pixeln der gleiche Grundsatz. Um eine feinere Rekonstruktion erreichen zu können, müssen mehr Voxel mit kleinerer Kantenlänge verwendet werden. Im Algorithmus wird die Oberfläche eines Objektes detektiert, indem wiederum zwei Bilder des Objektes aus unterschiedlichen Winkeln aufgenommen und die Farben von stichprobenartigen Punkten auf der Oberfläche analysiert werden. Wenn in beiden Bildern ein untersuchter Punkt bzw. ein Voxel die gleiche Farbe besitzt, so muss dieser zur Oberfläche des Objektes gehören. Durch die Untersuchung von vielen Probepunkten der Oberfläche entsteht ähnlich wie in der beschriebenen Veröffentlichung von Roth-Koch und Westkaemper eine Punktewolke, die die Form des Objektes rekonstruiert.

Jang und Ho verbinden die beiden Ideen der erwähnten Algorithmen zu einer Methodik, durch die sich auch die Oberflächen komplexer Formen durch eine voxelbasierte Rekonstruktion darstellen lassen.

Lee und Fang verfolgen das gleiche Prinzip [LeFa-12]. Von zwei bereits erprobten Methoden ('Cubic Corner Method' und 'Optimisation-Based Method') bilden sie einen Hybriden, um die Vorteile in einer Vorgehensweise zu vereinen. Das besondere ist an dieser Stelle die mögliche Anwendung auf Formen, die durchaus im Maschinenbau alltäglich sind.

3.8 Fingerprints und Signaturen

Wie schon in Kapitel 1.3 angesprochen, ist es bei einem Vergleich von zwei komplexen Systemen von großem Vorteil nicht alle Feinheiten, sondern nur einen Extrakt, der die wichtigsten Eigenschaften enthält, zu betrachten. Die Idee ist es ein solches komplexes System soweit auf die wesentlichen Merkmale zu reduzieren, dass eine eindeutige Identifikation noch immer möglich ist und damit ein Vergleich mit anderen Systemen einfacher ablaufen kann. Ein Beispiel zur eindeutigen Identifikation sind der menschliche Fingerabdruck und die DNS, dessen Konzepte sich auch auf Bauteile oder Baugruppen übertragen lassen. Es besteht jedoch folgender Unterschied: Während der Fingerabdruck nur ein eindeutiges Identifikationsmerkmal darstellt, beinhaltet die DNS außerdem noch Merkmale bzw. verschlüsselte Informationen, die Rückschlüsse auf die Eigenschaften zulassen.

Bei der Anwendung der Grundidee auf den Vergleich von ingenieurstechnischen Bauteilen, bleibt das Konzept bestehen. Eine sogenannte Signatur (Fingerprint) soll in verkürzter Form die wichtigsten Elemente eines Bauteils enthalten, was bei mäßig bis wenig ähnlichen Bauteilen zu einer großen Ähnlichkeit der Signatur führt und somit bei der Zusammenfassung zu Bauteilgruppen bzw. Clustern von großer Hilfe ist.

Eine besondere Form der Signaturen sind sogenannte Histogramme, die wie in folgendem Beispiel von Ashbrook et al. z.B. bei der Objekterkennung in Bildern angewendet werden [ATRB-95]. Ein Algorithmus extrahiert dabei alle Elemente, die in einem Bild enthalten sind und misst die jeweiligen Abstände untereinander. In einem PGH (Pairwise Geometric Histogram) werden diese Informationen abgespeichert und können bei der Wiederauffindung von skalierten Objekten hilfreich sein.

Auch in der Veröffentlichung von Osada et al. werden derartige Histogramme angewendet, um aus vielen verschiedenen dreidimensionalen Polygonmodellen Objektklassen zu bilden [OFCD-01]. Neben der Untersuchung von verschiedenen Vorgehensweisen konzentrieren sich Osada et al. dabei auf die folgende Methode: Bei der Analyse eines in der Datenbasis enthaltenen Objektes wird die Distanz zweier Punkte gemessen, die beliebig gewählt werden, sich allerdings auf der Oberfläche des Objektes befinden. Dieser Vorgang wird mit jedem Objekt viele Male durchgeführt und die Ergebnisse werden in einem Histogramm, das auf der Abszisse die Distanzen und auf der Ordinate die Wahrscheinlichkeitsverteilung dieser Distanzen beinhaltet, festgehalten. Je nachdem welche äußere Erscheinung das untersuchte Objekt besitzt, entsteht ein charakteristisches Histogramm. Osada et al. gelangen zu dem Ergebnis, dass die Histogramme, die auch für den Vergleich mehrere Teile benutzt werden, bei sich gleichenden Objektgruppen auch große Ähnlichkeiten aufweisen, so dass ein aussagekräftiges Mittel zum Vergleich zweier Objekte entstanden ist. Abbildung 3.2 zeigt nach

Osada et al. beispielhaft sowohl die Untersuchung (a) als auch das Ergebnis (b) eines zweidimensionalen Dreiecks.

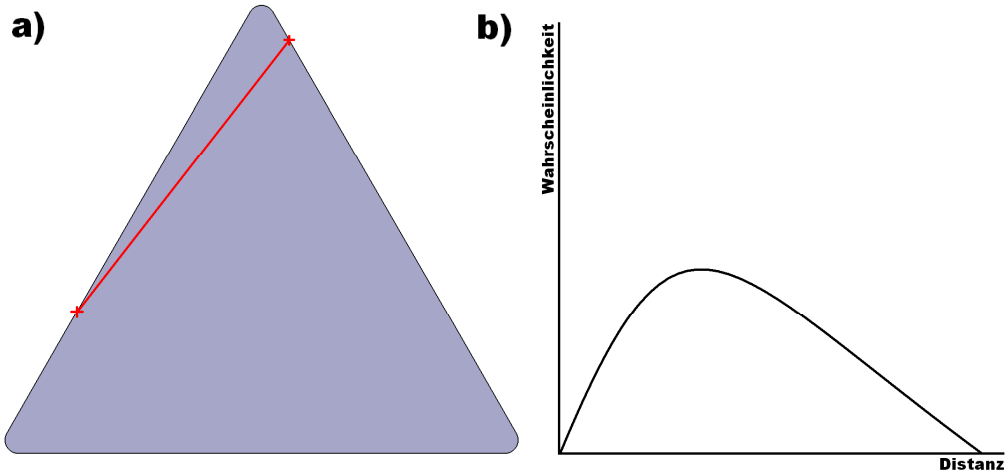

Abbildung 3.2: Untersuchung eines zweidimensionalen Beispielteils

Einen Histogrammansatz verfolgen auch Körtgen et al. [KPNK-03]. Bei der Fingerprinterstellung von dreidimensionalen Objekten stellen sie ein Schalenmodell und ein Sektorenmodell zum Histogrammaufbau vor. Beide Ansätze sind in Abbildung 3.3 anhand des dreieckigen Beispielteils zweidimensional abgebildet. Die Grundidee bleibt bei beiden Möglichkeiten bestehen. Das Schalenmodell (a) ist sphärenförmig aufgebaut und enthält das zu untersuchende Objekt, so dass der Schwerpunkt des Teiles mit dem der Kugel übereinstimmt. Unabhängig von der Orientierung im Raum wird dann untersucht in welchen Schalen bzw. in welchen Sphären Material des Objektes vorhanden ist. Beim Ansatz des Sektorenmodells (b) wird ein dreidimensionales Raster über das geometrische Objekt gelegt, wobei auch dort untersucht wird, in welchen Zonen Material bzw. kein Material existiert.

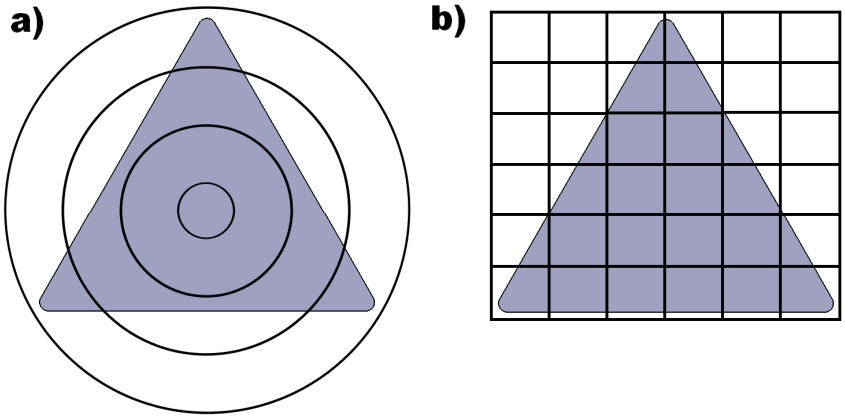

Abbildung 3.3: Schalen- und Sektorenmodell des Beispielteils

3.9 Graphen und Matrizen

Vergleichbar mit der mathematischen Megenlehre widmet sich die Graphentheorie ebenfalls mit der Darstellung von Strukturen und ihren Relationen zueinander. Dabei ist zwischen gerichteten und ungerichteten Graphen zu unterscheiden, womit beispielsweise Hierarchien dargestellt werden können (vgl. Kapitel 5.2). Die Aufteilung in die verschiedenen Hierarchieebenen findet sich etwa in den bereits erwähnten Spezifikationsbäumen der Konstruktionsprogramme wieder, in denen die Arbeitsschritte des Anwenders und somit auch die Features und Unterfeatures der Produkte spezifiziert sind. Der Aufbau geometrischer Formen in dreidimensionale Elemente, die wiederum aus zweidimensionalen Skizzen und z.B. Extrusionen, Drehungen oder Spiegelungen bestehen, ist ein typisches Beispiel für eine Rangfolge in Stufenform.

Für den Aufbau und die Definition von Bäumen, die bestimmte Strukturen widerspiegeln, werden zwei verschiedene Techniken namens Top-Down- und Bottom-Up-Vorgehen benutzt. Geoken beschreibt diese Techniken und die Beziehungen der Knoten in den einzelnen Ebenen untereinander mit englischen Fachausdrücken [Goek-06]:

> *Beim Top-Down-Vorgehen werden - ausgehend von dem Top-Hierarchieknoten - Has-Subset-Beziehungen konstruiert, bis eine dem Benutzerbedarf angemessene Detailebene erreicht ist (diese stellen die Hierarchieelemente dar). Beim umgekehrten Bottom-Up-Vorgehen werden Hierarchieelemente gemäß Mitgliedschaftsbedingungen über Instance-of- bzw. Subset-of Beziehungen bis zur obersten Ebene (Top-Hierarchieebene) abstrahiert.*

Die Knoten auf den Subebenen werden - ebenso wie in den XML-Dateien auch - als Kinder betitelt, während Knoten auf der nächsthöheren Ebene als Elternknoten bezeichnet werden. Wenn ein Strukturbaum in der höchsten Ebene zu einem Knoten zusammenläuft, wird dieser auch Wurzel genannt.

Beispiele für gerichtete oder auch ungerichtete Graphen sind Mindmaps (vgl. Abbildung 3.1) oder Petrinetze, die vor allem Objekte oder Gegebenheiten in ihre Bestandteile aufteilen und die Relationen untereinander darstellen.

Lee und Kim stellen eine Methode vor, die bei dem Gebrauch von parametrischer Konstruktion die Berechnungsdauer des geänderten CAD-Modells beschleunigt [LeKi-96]. Dabei werden die Vorteile einer Darstellung des CAD-Modelles als gerichtete Graphen benutzt. Wenn der Benutzer einen Parameter ändert und dessen Änderung sich auf viele andere Parameter auswirkt, dann wird dies als Inferenz bezeichnet und die Berechnung bzw. das Update des veränderten Modells kann sehr lange dauern.

Die Knoten des Graphen stellen bei dieser Methode die geometrischen Einträge bzw. die Features dar, während die Verbindungen der Knoten die Randbedingungen

oder die Restriktionen beschreiben. Zur Lösung des Problems werden zwei verschiedene Algorithmen vorgestellt, die die Zugänglichkeit komplizierter Graphen bzw. Bauteile beschleunigen und die gewünschten Ergebnisse liefern.

Auch der Ansatz von Peabody und Regli beschreibt eine Umwandlung eines CAD-Bauteiles in eine graphenförmige Struktur [PeRe-01]. Ausgehend vom Modell (Solid Model) wird ein MSG (Model Signature Graph) erstellt, der die Bauteilinformationen beschreibt und auch weitere Attribute den einzelnen Bestandteilen hinzufügt. Über einen sogenannte ITV (Invariant Topology Vector) werden die folgenden fünf Eigenschaften beschrieben, die die Vergleichbarkeit von zwei Graphen bzw. Modellen unterstützen:

1. Die Bestimmung der Anzahl an Knoten und Kanten
2. Die beiden Flächen mit den meisten und den wenigsten Verbindungen
3. Die gängigsten Verbindungsarten
4. Der Durchmesser bzw. der längste Pfad des Graphen
5. Ein Typenhistogramm, das das statistische Vorkommen der Flächen und Kanten analysiert

Der ITV wird von Peabody und Regli als eine Art Referenz verwendet, die es erlaubt die Ähnlichkeit zweier Graphen auf den ersten Blick festzustellen.

Im Gegensatz zu einer solchen Darstellung in ungerichteter Form zeigen Ding, Yu und Liu eine hierarchische Vorgehensweise anhand einer Bauteilzerlegung in die einzelnen Features [DiYl-14]. Dabei gehen sie auf verschiedene Graphentypen ein, die die einzelnen Bestandteile der Modelle enthalten. An dieser Stelle werden sowohl FDG (Feature Dependency Graph), als auch PAG (Property Adjacency Graph) erwähnt, die beide hierarchisch angeordnet sind und zum einen die Features bzw. Featurerelationen, und zum anderen die Eigenschaften, wie z.B. Oberflächenbeschaffenheiten, die für die Fertigung von Interesse sind, beinhalten. Ding, Yu und Liu leiten aus diesen gegebenen Informationen einen HG (Hierarchical Graph) ab, der beide Ansätze miteinander vereint und so die Vorteile beibehält.

Eine alternative Darstellungsmöglichkeit hierarchischer Strukturen ist die Adjazenzmatrix. In den gängigen CAE-Programmen, wie z.B. CATIA, wird diese verwendet um die Einbauverhältnisse der einzelnen Teile einer Baugruppe zu veranschaulichen. Eine solche Matrix beinhaltet sowohl in den Zeilen-, als auch in den Spaltenüberschriften alle verbauten Elemente. Die eigentliche Matrix ist symmetrisch und muss nur mit Kennzahlen - insbesondere Einsen und Nullen - gefüllt werden, die anzeigen ob bei einem Bauteil eine direkte Verbindung mit jeweils allen anderen Teilen der Baugruppe besteht oder nicht. Da ein Bauteil nicht mit sich selber benachbart sein kann, bilden alle nicht definierbaren Einträge die Diagonale bzw. die Spiegelachse.

Eine Mischung des graphen- und matrixbasierten Ansatzes stellen Vinodh, Kumar und Nachiappan vor [ViKN-12]. Die Wiederverwendungs- bzw. Recyclingmöglichkeiten werden am Beispiel eines Drehschalters, dessen Bestandteile in eine Hierarchie zerlegt werden, aufgezeigt. Neben einer Darstellung des Zerlegungsplans in Form von AND/OR-Graphen, Verbindungsgraphen und gerichteten Graphen, werden auch DPN (Disassembly Petri Net) vorgestellt, die die Kosten einer Zerlegung berücksichtigen und so bei der Entscheidungsfindung für oder gegen einen eventuellen Recyclingprozess hilfreich sind.

3.10 Ähnlichkeitssuche

Neben einer Anwendung im ingenieurstechnischen Umfeld sind Methodiken, die zwei komplexe dreidimensionale Formen miteinander vergleichen und einen Kennwert liefern, der die Ähnlichkeit beschreibt, auch in anderen Forschungszweigen, wie z.B. der Medizin oder der Chemie, von großem Interesse (vgl. Kapitel 6.1). Einen Überblick über den Stand der Forschung zum Thema dreidimensionale Formensuche bieten Bustos et al. Sie untersuchen verschiedene Ansätze zur dreidimensionalen Ähnlichkeitssuche und konzentrieren sich in einer Umfrage auf die folgenden Methoden [BKSS-05]:

1. Statistische 3D-Deskriptoren
2. Erweiterungsbasierte Verfahren
3. Volumenbasierte Verfahren
4. Oberflächengeometrie
5. Bildanalyse
6. Featurebefreiter Vektorenvergleich

Wie bereits in Kapitel 3.8 beschrieben, werden bei der geometrischen Ähnlichkeitssuche die grundlegenden Eigenschaften eines Bauteiles überwiegend in extrahierter Form repräsentiert. In den weiteren hier beschriebenen Forschungsansätzen werden dafür meist Signaturen verwendet, die jedoch in unterschiedlicher Art und Weise aufgebaut sind.

Die 'Princeton Shape Retrieval and Analysis Group' beispielsweise forscht mit verschiedenen Ansätzen an der Ähnlichkeitssuche von alltäglichen dreidimensionalen Objekten, die in einer großen Datenbank zur Verfügung gestellt werden [Prin-15]. Von besonderem Interesse ist die Dissertation von Min, die eine Suchmaschine beschreibt, die in der Lage ist auf Basis von Handskizzen, die der Anwender dem System liefert, Modelle in dieser Datenbank aufzufinden [Minp-04]. Neben den erwähnten Themen, wie z.B. Histogrammerstellung, Modellrepräsentation und Teilevergleich werden abgesehen davon auch Ansätze über Internetsuche oder Formerkennung behandelt. Min hat für die praktische Anwendung eine Software entwickelt, die es dem Anwender erlaubt in einem GUI eine Skizze zu erstellen, die nach einer Bestätigung automatisch analysiert wird und aus der Datenbasis passende Objekte heraussucht. Min kommt zu dem Schluss, dass eine Formensuche basierend auf zwei- oder dreidimensionalen Eingabeparametern der konventionellen Textsuche bei weitem überlegen ist und von den Anwendern eine solche einfache Eingabe schneller akzeptiert wird.

Die Veröffentlichungen von Wang, He und Tian sowie Wei und Yuanjun bedienen beide einen technischeren Ansatz zur Ähnlichkeitssuche [WaHT-07] [WeYu-08]. In beiden werden herkömmliche CAD-Modelle zuerst in voxelisierte Vergleichsmodelle

überführt, so dass ein Histogramm in einfacher Art und Weise im zweiten Schritt abgeleitet werden kann.

Wang, He und Tian evaluieren dabei zwei verschiedene Vorgehensweisen. In der ersten transformieren sie ein CAD-Modell, das als Gitternetz vorliegt, in ein Voxelmodell, das sie mit der in Kapitel 3.8 beschriebenen Methode, bei der ein Histogramm anhand der Distanzen zweier beliebiger Punkte auf der Oberfläche erstellt wird, untersuchen. Die Punkte werden dabei durch die Voxelzentren ersetzt und so ihre Anzahl im Vergleich mit dem Netzmodell drastisch reduziert.

In der zweiten angesprochenen Technik erfolgt ein Umweg über ein von den Autoren als 'Skelettextraktion' bezeichnetes Verfahren. Die Idee ist es vor der eigentlichen Untersuchung das Voxelmodell zuerst auszudünnen, so dass jegliches Material entfernt wird, das zum grundsätzlichen Erscheinungsbild des ursprünglichen CAD-Teiles keine Auffälligkeiten beisteuert. Das Ergebnis ist praktischerweise am Beispiel eines Schraubenschlüssels zu erklären, der an der einen Seite mit einem Maulschlüssel und auf der anderen Seite mit einem Auge ausgestattet ist. Nach der Verwandlung seines Voxelmodells ins Skelettmodell ist die ursprüngliche Grundform beider Enden noch immer ausgeprägt vorhanden, während der eigentliche Griff ausgedünnt nur noch aus einer einzigen Reihe von Voxeln besteht.

Die Autoren benutzen zur eigentlichen Ähnlichkeitssuche zum einen das genannte Histogramm und außerdem einen Nachbarschaftsgraphen, der aus dem ausgedünnten Skelettmodell hervorgeht.

Wei und Yuanjun vergleichen bei der Übertragung vom CAD- zum Voxelmodell zu anfangs verschiedene gegebene Algorithmen, die ähnliche interne Abläufe bei der Erstellung verfolgen. Die Ergebnisse werden dann weiter in Segmente unterteilt, in denen die enthaltenen Voxelmengen gezählt und ausgewertet werden, so dass am Ende ebenfalls eine Art Histogramm entsteht, das die Autoren als 'Feature Vektor' bezeichnen und das als Bauteilrepräsentation bei der Ähnlichkeitssuche genutzt werden kann.

Ein komplexerer Ansatz wird von Heczko et al. verfolgt [HKSV-01]. Hierbei handelt es sich um eine inhaltsbasierte Suche, bei der nicht an die Modelle angehängte Eigenschaften im Zuge des Ähnlichkeitsvergleiches analysiert (textbasierte Suche), sondern Merkmalsvektoren, die die Form der Körper beschreiben (vgl. Kapitel 3.8), automatisch erstellt werden. Somit entsteht eine Unabhängigkeit gegenüber der Rotation bzw. Translation der Geometrien und sogar eine unterschiedliche Skalierung kann vernachlässigt werden. Zwar handelt es sich bei der vorgestellten Testmenge auch um Modelle im VRML-Format, allerdings wird kein Bezug auf ingenieurstechnische Anwendungen genommen.

Die Merkmalsvektoren sind mehrdimensional und werden mit Hilfe verschiedener mathematischer Methoden auf der immer gleichen Art und Weise erstellt. Wie in Kapitel 3.8 beschrieben, wird auch hier ein zweidimensionales Sektorenmodell (Segmente) angewendet, das einen Teil des Gestaltmerkmalsvektors bildet. Neben einer solchen Analyse im zweidimensionalen Bereich, wird als weiterer Bestandteil des Merkmalvektors auch ein Strahlenmodell angewendet, bei dem automatisch eine virtuelle Kugel um die Hülle des Modells gelegt wird, zwischen deren Zentrum und der Hülle verschiedene Zeiger konstruiert werden, die in ihrer Gesamtheit einen charakteristischen Bestandteil des Merkmalsvektors bilden.

Heczko et al. kommen zu dem Schluss, dass die von ihnen entwickelte Methode für die Ähnlichkeitssuche von 3D-Modellen vielversprechende Ergebnisse liefert. Dennoch sind im Bezug auf eine sehr große Testmenge noch weitere Sonderfälle an komplexeren VRML-Dateien durchzuführen. Insbesondere ist auch eine geeignete Kombination zwischen mehreren Merkmalsvektoren zu untersuchen. Die Autoren gehen davon aus, dass ideale Ergebnisse durch eine solche Verknüpfung von Eigenschaften herbeigeführt werden können, die jedoch nicht so einfach vorhersagbar sind.

3.11 Clustering und Klassifikation

Nach der in Kapitel 3.10 angesprochenen Ähnlichkeitssuche, die es erlaubt ein Gemeinsamkeitsmaß zwischen zwei Bauteilen automatisch zu erstellen, ist die logische Konsequenz eine Sortierung bzw. Aufteilung aller Teile in einer Datenbank in Cluster (vgl. Kapitel 6.2). Dies führt zu einer besseren Übersicht und erlaubt es dem Anwender einen Überblick über die Gesamtheit aller Bauteile zu erhalten.

Einer der gängigsten Algorithmen, die für die Klassifikation verwendet werden, ist der sogenannte 'K-Means'-Algorithmus. Michalik, Štofa und Zolotová prüfen seine Konsistenz im Zuge einer Anwendung sowohl im technischen Bereich, als auch in anderen DM Applikationen, wie z.B. Kundensegmentierung, Social Media-Analyse oder diversen Problematiken aus dem Bereich der Medizin [MiŠZ-13]. Die Idee ist es zu Beginn ein Datenset mit vielen unsortierten Einträgen zur Verfügung zu stellen und eine Anzahl an Klassen festzulegen (Inputparameter). Der eigentliche Algorithmus läuft iterativ ab, weist die Einträge der Datenbasis den Clustern zu und verschiebt die Klassengrenzen so lange, bis eine Konvergenz erreicht ist und die Einträge in den Klassen nicht mehr verändert werden.

Shelza und Singh wenden den 'K-Means'-Algorithmus zur Klassifikation von CAD-Abbildungen an und vergleichen die Ergebnisse mit einem weiteren gegebenen hierarchischen Clusteralgorithmus sowie ihrer eigenen entwickelten Methode [ShSi-12]. Auch Ansary, Daoudi und Vandeborre werten die Ergebnisse des 'K-Means'-Algorithmus aus [AnDV-07]. Anhand der öffentlichen in Kapitel 3.10 erwähnten Datenbank der 'Princeton Shape Retrieval and Analysis Group' evaluieren sie die von ihnen vorgeschlagene Methode zur Auswahl von geeigneten zweidimensionalen Ansichten dreidimensionaler Teile. Die Idee ist es das dreidimensionale Objekt zuerst zu analysieren, um dann ein möglichst kleines Set aus zweidimensionalen Ansichten auszuwählen, das eine genaue Charakterisierung des Teiles zulässt. Sie kommen zu dem Schluss, dass ein großes Gefälle zwischen komplexen und einfachen Objekten besteht, wobei für die Repräsentation von komplizierten Modellen eine größere Anzahl von Ansichten nötig ist.

Ausgehend von den verschiedenen bereits erwähnten Möglichkeiten die Erscheinung eines Bauteiles in extrahierter Form darzustellen, testen Jayanti, Kalyanaraman und Ramani zwei verschiedene Methoden zur Klassifikation von CAD-Bauteilen [JaKR-09]. In der ersten wird ebenfalls die Idee des 'K-Means'-Algorithmus verwendet, wobei die Distanzen der einzelnen Entitäten zuerst in einen von den Autoren als mehrdimensionaler 'Featureraum' bezeichneter Übergangsparameter überführt werden. Die zweite Vorgehensweise konzentriert sich unmittelbar auf einen distanzbasierten Clusteralgorithmus. Am Ende wird die Effizienz beider Methodiken ausgewertet

und die Autoren kommen zu dem Schluss, dass für unterschiedliche Formrepräsentationen diverse Clusteralgorithmen von Vorteil sind.

Einen weiteren Ansatz zur Klassifikation von CAD-Modellen, der jedoch nicht vollautomatisch, sondern unter Zuhilfenahme von Benutzereingaben abläuft, stellen Hou, Lou und Ramani vor [HoLR-05]. Dabei werden die Klassen als erstes durch den Benutzer anhand weniger Teile spezifiziert und dem System somit die Klassengrenzen dargelegt. Im nächsten Schritt erfolgt eine Skalierung auf große Datenmengen, die der Algorithmus dann automatisch analysiert und eventuell aufgefundene Teile in die zuvor definierten Klassen einsortiert.

Ein letzter Ansatz, der sich von den bisher vorgestellten Methoden unterscheidet, wird von Moon, Kumara und Simpson untersucht [MoKS-06]. Bezogen auf die Entwicklung von Teilefamilien in der ingenieurstechnischen Konstruktion von neuen Produkten wird eine Clustermethode vorgestellt, die sich den Grundsätzen des DM unter Zuhilfenahme der Fuzzy-Logik bedient und so einen modulbasierten Ansatz bei der Herstellung von neuen Produkten bildet.

Intermezzo

Die in den Kapiteln 2 und 3 beschriebenen Industrieanwendungen und Forschungsansätze, lassen erkennen, dass zurzeit noch eine große Diskrepanz zwischen den marktreifen, sich im Gebrauch befindlichen Softwaresystemen und den theoretischeren Ansätzen aus der Wissenschaft herrscht. So wenden Unternehmen, die große CAD-Datenbanken erstellen und verwalten müssen, meist PLM-Systeme an, die die Anwender beim Umgang mit den großen Datenmengen unterstützen und gewünschte Informationen mit geringem Suchaufwand relativ schnell zur Verfügung stellen können. Ein wichtiger Aspekt, der jedoch beachtet werden muss, ist der Aufwand, der im Vorfeld beim Aufbau eines solchen PLM-Systems betrieben werden muss. Die meisten Daten, die jeder CAD-Datei zugeordnet sind und von der Software abgerufen werden können, müssen nach wie vor textbasiert bei der Erstellung eines jeden CAD-Modells manuell eingepflegt werden.

Ein Forschungsziel dieser Arbeit ergibt sich somit aus der automatisierten Informationsextraktion. In Kapitel 4 wird untersucht, welche Möglichkeiten insbesondere die herstellerunabhängigen (vgl. Kapitel 2.2), aber auch die proprietären Dateiformate (vgl. Kapitel 2.1.2), im Hinblick auf die Charakterisierung der Bauteile bieten. Die Informationen, auf die die PLM-Systeme zugreifen, sollen folglich in Zukunft vollautomatisiert in die Software eingespeist werden können und gleichzeitig so umfangreich wie möglich sein.

Die Auswertung der PDM-Systeme in Kapitel 2.8 hat gezeigt, dass einerseits bereits funktionierende Ansätze bestehen, diese andererseits aber noch nicht gänzlich ausgereift sind. Einer dieser Ansätze ist die geometrische Ähnlichkeitssuche. Insbesondere die Erkennung der Similaritäten und der Bauteilvergleich spielen für die automatisierte Suche nach charakteristischen Bauteileigenschaften eine wichtige Rolle. Wie Kapitel 3.8 zeigt, ist die gängigste Vorgehensweise die Bildung einer repräsentativen Signatur bzw. die Erstellung eines Fingerprints, um später als Vergleichsmaß dienen zu können. Ein ähnlicher Ansatz wird in Kapitel 5 im Zuge der Informationsaufbereitung bedient. Es wird untersucht, welche Möglichkeiten zur Signaturerstellung sich aus den gesammelten Informationen ergeben.

Ebenso wie die Ausführungen zur Ähnlichkeitssuche (vgl. Kapitel 3.10) und zur Klassifikation (vgl. Kapitel 3.11) ist der Teil der Informationsweiterverarbeitung in einen Suchmaschinen- und einen Clusterbereich aufgeteilt. Im Speziellen wird in Kapitel 6 auf neuartige Herangehensweisen eingegangen, die in ähnlicher Form zwar Thema in der Wissenschaft sind, aber die die untersuchten PDM-Systeme noch nicht bieten bzw. die dort noch nicht ausgereift sind.

Zusammenfassend ergibt sich somit in den folgenden fünf Punkten die Forschungszielsetzung, die den Rahmen eines neuartigen Grundkonzeptes bildet und im zweiten Teil dieser Arbeit realisiert wird:

1. Entwicklung innovativer Ansätze zur automatisierten Extraktion relevanter Informationen aus herstellerunabhängigen sowie proprietären Dateiformaten
2. Herausbildung von Bauteilsignaturen zur Formenrepräsentation im Zuge der Informationsaufbereitung
3. Informationsweiterverarbeitung zur Konzeption von signaturvergleichenden Suchmaschinen
4. Clusterbildung auf Basis zuvor erkannter Ähnlichkeiten
5. Erforschung sich zusätzlich bietender Möglichkeiten, wie z.B. Auffindung von Substrukturen oder topologisch skalierter Ähnlichkeiten

Die hier genannten Ziele werden in den nächsten Kapiteln anhand der Eigenentwicklungen umgesetzt und hauptsächlich mit der Programmiersprache Python und dem CAE-Programm CATIA V5 implementiert. Zuerst wird jeweils das theoretische Konzept vorgestellt, worauf die Realisierung und die Evaluation folgen.

Teil 2: Eigenentwicklung

Kapitel 4. Informationsextraktion

4.1 STL-Analyse

Konzept

Wie in Kapitel 2.2.1 erwähnt, wurde das STL-Format für Rapid-Prototyping-Anwendungen entwickelt und stellt einzig und allein die Geometrie eines Bauteiles in angenäherter Form dar. Die Grundidee in dem hier vorgestellten Konzept besteht darin diese Geometrie zu analysieren, bestimmte Eigenschaften zu identifizieren und infolgedessen Rückschlüsse auf die Form des Bauteils ziehen zu können. Im Zuge studentischer Projekte sind somit die Arbeiten von Zhu, Scheider und Sheth entstanden, die eine Vielzahl von Möglichkeiten zur Featureerkennung in STL-Dateien untersuchen [*Zhu-13] [*Sch-13] [*She-14].

In der vorliegenden Arbeit sollen nur vier Kennwerte aus den STL-Dateien ausgelesen werden. Die Anzahl aller Dreiecksflächen und Vertices kann in einfacher Art und Weise durch eine Analyse des Quelltextes identifiziert werden. Die Mengen sind in jedem Fall unterschiedlich, da ein Vertex Bestandteil von mehreren Dreiecksflächen sein kann.

Mit der im Folgenden beschriebenen Methode werden neben diesen beiden einfachen Merkmalen auch die im Modell enthaltenen Kanten detektiert und infolgedessen in 'offen' oder 'geschlossen' eingeteilt. Der erste Schritt besteht in der Identifikation von allen Kanten, die im untersuchten Bauteil enthalten sind. Dafür werden die senkrecht zur Fläche stehenden Normalenvektoren eines jeden tesselierten Dreiecks analysiert und im zweiten Schritt die Winkel zwischen den Normalenvektoren aller Nachbarflächen berechnet.

Wenn zwei Dreiecke auf der gleichen Ebene nebeneinander liegen, sind ihre Normalenvektoren parallel bzw. ihr Winkel 0°. Sollten aber zwei Dreiecksflächen Teil einer Rundung oder einer Bauteilkante sein, entsteht je nach Schärfe ein Winkel größer 0° und kleiner 180° und alle Flächen, die entweder Teil einer Rundung oder einer Kante sind, können durch die Berechnung des Winkels mathematisch erkannt werden. Um alle gefundenen Rundungen von den Kanten zu trennen, werden die Flächen, die einen Winkel kleiner als z.B. 45° zu ihren Nachbarflächen aufweisen als zu einer Rundung gehörig definiert und bei der eigentlichen Kantenextraktion übergangen.

Da eine Bauteilkante aus vielen verschiedenen gewinkelten Dreiecksflächen aufgebaut ist, wird anhand der Vertexkoordinaten überprüft welche Dreiecke zusammengehören, damit diese so in die richtige Reihenfolge gebracht werden können. Liegen zwei Dreiecksflächen auf einer Kante nebeneinander, teilen sie sich den Vertex, der beide Flächen miteinander verbindet. Somit kann eine Art Kette gebildet werden, die alle Vertices beinhaltet, die auf einer Kante liegen bzw. die die Kante bilden.

Wurden alle Kanten in einem Bauteil gefunden, werden diese im letzten Schritt in die zwei Kategorien 'offen' oder 'geschlossen' eingeteilt. Das Prinzip ist simpel. Es wird automatisch ein beliebiger Vertex als Startpunkt auf der Kante ausgewählt und in eine Richtung der Kante gefolgt. Wurden alle Knoten der Kante untersucht und die Koordinaten des Ausgangspunktes nicht noch einmal entdeckt, so wird die Kante als 'offen' markiert. Sollte jedoch der Ausgangspunkt wieder erreicht werden, wird die Kante als 'geschlossen' identifiziert, da eine in sich abgeschlossene Kontur entstanden ist.

In den Arbeiten von Scheider, Zhu und Sheth werden außerdem weitere Bauteilfeatures identifiziert, wie z.B. Kreise oder Bauteilsegmente, die ebenfalls aus der Kantenuntersuchung abgeleitet werden können.

Implementierung

Die Umsetzung des beschriebenen Konzeptes erfolgt wie die meisten Algorithmen mithilfe der Programmiersprache Python 2.7 und dem CAE-Programm CATIA V5. Da das Grundprinzip für fast ausschließlich alle der folgenden Programme gilt, soll es an dieser Stelle grundlegend erklärt werden.

So wie viele CAE-Systeme, stellt auch CATIA V5 den Anwendern eine Möglichkeit zur Automatisierung von bestimmten Programmabläufen zur Verfügung. Dieses Konzept wird als Makroprogrammierung bezeichnet und erlaubt es beispielsweise immer gleiche Befehlsfolgen vorher zu definieren, um sie dann ohne weitere Bedienung automatisch ablaufen zu lassen. Im Falle von CATIA V5 muss dafür vom Benutzer ein Skript in der Programmiersprache VBA (Visual Basic for Applications) erstellt werden, das die einzelnen Befehle in Textform enthält und nach einer Bestätigung innerhalb von CATIA gelesen und umgesetzt wird.

Eine ähnliche Vorgehensweise entsteht auch bei der Anwendung von Python. Über die COM-Schnittstelle (Component Object Model) wird am Anfang eines jeden Python-Programmes eine Verbindung mit CATIA hergestellt, die es erlaubt mit einem

Python-basierten Quelltext auf die CATIA-eigenen Befehle zuzugreifen und so das Programm von außen zu steuern. Die Vorteile eines solchen ausgelagerten Zugriffes ergeben sich durch die Möglichkeiten, die sich im Hinblick auf die Verwaltung von Dateien ergeben. Python ermöglicht beispielsweise einen Zugriff auf weitere Programme und erlaubt es insbesondere Textdateien zu erstellen, zu modifizieren und zu verwalten.

Die Implementierung der STL-Analyse wird im Programm stl.py umgesetzt und soll anhand von Abbildung 4.1 erklärt werden. So wie alle Programme dieser Arbeit geht auch stl.py von einer gegebenen CAD-Datenbasis aus, in der die Bauteile im CATPart-Format vorliegen (a). Nachdem vom Benutzer der Ein- und Ausgabepfad gewählt wurde, läuft das Programm automatisch ab und alle Teile im Eingabepfad werden nacheinander abgearbeitet.

Jedes Bauteil wird im ersten Schritt in CATIA geöffnet und bei der automatischen Abspeicherung so ins STL-Format übertragen (b). Die weitere Analyse des Quelltextes läuft danach nur noch mittels Python ab. In einer Schleife werden alle Zeilen analysiert und die Koordinaten einer jeden Dreiecksfläche in eine Liste geschrieben. Mit der im Konzept beschriebenen Methode werden im nächsten Schritt zuerst die Normalenvektoren berechnet (c), dann alle Bauteilkanten identifiziert und entschieden ob diese der offenen oder der geschlossenen Kategorie zuzuordnen sind (d). Für die spätere Weiterverarbeitung werden die gewonnen Daten nur in Form der Anzahl von Dreiecksflächen, Vertices und offenen sowie geschlossenen Kanten im Ausgabepfad in eine Textdatei geschrieben, so dass pro Inputbauteil jeweils eine STL- und eine Textdatei entstehen. Ist ein Teil abgearbeitet, springt eine Schleife so lange zum nächsten CATPart im Eingabepfad bis alle Bauteile umgewandelt und analysiert wurden.

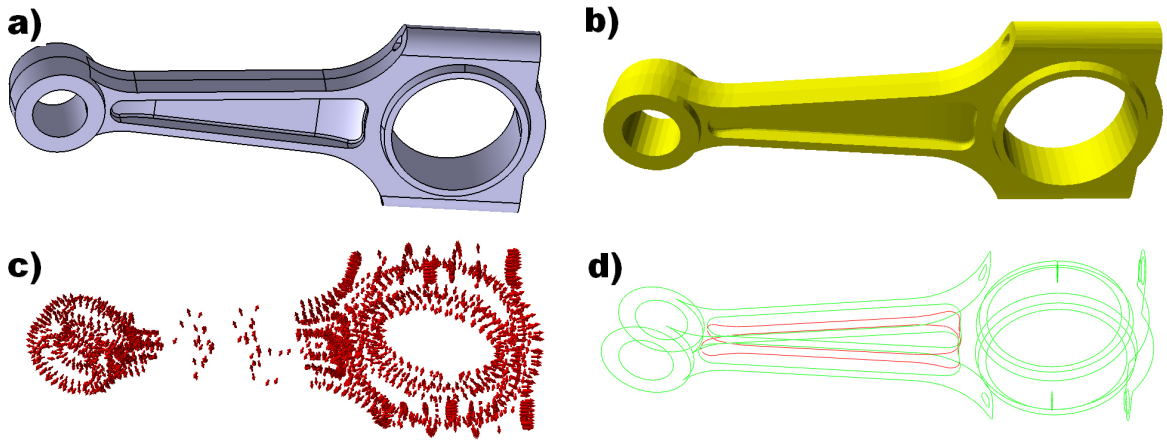

Abbildung 4.1: Vier verschiedene Stadien der STL-Analyse

Evaluation

Im Großen und Ganzen ist anzumerken, dass sich das STL-Format aufgrund der Tatsache, dass die gesamte Geometrie nur angenähert ohne Aufteilung in die einzelnen Bestandteile oder gar Features vorliegt, schlecht zur Analyse eignet. Aus der möglichen Mehrfachnennung vieler Vertices im Quelltext entstehen insbesondere bei großen komplexen Bauteilen große Datenmengen und eine einzige Datei kann somit viele Megabytes in Anspruch nehmen. Aus den gegebenen Datenmengen entstehen bei der Analyse lange Rechenzeiten, die pro Bauteil mehrere Minuten betragen können und sich somit für eine Anwendung auf große Datenmengen schlecht eignen. Dennoch ist anzumerken, dass nach der Eingabe der Parameter das Programm gänzlich automatisch abläuft und kein weiteres Eingreifen mehr nötig ist.

4.2 Voxel-Analyse

Konzept

Die Voxelisierung eines Bauteiles wird in den meisten Fällen durchgeführt um das Objekt zu vereinfachen und so die Untersuchbarkeit bzw. die Zugänglichkeit zu den Eigenschaften des Teiles zu erhöhen. Ebenso wie in den erwähnten Veröffentlichungen aus Kapitel 3.10, in denen eine Bauteilsignatur vom Voxelmodell abgeleitet wird, soll hier eine Vorgehensweise vorgestellt werden, die bestimmte Kennwerte eines CAD-Modells mithilfe einer Voxeldatei ausliest.

Für eine anfängliche Übertragung eines gegebenen CAD-Modells in eine voxelisierte Form, muss die Bauteilkontur analysiert werden. Dafür wird ein dreidimensionales Raster über bzw. um das CAD-Modell gelegt, in dem jeder Kasten eine zuvor festgelegte Kantenlänge besitzt, die später die Größe der würfelförmigen Voxel definiert. Dem Konzept der Tesselierung von STL-Dateien entsprechend ist es somit nicht möglich Rundungen darzustellen. Doch je kleiner die Voxelgröße definiert wird, desto genauer gleicht das Voxelmodell dem Original.

Im nächsten Schritt wird das dreidimensionale Raster in zweidimensionale Sektionen zerlegt, die anschließend 'scheibchenweise' untersucht werden. Ähnlich wie bei einem Bild, auf dem nur die Kontur zu erkennen ist, analysiert ein Algorithmus die Silhouette des Bauteiles und platziert ausschließlich an den Stellen Voxel, an denen auch Material vorhanden ist. Wenn alle Bauteilsektionen bzw. entstandene Segmente untersucht wurden, können sie wieder in Form des ursprünglichen Rasters gebracht werden und bilden somit das dreidimensionale voxelisierte Modell.

Mit der eigentlichen Analyse dieses Modells können die originalen Kennwerte des ursprünglichen Teiles, wie z.B. die Maximalabmaße (Bounding Box), das Volumen oder die Massenträgheitsmoment, angenähert werden. Doch auch einfach zu erkennende Informationen, wie z.B. die Größe eines Voxels oder die Anzahl aller im Modell enthaltenen Voxel, können aussagekräftig sein.

Implementierung

Die Umsetzung des oben beschriebenen Konzeptes erfolgt im Programm voxel.py und kann in die Abschnitte Voxelisierung und Analyse eingeteilt werden. Abbildung 4.2 zeigt die unterschiedlichen Stadien bei der Erstellung eines Voxelmodells. Die eigentliche Übertragung geschieht mithilfe des Programmes binvox, das zum kostenlo-

sen Download verfügbar ist [Binv-15]. Da dieses als Ausgangsmodell (a) eine Datei im STL-Format (b) verlangt, müssen alle Bauteile, die in der Datenbasis eventuell im CATPart-Format vorliegen unter Zuhilfenahme der Makroprogrammierung zuerst automatisch in CATIA geöffnet und dann im STL-Format abgespeichert werden. In einer Schleife, die alle Dateien im Eingabepfad fortlaufend aufruft, werden dann nacheinander die im gewählten Verzeichnis befindlichen Bauteile in binvox eingespeist, wo sie einzeln abgearbeitet und voxelisiert werden.

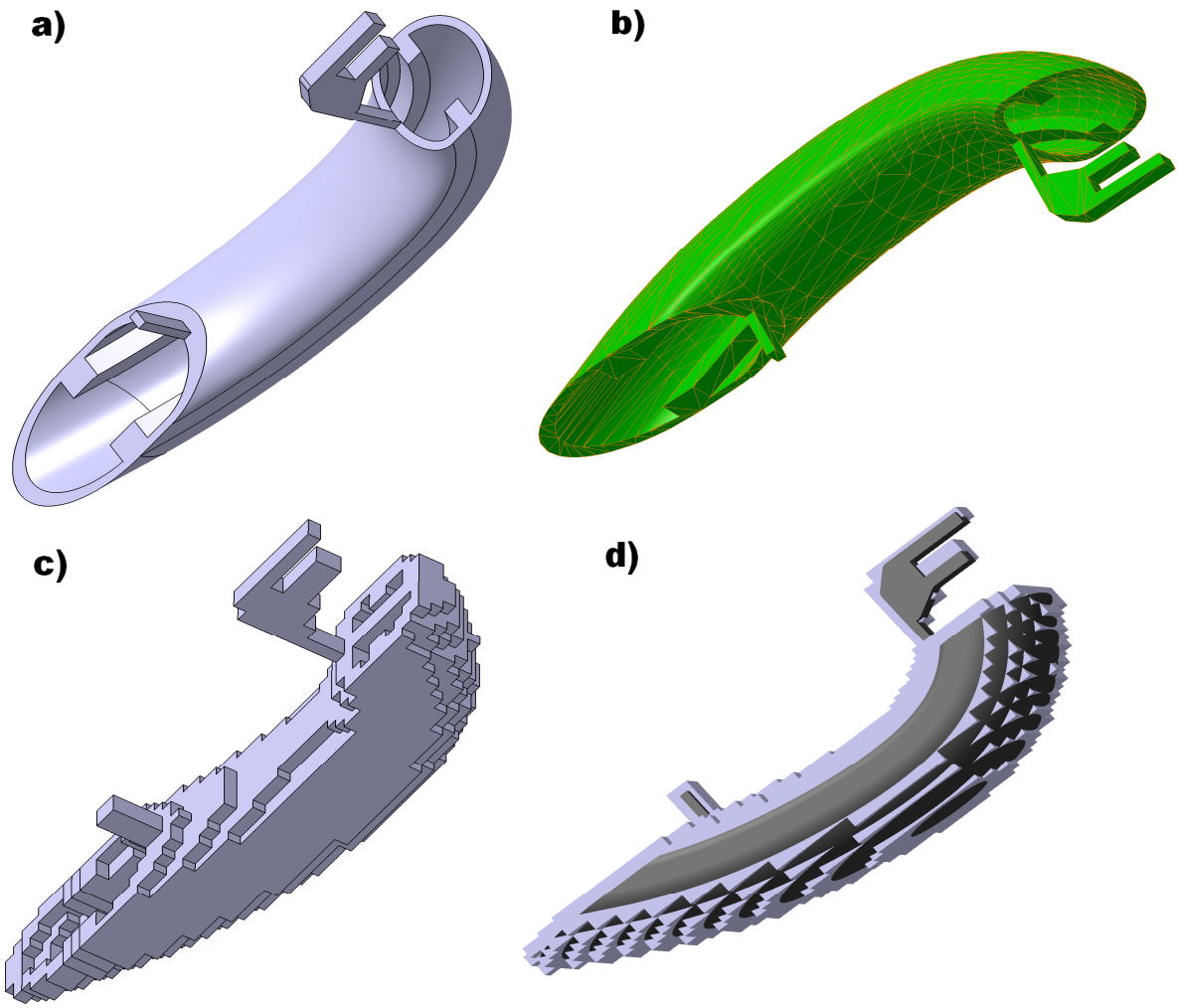

Abbildung 4.2: Der Prozess vom CAD- zum Voxelmodell

Die Definition einer solchen Voxeldatei (c) erfolgt im MSH-Format in der Ausgabe von binvox. Basierend auf einer Textdatei, die später mithilfe von Python analysiert wird, sind die Koordinaten aller Voxel und somit die gesamte Geometrie in einer solchen MSH-Datei enthalten (vgl. Kapitel 2.2.2). Eine Visualisierung der neu entstandenen Geometrie wird beispielsweise von Dhandam in CATIA V5 umgesetzt [*Dha-14]. Abbildung 4.2 zeigt das Ergebnis der Visualisierung der gezeigten Beispieldatei. Im letzten Teilbild sind außerdem beide Versionen des Bauteils übereinander

mit der gleichen Ausrichtung auf die Ursprungskoordinaten gelegt, so dass der Unterschied der entstandenen Modelle und ihre Größenverhältnisse verdeutlicht werden (d).

Die auf die Voxelisierung folgende Analyse der MSH-Datei erfolgt durch eine zeilenweise Untersuchung des Quelltextes. Nachdem die Koordinaten aller Voxel in eine Liste geschrieben wurden, werden die Extremwerte (maximale Bauteilabmessungen) der Würfelkoordinaten extrahiert. Für die Volumenberechnung muss die gesamte Voxelanzahl (gleich der Listenlänge) mit der Größe eines einzigen Voxels multipliziert werden, da diese im gesamten Modell konstant ist. Weiterhin werden auch die Massenträgheitsmomente der Gesamtgeometrie berechnet, so dass nach Vollendung der Listenuntersuchung alle gesammelten Informationen in eine Ausgabedatei geschrieben werden können. Somit entstehen für jede Ausgangsdatei im CATPart-Format am Ende eine STL-, eine MSH- und eine Textdatei.

Evaluation

Verglichen mit den professionelleren Methoden zur Untersuchung von voxelisierten Modellen (vgl. Kapitel 3.10), ist das hier vorgestellt Konzept relativ unscheinbar und nur dafür geeignet einige grundlegende Kennwerte zu identifizieren. Neben dem Informationsgehalt müssen außerdem die Bedienung und die Dauer für einen Programmdurchlauf bei der Bewertung in Betracht gezogen werden. Diese können beide als positiv eingeschätzt werden, da das Programm nach der Eingabe der Inputparameter automatisch abläuft und für jedes Bauteil relativ wenig Zeit in Anspruch nimmt. Dennoch hängt die Berechnungsdauer maßgeblich von der Größe der Voxel und der Komplexität des Bauteiles ab. Bei einer Verfeinerung der Voxelgröße konvergieren einerseits die Kennwerte der voxelisierten Modelle mit den korrekten Werten der Originalteile, aber andererseits werden auch die Datenmengen und Berechnungszeiten gesteigert, so dass ein angemessener Mittelweg gefunden werden sollte.

In der Arbeit von Dong werden weitere Möglichkeiten zur Modellanalyse untersucht und die Performancezeiten sowie die Bauteilkennwerte an einigen Beispielen verdeutlicht [*Don-14].

4.3 Strukturbaumanalyse

Konzept

Ähnlich der Vorgehensweise der in Kapitel 2.8 beschriebenen PDM-Programme bei der Bauteilanalyse, soll hier eine Methode zur Strukturuntersuchung vorgestellt werden. So wie bei den gängigen CAE-Systemen üblich, sind alle Schritte, die bei der Bauteilerstellung vollzogen wurden, im Strukturbaum abgespeichert. In einigen Fällen sind darüber hinaus noch weitere Kennwerte, wie z.B. Material, Schwerpunkt oder Gewicht, zusätzlich vorhanden. Diese Informationsfülle ist für eine externe Charakterisierung des Bauteiles von großem Interesse und legt nahe den Strukturbaum zu untersuchen und die Daten herauszulesen bzw. zu extrahieren.

Wie in Abbildung 4.3 zu erkennen, ist der Strukturbaum in einer gewissen Hierarchie aufgebaut. Ausgehend von den zweidimensionalen über die dreidimensionalen Features und einzelnen Bestandteile steigt die Hierarchie bis zum Bauteilnamen auf und wird dabei immer gröber. Beim Vorgang des Auslesens sollte die Baumstruktur beibehalten werden, da sie neben ihren Einträgen auch Informationen anhand ihrer Form überträgt.

Um auf den Strukturbaum zugreifen zu können, sollte eine Zugangsmöglichkeit von der CAE-Software geboten werden. Mithilfe der bereits erwähnten Makroprogrammierung ist dies leicht umzusetzen, da die einzelnen Einträge anhand ihrer Namen angesprochen und untersucht werden können.

Implementierung

Wie in Abbildung 4.3 angedeutet, erfolgt die Umsetzung des Konzeptes in CATIA V5 mithilfe der Makroprogrammierung in Python (read_tree.py). Nachdem der Benutzer den Eingabepfad mit allen zu untersuchenden Bauteilen bestimmt und sich Python mit CATIA verbunden hat, werden die Bauteile nacheinander geöffnet und jeweils zu Anfang die im oberen Teil von Abbildung 4.4 zu erkennenden Kennwerte mithilfe der sogenannten 'SPAWorkbench' aus dem Modell extrahiert. Dabei handelt es sich um ein Analysewerkzeug, das es dem Benutzer beispielsweise erlaubt die Eigenschaften des Bauteils angezeigt zu bekommen oder auch punktgenaue Messungen zwischen zwei Elementen durchzuführen. Hier werden Parameter die das vollständige Teil betreffen, wie z.B. die Gesamtoberfläche oder das Volumen zur Verfügung gestellt.

Im nächsten Schritt wird das Material des Bauteiles untersucht. Da eine Materialdefinition normalerweise nicht in allen Bauteilen konsequent durchgeführt wird, werden beide Möglichkeiten (definiert oder nicht definiert) berücksichtigt. Sollte keine Zuweisung vorliegen, müssen auch Dichte und Masse, die sich aus den Material- und Bauteilparametern berechnen lassen, als 'not available' markiert werden und tauchen später nicht mehr auf.

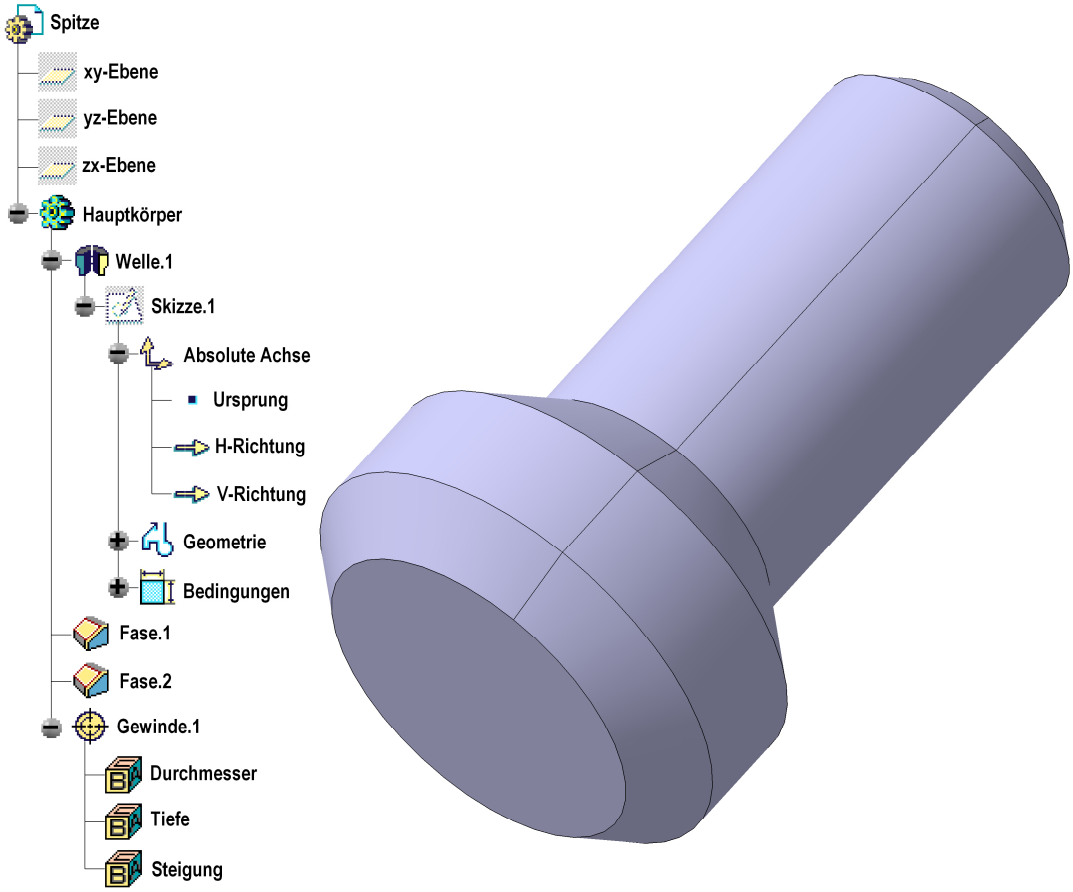

Abbildung 4.3: Beispielteil und Strukturbaum in CATIA V5

Für die Bestimmung der Bounding Box, die vom Programm im nächsten Schritt durchgeführt wird, muss das Bauteil in einer CATIA-eigenen technischen Zeichnung in zwei verschiedenen Ansichten auf ein Blatt projiziert werden, dessen Maße sich an die Größe des Bauteils anpassen (vgl. Kapitel 4.4). Gleichzeitig kann auch der Blattrand definiert und so eingestellt werden, dass ein festes Maß von z.B. 5mm entsteht.

Nachdem die drei Maximalmaße in X-, Y- und Z-Richtung mit dieser Methodik ermittelt wurden, beginnt in einer weiteren Schleife die eigentliche Strukturbaumanalyse. Dafür werden alle Entitäten nacheinander ebenenweise angesprochen und auf die Informationen eventuell vorhandener geometrischer Primitive zugegriffen.

Da sich das XML-Format für die Speicherung von baumartigen Strukturen eignet (vgl. Kapitel 3.4), erstellt das Programm nach dem Abschluss aller Untersuchungen

eine Textdatei, in die alle Daten XML-konform eingetragen werden. Wird dieser Quelltext mit einem XML-kompatiblen Programm geöffnet (z.B. einem Browser), so wird es ermöglicht den Quelltext in verschiedenen Farben übersichtlich zu formatieren. Abbildung 4.4 zeigt den ausgelesenen Quelltext des Bauteils in Abbildung 4.3.

```
<PartDocument
name="Kesselspitze.CATPart"
sizex="13.08125"
sizey="7.9625"
sizez="8.00625"
volume="354.999969729"
surface="301.184668811"
density="n/a"
mass="n/a"
material="n/a"
centroid="(-5.1924778777645, 3.03606679321635e-10, 1.11137483223370e-09)"
momentofinertia="(1.8521259475647735e-09, 0.0, 0.0, 0.0,
5.9651679726391547e-09, 0.0, 0.0, 0.0, 5.9651679726391547e-09)">
  <AxisSystems> </AxisSystems>
  <bodies name="Hauptkörper">
   <Shaft name="Welle.1">
    <GeometricElements>
     <AbsoluteAchse> </AbsoluteAchse>
     <Linie.1> </Linie.1>
     <Punkt.1> (13.0, 0.0, None) </Punkt.1>
     <Linie.2> </Linie.2>
     <Punkt.3> (0.0, 4.0, None) </Punkt.3>
     <Linie.3> </Linie.3>
     <Punkt.4> (2.9999, 3.9999, None) </Punkt.4>
     <Linie.4> </Linie.4>
     <Punkt.5> (4.5, 2.4999, None) </Punkt.5>
     <Linie.5> </Linie.5>
     <Punkt.6> (13.0, 2.4999, None) </Punkt.6>
     <Linie.6> </Linie.6>
     <Punkt.7> (13.0, 0.0, None) </Punkt.7>
    </GeometricElements>
    <Constraints>
     <Parallelität.1> </Parallelität.1>
     <Parallelität.3> </Parallelität.3>
     <Parallelität.4> </Parallelität.4>
     <Parallelität.5> </Parallelität.5>
     <Parallelität.6> </Parallelität.6>
     <Kongruenz.7> </Kongruenz.7>
     <Winkel.12> 45.0 </Winkel.12>
     <Länge.2> 13.0 </Länge.2>
     <Länge.8> 4.0 </Länge.8>
     <Länge.9> 3.0 </Länge.9>
     <Länge.10> 8.5 </Länge.10>
    </Constraints>
   </Shaft>
   <Chamfer name="Fase.1"> </Chamfer>
   <Chamfer name="Fase.2"> </Chamfer>
   <Thread name="Gewinde.1"> </Thread>
  </bodies>
  <GeometricalSets> </GeometricalSets>
</PartDocument>
```

Abbildung 4.4: XML-Quelltext

Für die Erstellung wird innerhalb von Python ein Modul mit dem Namen 'Beautiful Soup' verwendet, das als 'Parser' bezeichnet wird und die Erstellung sowie die Zugänglichkeit zu XML-Dateien vereinfacht [Beau-15]. Da als Basiseinheit für Längenmaße in den meisten CAE-Programmen mm verwendet wird, sind auch im Quelltext alle Längen-, Flächen-, und Volumenmaße in mm angegeben. Lediglich die Massenträgheitsmomente werden in kg*m^2 angezeigt. Eine vollständige Liste aller anderen Informationen und den jeweiligen Einheiten wird in Kapitel 5.1 vorgestellt.

Nachdem das Programm alle Bauteile geöffnet, den Strukturbaum ausgelesen, abgespeichert und wieder geschlossen hat, ist im Ausgabeverzeichnis pro Ausgangsbauteil jeweils eine XML-Datei entstanden.

Evaluation

Die Anwendbarkeit des Programmes read_tree.py kann als positiv bewertet werden. So wie alle anderen Programme, läuft die Strukturbaumextraktion automatisch ab und die Dauer eines Durchlaufs wird maßgeblich von der Komplexität der Bauteile beeinflusst.

Obwohl die genauen Abläufe der professionellen PDM-Systeme nicht genau nachvollzogen werden können, kann angenommen werden, dass im Prinzip das gleiche Konzept bei der Teileanalyse umgesetzt wird. In Programmen, wie z.B. Exalead One Part (vgl Kapitel 2.8.2) wird für die schnelle Bauteilsuche auch ein Hilfsformat angelegt, das mit der hier entstehenden XML-Datei vergleichbar ist. Eine textbasierte Suche auf Basis regulärer Ausdrücke kann viel schneller durchgeführt werden als eine Untersuchung der Geometrie. Grundsätzlich kann der Strukturbaum bzw. die zugehörige XML-Datei schon als Bauteilsignatur angesehen werden, die zwar nicht eindeutig ist, aber dennoch viele Eigenschaften und Kennwerte des Bauteils beinhaltet.

Auf Basis der Idee allein den Strukturbaum zur Ähnlichkeitssuche zu verwenden wurde ein Journalartikel sowie ein studentisches Projekt veröffentlicht, die im Folgenden kurz vorgestellt werden sollen [*Roj-14] [*Vad-14]. Im Artikel werden nach der Strukturbaumextraktion drei verschiedene Suchmaschinen entwickelt und ausgewertet, die auch auf die Anforderungen einer großen Datenbank mit proprietären CAD-Modellen zugeschnitten sind.

Die erste eigens entwickelte Suchmaschine ist auf die Auffindung von einfachen Normteilen, wie z.B. Unterlegscheiben, Passfedern oder Muttern, spezialisiert, deren Bauteileigenschaften immer ähnlich sind und aus diesem Grund auf einfache Art und Weise in das System eingepflegt werden können.

Die zweite Suchmaschine erlaubt es dem Benutzer die Features, die in den Teilen enthalten sind, für die Suche so zu definieren, dass nicht nur nach festgelegten Zahlen, sondern auch nach ganzen Wertebereichen gesucht werden kann. Dieses Programm ist am besten geeignet, wenn der Anwender schon vorher eine gewisse Vorstellung von den gewünschten Suchergebnissen besitzt.

Mithilfe der letzten vorgestellten Suchmaschine lässt sich ein Bauteil bzw. eine XML-Datei als Referenz angeben, die zuerst analysiert wird, um danach alle gleichen oder ähnlichen Bauteile aus der Datenbasis herauszufiltern.

Vadlamani überprüft die Performance-Zeiten anhand von großen Datenmengen und erweitert insbesondere die Suchmaschine für Normteile. Er kommt zu dem Schluss, dass alle Ansätze auch bei längeren Ladezeiten praktikabel sind und durchaus für die Anwendung im industriellen Alltag von Interesse sein könnten.

4.4 Bauteilrekonstruktion aus Bildern und Zeichnungen

Konzept

Das hier vorgestellte Konzept zur Bauteilrekonstruktion anhand von Bildern oder Skizzen ist mit den in Kapitel 3.7 vorgestellten Vorgehensweisen im Bereich des Reverse Engineering zu vergleichen. Die Grundidee der 3D-Modellerstellung aufgrund von zweidimensionalem Input bleibt bestehen. Im besten Falle wird hier von detaillierten Fertigungszeichnungen ausgegangen, die z.B. auch verdeckte Kanten anhand von gestrichelten Linien visualisieren. Die gleiche Schrittfolge kann allerdings auch auf Basis von Fotos oder Handskizzen umgesetzt werden, die jedoch nicht über den Informationsgehalt einer technischen Zeichnung verfügen.

Die gesamte Methode soll am Beispiel der folgenden Abbildungen demonstriert werden. Für die Erstellung der technischen Zeichnungen wird das originale CAD-Modell, das in Abbildung 4.5 in zwei Ansichten zu sehen ist, als Beispiel verwendet, so dass die Rekonstruktionsergebnisse später mit den Kennwerten des Originals gut zu vergleichen sind. Das eigentliche Konzept soll an dieser Stelle mithilfe von CATIA V5 demonstriert werden, ist aber prinzipiell ohne Änderungen auch auf andere Software übertragbar.

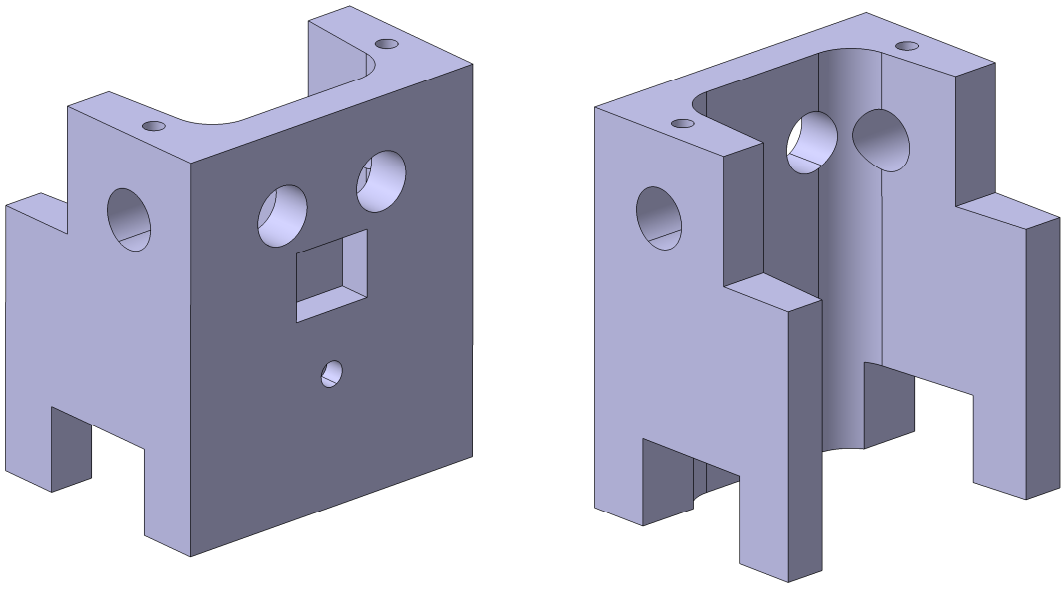

Abbildung 4.5: Ausgangsbauteil zur Rekonstruktion

Als Ausgangsmaterial werden für die Rekonstruktion von technischen Zeichnungen für die vollständige Durchführung sechs Abbildungen des Bauteiles in den Hauptansichten eines Würfels benötigt. Abbildung 4.6 zeigt drei dieser Ansichten des Beispiel-

teils aus Abbildung 4.5 im SVG-Format (vgl. Kapitel 2.2.4). Neben den Durchgangsbohrungen, sind für eine detailgetreue Rekonstruktion insbesondere die Sacklochbohrungen, die nicht durchgängig sind, und die zugehörigen verdeckten Linien von Interesse.

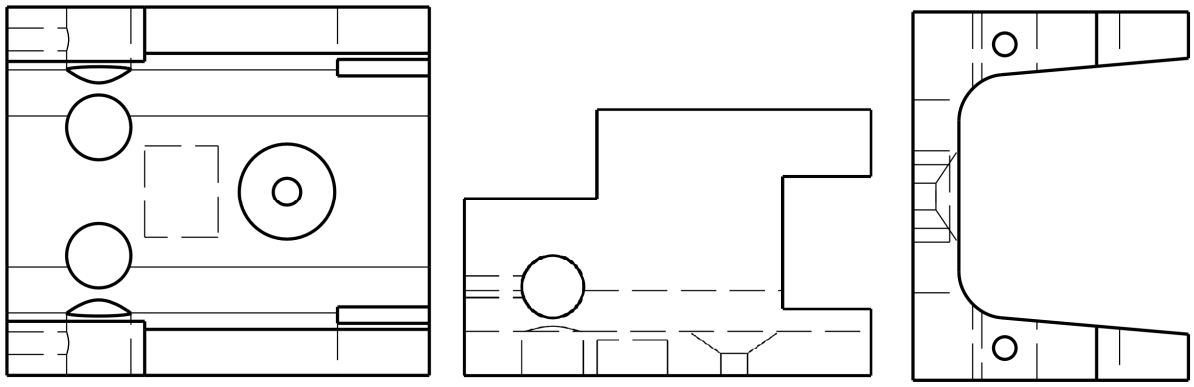

Abbildung 4.6: Drei Hauptansichten im SVG-Format

Um den dreidimensionalen Hauptkörper zu erhalten, dessen Details am Ende noch verfeinert werden, müssen die Konturen von drei Ansichten zuerst untersucht und dann in das CAD-Programm übertragen werden, wo sie als Teilkörper erzeugt und mithilfe von zwei der booleschen Operationen 'Verschneidung' zu vereinen sind. Die Bauteilsilhouette wird aus den drei Hauptansichten extrahiert, deren Umrisse mit einem Schatten zu vergleichen sind. Wenn beispielsweise eine Durchgangsbohrung in der Ansicht enthalten ist, so ist innerhalb des Bohrungsdurchmessers die Hintergrundfarbe zu erkennen und sie gehört nicht zur endgültigen Kontur.

Im nächsten Schritt werden die gefundenen Silhouetten in ein CAD-Programm übertragen, so dass sie dort in die dritte Dimension auf die richtige Länge extrudiert werden können. Abbildung 4.7 zeigt das Ergebnis der drei Konturextrusionen des Beispielteils. Die abgestufte Struktur, die speziell an den rekonstruierten Rundungen zu erkennen ist, entsteht durch eine pixelbasierte Konturerkennung, die, wie z.B. die dreidimensionalen Voxel, eine Geometrie nur annähert, aber nicht ideal wiedergibt.

Nachdem die Zeichnungen vom zweidimensionalen Raum im Modell in die dritte Dimension übertragen wurden und somit drei einzelne Bauteile vorliegen, müssen diese als nächstes vereint werden. Dafür wird das Werkzeug der booleschen Verschneidung gewählt, die zwei Male durchgeführt werden muss. Anhand der Ausgangskoordinaten werden zuerst zwei der beiden Extrusionsteile übereinandergelegt und das Material automatisch überall dort entfernt wo es nur einfach vorhanden ist. Nachdem die erste boolesche Operation durchgeführt wurde, muss bei der zweiten das entstandene Bauteil beibehalten und mit der dritten Extrusion verschnitten werden.

Das Ergebnis dieser beiden Verschneidungen ist in Abbildung 4.8 dargestellt. Sie bildet eine Art Rohkörper, der in den nächsten Schritten noch genauer an das Original angenähert wird.

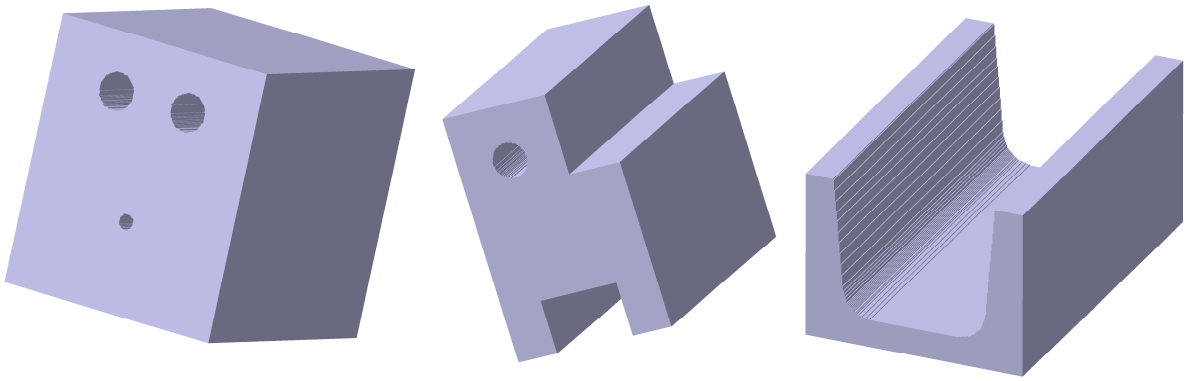

Abbildung 4.7: Die drei extrudierten Hauptansichten

Weitere Genauigkeiten bzw. Charakteristiken des Originalbauteils lassen sich durch eine eingehendere Untersuchung der SVG-Dateien ermitteln. Im weiteren Verlauf der Methode werden die anfangs erwähnten sechs Hauptansichten auf Kreise und Rechtecke untersucht und die zweidimensionalen Koordinaten abgespeichert. Bei jedem Kreis oder jedem Rechteck wird davon ausgegangen, dass es sich um eine Sacklochbohrung oder eine Tasche handelt, die nicht durchgängig ist.

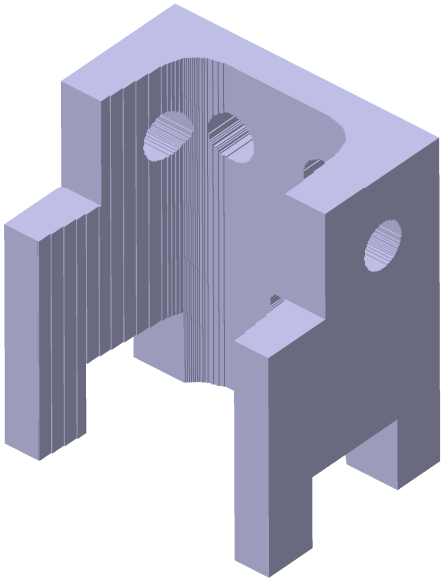

Abbildung 4.8: Rekonstruierter Grundkörper

Anhand einer Untersuchung der zugehörigen Seitenansichten wird dieser Verdacht bestätigt oder kann wieder verworfen werden. Erst wenn in den Seitenansichten an der richtigen Stelle zwei parallele, gleich lange gestrichelte Linien zu finden sind, kann

davon ausgegangen werden, dass es sich bei dem Kreis um eine Durchgangsbohrung und bei dem Rechteck um eine Tasche handelt.

Nachdem alle Werte, die für die Umsetzung im 3D-Modell von Nöten sind, gesammelt wurden, müssen diese Verfeinerungsfeatures in das Rohteil eingearbeitet werden. Das Ergebnis des Beispielteils nach dieser Verfeinerung ist in Abbildung 4.9 zu sehen.

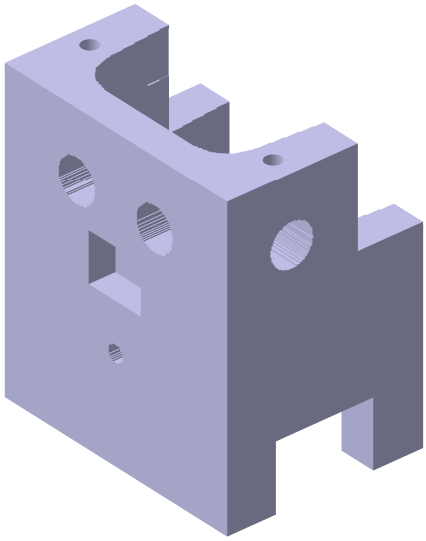

Abbildung 4.9: Vollständig rekonstruiertes Beispielteil

Für die Umsetzung der Sacklochbohrungen werden die Kreiskoordinaten, der Kreisdurchmesser und die Bohrungstiefe benötigt. Bei den rechteckigen Taschen entspricht dies den Parametern der vier Eckpunkte des zuvor erkannten Rechtecks.

Nach der Umsetzung der Verfeinerungsfeatures ist die Rekonstruktion abgeschlossen und das Bauteil der ursprünglichen technischen Zeichnung liegt als dreidimensionales CAD-Modell vor.

Implementierung

Wie in Tabelle 1.1 zu erkennen, wird die gesamte Methodik in den vier Programmen create_stl.py, stl_to_bool.py, drilling.py und pocket.py realisiert. Als Inputdateien bei der Umsetzung verlangt create_stl.py mindestens die drei Hauptansichten im SVG- und TIF-Format. Alle sechs Ansichten werden später für die Rekonstruktion der Bohrungen und Taschen benötigt. Nachdem das Programm die Pixel der Bilder zeilenweise untersucht und so die Koordinaten der Kontur ermittelt hat, werden diese für jede Ansicht in jeweils eine zweidimensionale STL-Datei geschrieben, die lediglich die Bauteilsilhouette enthält.

Das nächste Programm stl_to_bool.py greift die drei erstellten STL-Dateien eines jeden Ausgangsbauteils wieder auf und setzt die beschriebenen booleschen Operationen mithilfe von Makroprogrammierung innerhalb von CATIA um. Dabei kann die Struktur des Programmes in die zwei Schritte, die in Abbildung 4.7 und 4.8 zu erkennen sind, unterteilt werden. Im ersten Teil werden die STL-Dateien in der Bauteilumgebung geöffnet und auf die zuvor berechnete Länge in die dritte Dimension extrudiert. Für die Weiterverarbeitung im zweiten Teil müssen die drei extrudierten Ansichten zwischengespeichert werden, so dass sie im zweiten Teil des Programmes nach und nach geöffnet und miteinander verschnitten werden können. Wichtig ist dabei, dass die Extrusionsrichtung und die Platzierung der STL-Dateien an den Ursprungskoordinaten beachtet werden. Nach der zweiten Verschneidung wird das rekonstruierte Rohteil abgespeichert und liegt für die weitere Verfeinerung in den Programmen drilling.py und pocket.py bereit.

Diese besitzen im Prinzip die gleiche Quelltextstruktur. Die Kreiserkennung erfolgt durch die Analyse des SVG-Quelltextes. Zur Erkennung der Taschen werden die Rechtecke mithilfe des Python-Moduls 'SimpleCV' analysiert und ihre Koordinaten als sogenannte 'Blobs' abgespeichert [Simp-15]. Die Untersuchung der verdeckten Strichlinien erfolgt bei beiden Programmen in den jeweiligen Seitenansichten der SVG-Dateien. Nach Ermittlung aller nötiger Informationen werden im CATIA-Modell von den richtigen Seitenansichten ausgehend automatisch Skizzen erstellt, die mit den Befehlen 'Kreis' bzw. 'Rechteck' gefüllt und anschließend materialabtragend extrudiert werden. So entsteht ein noch detaillierteres rekonstruiertes Bauteil, das sehr genau an das Originalteil angelehnt ist.

Evaluation

Verglichen mit den Ansätzen im Bereich des Reverse Engineering, die in Kapitel 3.7 vorgestellt wurden, unterscheidet sich die grundsätzliche hier beschriebene Vorgehensweise maßgeblich. Die Ergebnisse können dennoch als sehr genau bewertet werden, wie in Tabelle 4.1 zu erkennen ist. Dort werden einige grundlegende Bauteilparameter zwischen dem originalen und dem rekonstruierten Bauteil aus den Abbildungen 4.5 und 4.9 verglichen und es kann geschlussfolgert werden, dass die Rekonstruktion gut gelungen ist. Die Unterschiede, die insbesondere bei der Bauteiloberfläche zu erkennen sind, können auf die abgestuften Oberflächen der rekonstruierten Rundungen zurückgeführt werden. Lediglich die kegelförmige 90°-Senkung auf der Innenseite des Bauteils wurde bei der Rekonstruktion nicht berücksichtigt. Auch der Speicherplatzbedarf sollte bei großen Bauteilmengen beachtet werden.

Zur weiteren Evaluation der Implementierung soll auf eine Veröffentlichung verwiesen werden, die neben einer umfassenderen Beschreibung auch Ausführzeiten und weitere Beispiele umfasst [*RoW-13].

Eine Anwendung mit handgezeichneten Skizzen als Inputbilder kann für relativ einfache Formen auch als praktikabel angesehen werden [*RoW-14].

Auch die Studentenprojekte von Miao, Peng und Zhu untersuchen die Ergebnisse der vorliegenden Rekonstruktionsmethodik eingehender [*Mia-14] [*Pen-13] [*Zhu-14]. In allen drei Untersuchungen wurde die Anwendbarkeit der Programme im Hinblick auf ihre Benutzerfreundlichkeit und Prozessgeschwindigkeit untersucht.

	Größe X	Größe Y	Größe Z	Oberfläche	Volumen	Speicher
Originalteil	46mm	40mm	30mm	9064mm²	19850mm³	0,3MB
Rekonstruktion	46,375mm	40,425mm	30,275mm	9770mm²	20975mm³	10,5MB
Abweichung	0,82%	1,06%	0,92%	7,79%	5,67%	3400%

Tabelle 4.1: Parametervergleich

Miao verwendet die Methode ohne die Bohrungs- und Taschenrekonstruktion anhand selbst erstellter Fotos von Bauteilen und gelangt zu dem Schluss, dass die Genauigkeit der Ergebnisse sowohl von der Fotoqualität, als auch von der Form des Objektes abhängen. In erster Linie sind verwinkelte und rotationssymmetrische Geometrien nicht so gut geeignet, wie eckige oder ebene Teile.

Peng erstellt mit Makros, die basierend auf einem Zufallsgenerator Normteile erstellen, eine große Datenbasis und gelangt nach einer Anwendung der Rekonstruktionsmethode zum Schluss, dass diese zwar automatisch abläuft, sich aber auch rechen- und zeitintensiv verhält.

Im Projekt von Zhu werden mit der gesamten Methode Bauteile anhand von Bildern rekonstruiert und dann mithilfe des CATIA-eigenen FEM-Moduls auf ihre mechanischen Eigenschaften untersucht. Obwohl auch hier die Konsistenz der Ergebnisse stark von der Form der Ausgangsbauteile abhängt, unterscheiden sich die technischen Kennwerte relativ wenig. Lediglich die bereits erwähnte abgestufte Struktur bei Rundungen kann zu Spannungsspitzen und damit Bruchstellen führen.

4.5 VRML-Analyse

Konzept

Bei einer Analyse der Struktur von VRML-Dateien liegt der Gedanke nahe, den Quelltext zur Informationsgewinnung eingehend zu untersuchen (vgl. Kapitel 2.2.5). Da eine jede VRML-Datei in Flächen- und Liniensets unterteilt ist, die jedes einzelne geometrische Element, aus dem das Bauteil aufgebaut ist, genau beschreiben, besteht die Grundidee der VRML-Analyse darin alle Flächen in bestimmte Kategorien zu unterteilen. Der Vorteil im Gegenteil zum STL- oder MSH-Format liegt in der automatischen Einteilung des Quelltextes in die verschiedenen Flächenelemente, die in der Datei nur noch identifiziert werden müssen.

Bei einer genaueren Untersuchung eines Flächen- bzw. eines Liniensets ist zwar die Abgrenzung im Quelltext genau zu erkennen und somit eindeutig welche Punkte zu welcher Fläche gehören, aber dennoch besteht der Text nur aus einer Auflistung von dreidimensionalen Punktkoordinaten. Die genaue Vorgehensweise bei der Vernetzung und die Flächenform sind zunächst unbekannt. In der hier vorgestellten Methode zur VRML-Analyse sollen aus diesem Grund zuerst alle Flächen- und Liniensets aus dem Quelltext mithilfe regulärer Ausdrücke extrahiert und dann auf ihre Form hin untersucht werden. Wurde der jeweiligen Fläche eine bestimmte Form zugeordnet, wird die gesamte Fläche durch die Definition gewisser Attribute im Quelltext der VRML-Datei eingefärbt. Dabei wird zwischen den folgenden sieben Flächenformen unterschieden:

1. Dreiecksflächen (grün)
2. Viereckflächen (rot)
3. Sechseckflächen (türkis)
4. Flächen in einer Ebene (magenta), die keine Dreiecks-, Vierecks-, Sechsecks- oder Kreisflächen sind.
5. Kreisflächen (blau)
6. Rundungen (gelb)
7. Freiformflächen (grau)

Alle Flächen, die nicht einer bestimmten Kategorie zugewiesen werden können, werden hier als Freiformflächen bezeichnet und behalten ohne spezielle Färbung ihre graue Standardfarbe.

Nach der Untersuchung und Kolorierung können schon einige Rückschlüsse auf die Charaktereigenschaften des Ausgangsbauteiles gezogen werden. So sollte bei einer Form mit vielen Rundungen und Kreisen von einem rotationssymmetrischen Objekt ausgegangen werden, während viele Viereckflächen auf ein einfaches eckiges Teil

hindeuten. Des Weiteren muss die Gesamtflächenzahl berücksichtigt und ins Verhältnis mit eventuell gefärbten Flächen gesetzt werden.

Nachdem für jede ursprüngliche VRML-Datei eine eingefärbte Version entstanden ist, sind diese für die weitere Analyse bereitgelegt. Weiterhin entsteht ein Fingerprint aus der zugrunde liegenden kolorierten VRML-Datei (vgl. Kapitel 5.3).

Implementierung

Die Implementierung erfolgt im Programm vrml.py. Auch dieses geht von einer Datenbasis aus, die im CATPart-Format vorliegt, und überträgt am Anfang des Algorithmus die Dateien ins VRML-Format. Im nächsten Schritt wird der Quelltext zeilenweise untersucht und alle Flächen- sowie Liniensets mit regulären Ausdrücken zuerst aufgespürt, dann alle dreidimensionalen Punktkoordinaten extrahiert und am Ende in jeweilige Listen geschrieben.

Die eigentliche Untersuchung der Flächentypen erfolgt im zweiten Teil des Programmes, in dem die Listeneinträge untersucht und dann einer der oben angegebenen Flächenkategorien zugewiesen werden. Die Erkennung der ersten drei Flächentypen (Dreiecks-, Vierecks- und Sechsecksflächen) wird durch die Untersuchung der Setlänge vollzogen. Wenn ein Flächenset beispielsweise nur aus drei Einträgen und somit aus drei Vertices besteht, so muss es sich bei der vernetzten Geometrie im Modell um eine Dreiecksfläche handeln.

Nach der Filterung bzw. Einfärbung dieser einfachen Flächentypen, werden als nächstes die Rundungen herausgesucht. Eine schaftförmige Struktur, wie sie bei Drehteilen oft vorkommt, oder eine Bohrung wird in den VRML-Dateien, die mit CATIA erzeugt werden, aus zwei Halbschalen zusammengesetzt. Die Vertices, die eine solche Halbschale bilden, befinden sich nur an den halbkreisförmigen Enden. In den Flächensets sind lediglich diese Endpunkte enthalten, was eine Zuhilfenahme der zuvor gefundenen Liniensets zur Folge hat. In den Liniensets sind nicht ganze Flächen, sondern nur die Konturlinien enthalten. Bei runden Strukturen, zu denen viele Punkte gehören, ist beispielsweise ein Halbkreis einer Zylinderschale in einem solchen Linienset enthalten und muss zur Erkennung einer runden Fläche nur mit dem Flächenset abgeglichen werden. Wenn also für die Konstruktion einer runden Fläche nur die Vertices der Endstrukturen definiert werden müssen, so sind für jedes Halbschalenelement zwei Liniensets zu finden, die genau die gleichen Vertices enthalten. Eine Bohrung oder ein Schaft werden folglich aus zwei Flächensets und vier Liniensets aufgebaut, die im Quelltext auf einfache Art und Weise detektiert werden können und die den Grundaufbau für runde Strukturen in VRML-Dateien bilden.

Ein weiterer Flächentyp, der im Programm nach den Rundungen untersucht wird, ist die nicht weiter spezifizierte ebene Fläche. All diese Flächen, die in einer beliebigen Orientierung im Raum liegen und deren Punkte sich alle in der gleichen Ebene befinden, werden mit Ausnahme von kreisförmigen Strukturen in Magenta eingefärbt. Abbildung 4.10 verdeutlicht am Quelltextauszug des Programmes vrml.py wie die Überprüfung abläuft. In der Liste mit dem Namen 'f' sind alle Flächensets abgespeichert, die im Folgenden auf ihre Lage in einer Ebene überprüft werden sollen. Die Liste 'f' ist dabei wie ein Dreifachtupel aufgebaut, auf dessen erster Ebene sich die Flächensets befinden, die wiederum aus vielen Punkten bzw. Vertices auf der zweiten Ebene aufgebaut sind, und auf der untersten Ebene die drei Knotenkoordinaten beinhalten. In der äußersten Schleife werden somit alle Flächensets nacheinander aufgerufen und in der nächsten Schleife eingehend untersucht. Wenn das Flächenset als in einer Ebene liegend identifiziert wird, so ist es ganz am Ende an die Liste 'h' anzuhängen.

Die eigentliche Untersuchung erfolgt durch die Bildung einer Ebene mithilfe dreier Punkte. Damit diese Punkte im Raum nicht zu nah beieinander liegen, werden im Quelltext der erste, der vierte und der vorvorletzte Punkt des Flächensets gewählt. Um nun zu gewährleisten, dass alle anderen Punkte des Flächensets auch in dieser willkürlich gebildeten Ebene liegen, muss in einer Schleife überprüft werden, ob die Ebenengleichung beim Einsetzen eines jeden Punktes erfüllt ist. Im Quelltext geschieht dies durch die Lösung eines Gleichungssystems mit der 'Least Square'-Methode und mit dem darauffolgenden Vergleich der Distanzergebnisse. Im unteren Teil von Abbildung 4.10 sind Rundungs- bzw. Abschneideoperationen zu langer Gleitpunktzahlen zu erkennen, die für einen Abstandsvergleich nötig sind. Da bei den Berechnungen Zahlen mit vielen Nachkommastellen entstehen, muss beim Vergleich eine Toleranz berücksichtigt werden, die auch bei kleinen Abweichungen die Fläche als eben identifiziert. Bei jedem Punkt, der in die Ausgangsebene gehört, wird ein Zähler um Eins erhöht. Wenn am Ende alle Punkte überprüft wurden, und der Zähler gleich der Anzahl aller Punkte im aktuellen Flächenset ist, wird die Fläche zur Einfärbung freigegeben.

Diese so neu entstandene Liste 'h', die nur noch Flächen in einer Ebene enthält, muss als nächstes untersucht werden, damit alle kreisförmigen Strukturen herausgefiltert werden können, um diese nicht in Magenta, sondern in Blau einzufärben. Das zur Ebenenidentifizierung angewendete Prinzip, das in einer Schleife alle Punkte nacheinander untersucht, wird auch an dieser Stelle angewendet. Auch dabei müssen beim Vergleich von Distanzen die Zahlen gerundet oder auch einige Nachkommastellen abgeschnitten werden, damit ein Toleranzbereich entsteht. Die Einfärbung erfolgt am Ende, wenn eine neue Liste ausschließlich mit Kreisflächen entstanden ist.

```
for i in range(len(f)):
    counter=0
    k=len(f[i])
    for j in range(len(f[i])-3):
        left=np.array([[float(f[i][4][0])-
        float(f[i][0][0]),float(f[i][k-3][0])-float(f[i][0][0])],
        [float(f[i][4][1])-float(f[i][0][1]),float(f[i][k-3][1])-
        float(f[i][0][1])],[float(f[i][4][2])-
        float(f[i][0][2]),float(f[i][k-3][2])-float(f[i][0][2])]])
        right=np.array([float(f[i][j+3][0])-float(f[i][0][0]),
        float(f[i][j+3][1])-float(f[i][0][1]),float(f[i][j+3][2])-
        float(f[i][0][2])])
        r,s=np.linalg.lstsq(left,right)[0]

        rstestx=float(r)*(float(f[i][4][0])-
        float(f[i][0][0]))+float(s)*(float(f[i][k-3][0])-
        float(f[i][0][0]))
        rstest2x=float(f[i][j+3][0])-float(f[i][0][0])
        rstesty=float(r)*(float(f[i][4][1])-
        float(f[i][0][1]))+float(s)*(float(f[i][k-3][1])-
        float(f[i][0][1]))
        rstest2y=float(f[i][j+3][1])-float(f[i][0][1])
        rstestz=float(r)*(float(f[i][4][2])-
        float(f[i][0][2]))+float(s)*(float(f[i][k-3][2])-
        float(f[i][0][2]))
        rstest2z=float(f[i][j+3][2])-float(f[i][0][2])
        if "e" in str(rstestx):
            rstestx=round(rstestx,6)
        if "e" in str(rstest2x):
            rstest2x=round(rstest2x,6)
        sp=str(rstestx).split(".")
        rstestx=float(sp[0]+"."+sp[1][:1])
        sp=str(rstest2x).split(".")
        rstest2x=float(sp[0]+"."+sp[1][:1])

        if "e" in str(rstesty):
            rstesty=round(rstesty,6)
        if "e" in str(rstest2y):
            rstest2y=round(rstest2y,6)
        sp=str(rstesty).split(".")
        rstesty=float(sp[0]+"."+sp[1][:1])
        sp=str(rstest2y).split(".")
        rstest2y=float(sp[0]+"."+sp[1][:1])

        if "e" in str(rstestz):
            rstestz=round(rstestz,6)
        if "e" in str(rstest2z):
            rstest2z=round(rstest2z,6)
        sp=str(rstestz).split(".")
        rstestz=float(sp[0]+"."+sp[1][:1])
        sp=str(rstest2z).split(".")
        rstest2z=float(sp[0]+"."+sp[1][:1])

        if rstestx==rstest2x and rstesty==rstest2y and
        rstestz==rstest2z:
            counter=counter+1
            if counter==len(f[i])-3:
                h.append(f[i])
```

Abbildung 4.10: Bestimmung von Flächen in einer Ebene

Am Beispiel der automatischen Kreiserkennung soll hier nicht der Quelltext angezeigt, sondern die theoretische mathematische Vorgehensweise exemplarisch beschrieben werden. In einer kreisförmigen Struktur besitzen alle Punkte den gleichen Abstand zum Kreismittelpunkt. Dieser muss als erstes ermittelt werden, um dann die Distanzen eines jeden Punktes überprüfen zu können. Sollten bei der Untersuchung verschiedene Abstände berechnet werden, so ist die Fläche weiterhin in die Kategorie der ebenen Flächen einzuordnen und kann in Magenta eingefärbt werden.

Zur Ermittlung einer Kreisfläche muss folglich zunächst der vermeintliche Mittelpunkt des Kreises gefunden werden. Das Programm macht sich dabei die Tatsache zunutze, dass ein Kreis und somit auch sein Mittelpunkt durch drei Punkte vollständig bestimmt werden können. Aus dem Flächenset werden also die drei Punkte P1, P2 und P3 beliebig gewählt.

$$P1 = \begin{pmatrix} P1_x \\ P1_y \\ P1_z \end{pmatrix}, P2 = \begin{pmatrix} P2_x \\ P2_y \\ P2_z \end{pmatrix}, P3 = \begin{pmatrix} P3_x \\ P3_y \\ P3_z \end{pmatrix}$$

Um den Normalenvektor der Ebene, in der die drei Punkte liegen, zu berechnen, müssen die folgenden zwei Vektoren, die die Distanzen zwischen dem ersten und dem zweiten sowie zwischen dem ersten und dem dritten Punkt bilden, im Kreuzprodukt in Vektorform miteinander verrechnet werden. Es entsteht ein Vektor, der senkrecht zur Ebene steht.

$$\vec{n} = \begin{pmatrix} P1_x - P2_x \\ P1_y - P2_y \\ P1_z - P2_z \end{pmatrix} \times \begin{pmatrix} P1_x - P3_x \\ P1_y - P3_y \\ P1_z - P3_z \end{pmatrix} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

Als nächstes werden zwei Geraden gebildet, die sich im Mittelpunkt des Kreises schneiden. Wenn angenommen wird, dass die drei Punkte ein Dreieck aufspannen, so muss zur Bestimmung des Kreismittelpunktes jeweils eine Gerade konstruiert werden, die sowohl durch den Kreismittelpunkt, als auch durch den Punkt geht, der genau auf der Mitte eines Dreiecksschenkels liegt.

Der Richtungsvektor der ersten Geraden lässt sich aus dem Kreuzprodukt zwischen Normalenvektor und erstem Richtungsvektor der Ebene berechnen.

$$\overrightarrow{r_1} = \begin{pmatrix} P1_x - P2_x \\ P1_y - P2_y \\ P1_z - P2_z \end{pmatrix} \times \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} r_{1x} \\ r_{1y} \\ r_{1z} \end{pmatrix}$$

Der Stützvektor ergibt sich aus dem ersten Punkt und der Hälfte des Dreiecksschenkels.

$$\overrightarrow{s_1} = \begin{pmatrix} P1_x \\ P1_y \\ P1_z \end{pmatrix} + 0{,}5 * \begin{pmatrix} P1_x - P2_x \\ P1_y - P2_y \\ P1_z - P2_z \end{pmatrix} = \begin{pmatrix} s_{1x} \\ s_{1y} \\ s_{1z} \end{pmatrix}$$

Werden beide Vektoren für die Konstruktion der Geradengleichung verwendet, ergibt sich die erste Gerade.

$$g_1 = \overrightarrow{s_1} + u * \overrightarrow{r_1}$$

Analog dazu wird auch die zweite Gerade erstellt, bei der jedoch der dritte Punkt den zweiten ersetzt.

$$\overrightarrow{r_2} = \begin{pmatrix} P1_x - P3_x \\ P1_y - P3_y \\ P1_z - P3_z \end{pmatrix} \times \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} r_{2x} \\ r_{2y} \\ r_{2z} \end{pmatrix}$$

$$\overrightarrow{s_2} = \begin{pmatrix} P1_x \\ P1_y \\ P1_z \end{pmatrix} + 0{,}5 * \begin{pmatrix} P1_x - P3_x \\ P1_y - P3_y \\ P1_z - P3_z \end{pmatrix} = \begin{pmatrix} s_{2x} \\ s_{2y} \\ s_{2z} \end{pmatrix}$$

$$g_2 = \overrightarrow{s_2} + v * \overrightarrow{r_2}$$

Wenn am Ende beide Geraden gleichgesetzt werden, so lässt sich der Schnittpunkt berechnen, der gleichzeitig den Kreismittelpunkt darstellt.

$$u * \begin{pmatrix} r_{1x} \\ r_{1y} \\ r_{1z} \end{pmatrix} - v * \begin{pmatrix} r_{2x} \\ r_{2y} \\ r_{2z} \end{pmatrix} = \begin{pmatrix} s_{2x} \\ s_{2y} \\ s_{2z} \end{pmatrix} - \begin{pmatrix} s_{1x} \\ s_{1y} \\ s_{1z} \end{pmatrix}$$

Wie bereits erwähnt, überprüft der Algorithmus die Abstände eines jeden Punktes zu dem berechneten Mittelpunkt. Sollten alle Distanzen in einem gewissen Toleranzbereich gleich sein, wird das Flächenset als Kreis identifiziert und blau gefärbt.

Wurden von einer Datei alle Sets untersucht, erstellt das Programm pro Ausgangsbauteil im CATPart- bzw. VRML-Format jeweils eine kolorierte VRML-Datei und eine Textdatei, die Informationen über die Anzahl der gefärbten Flächen beinhaltet. Diese Textdatei wird später bei der Erstellung einer Signatur wieder aufgegriffen und weiterverarbeitet.

Wenn die Verarbeitung einer CATPart-Datei abgeschlossen ist, springt das Programm in einer Schleife zum nächsten Bauteil im Eingabepfad und führt alle Schritte erneut aus. Auch das hier beschriebene Programm vrml.py läuft nach der Spezifikation der Ein- und Ausgabeparameter gänzlich automatisch ab.

Evaluation

In Abbildung 4.11 sind drei Beispielteile unterschiedlicher Komplexität und in verschiedenen Ausrichtungen abgebildet. In der oberen Zeile sind jeweils die Ausgangsteile im CATPart-Format zu sehen. Die mittlere Zeile beinhaltet die VRML-Dateien, in denen auch die Vernetzung zu erkennen ist. Das Ergebnis der eingefärbten Teile wird im unteren Bereich angezeigt.

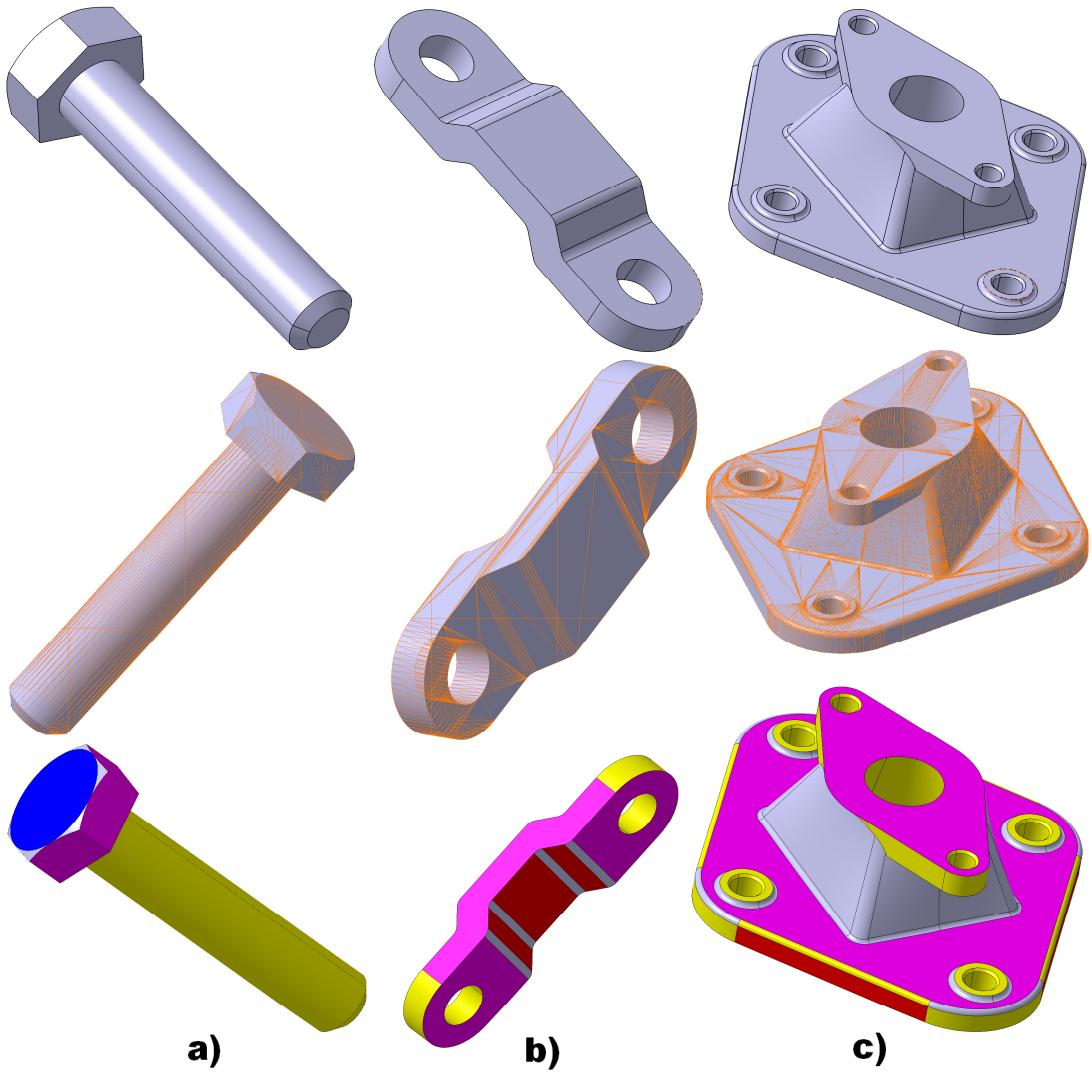

Abbildung 4.11: Drei Arbeitsschritte von vrml.py an Beispielteilen

Im Großen und Ganzen kann geschlussfolgert werden, dass bei der Einfärbung der CAD-Modelle ähnlich wie in Abbildung 4.11 den meisten Flächen eine bestimmte Kategorie zugewiesen werden kann. Besonders die hier untersuchten ingenieurstechnischen Teile bestehen oft aus großen Teilflächen, die für sich betrachtet eine simple Struktur besitzen. Dennoch werden meist kleine unwichtige Flächen, wie z.B. Verrundungen, nicht einer bestimmten Farbe zugewiesen und bleiben grau.

Auch die Anwendbarkeit des Programmes auf große Datenmengen erweist sich als praktikabel. Zwar hängt die Laufdauer stark von der Anzahl der Flächen in einem Bauteil ab, jedoch spielt die geometrische Größe keine Rolle. Da die gesamte Untersuchung der VRML-Dateien auf Basis des Quelltextes abläuft, kann auch die Einfärbung durch die Definition bestimmter Attribute automatisch durchgeführt und in einer geänderten VRML-Datei abgespeichert werden. Eine weitere Untersuchung dieser kolorierten VRML-Dateien erfolgt bei der Erstellung von Signaturen in Kapitel 5.2.

4.6 IGES-Analyse

Konzept

Ebenso wie die in Kapitel 2.2.5 beschriebenen VRML-Dateien eignen sich auch Bauteile im IGES-Format zur späteren Signaturerstellung. Im Gegensatz zum Einfärbungskonzept des VRML-Ansatzes soll an dieser Stelle allerdings nicht eine Untersuchung der einzelnen Dateibestandteile bzw. zweidimensionaler Flächen vollzogen werden, sondern nur eine Übertragung vom CATPart- ins IGES-Format stattfinden (vgl. Kapitel 2.2.6). Somit werden die Bauteile für die Signaturerstellung, die in Kapitel 5.3 eingehend erläutert wird, vorbereitet. Das Grundkonzept gleicht der VRML-Methodik, soll jedoch nicht auf genau die gleiche Art und Weise ablaufen. Da zur Vorbereitung keine weiteren Schritte nötig sind, soll in diesem Kapitel ausschließlich die Umwandlung von CATIA-eigenen Bauteilen ins neutrale IGES-Format vorgestellt werden.

Implementierung

Das Programm iges.py setzt die Umwandlung proprietärer CATPart-Dateien ins IGES-Format mithilfe der Makroprogrammierung um. In einer Schleife werden alle Bauteile im Eingabepfad nacheinander in CATIA geöffnet und sofort nach der Übertragung ins IGES-Format wieder geschlossen. Das Programm endet erst nach dem Durchlauf aller Eingabebauteile und erstellt für jedes Originalteil jeweils eine IGES-Datei im Ausgabepfad.

Evaluation

Auch dieses Programm läuft nach der Definition der Ein- und Ausgabepfade gänzlich automatisch ab und eignet sich deshalb auch für große Datenmengen. Das Grundprinzip lässt sich auch auf andere CAE-Systeme übertragen, da diese meist die Möglichkeit besitzen ihre proprietären Daten in herstellerneutrale Formate zu übertragen. Auch die Ausführzeit wird kaum von der Größe und Komplexität der Bauteile beeinflusst.

Abbildung 4.12 zeigt auf der linken Seite ein Beispielbauteil im originären CAT-Part-Format und auf der rechten Seite das gleiche Bauteil als IGES-Datei. Um die

Struktur aus zweidimensionalen Flächen im dreidimensionalen Raum zu verdeutlichen wurde eine Fläche unsichtbar gemacht, so dass ein Blick ins Innere des Bauteils ermöglicht wird.

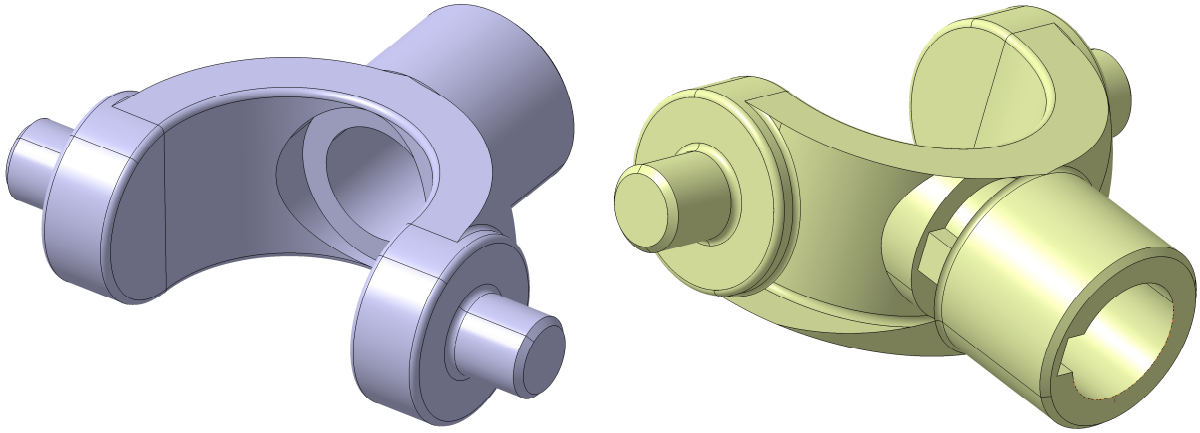

Abbildung 4.12: Beispielbauteil als CATPart- und IGES-Datei

Kapitel 5. Informationsaufbereitung

5.1 Attribute-Fingerprint

Konzept

Alle in Kapitel 4 gesammelten Informationen, die aus den verschiedenen Dateitypen extrahiert wurden, sollen in diesem Kapitel für die spätere Weiterverarbeitung aufbereitet werden. In allen drei Unterkapiteln geschieht dies in Form von Bauteilsignaturen bzw. Fingerprints, die den Zweck haben eine einfach zu vergleichende Essenz zu bilden, die möglichst viele Charaktereigenschaften des Bauteils in sich vereint, so dass die spätere Untersuchung im Zuge der Klassifikation oder der Ähnlichkeitssuche textbasiert und so schnell wie möglich ablaufen kann. Dennoch unterscheiden sich die drei Ansätze zur Erstellung von Fingerprints maßgeblich. Während in den Kapiteln 5.2 und 5.3 aus den zuvor gewonnenen Informationen Adjazenzmatrizen gebildet werden, wird hier ein anderes Konzept vorgestellt.

Die Grundidee besteht darin alle Daten aus den Kapiteln 4.1 bis 4.5 zu vereinen und daraus für jedes Bauteil eine aussagekräftige Signatur zu formen. Diese Signatur besteht in den folgenden Programmen aus der Bounding Box des ursprünglichen CAD-Modells sowie weiteren 13 Kategorien, die alle nur aus einem Kennwert bestehen, der den Gehalt der jeweiligen Kategorie im Modell widerspiegelt. Jede Kategorie kann dabei höchstens den Wert '10' erreichen, was bedeutet, dass die wesentliche Eigenschaft dieses Bauteils durch diese Kategorie hinreichend beschrieben wird. Die hier folgenden Kategorien weisen auf eine nicht ganz eindeutige Beschreibung des Bauteils hin und sollen nur die wichtigsten Grundzüge beschreiben:

1. Größe in X-Richtung
2. Größe in Y-Richtung
3. Größe in Z-Richtung
4. Rund einfach
5. Rund komplex
6. Eckig einfach
7. Eckig komplex
8. Eben einfach
9. Eben komplex
10. Freiformflächen
11. Detailgenauigkeit (Kantenverrundungen, Fasen, Bedingungen)
12. Geometrische Primitive

13. Bohrungen und Gewinde
14. Komplexität der Skizzen
15. Wiederholungen
16. Zweidimensionale Flächen (ohne Dicke)

Während die absolute Größe eines jeden Bauteiles aus der Bounding Box abgelesen werden kann, sind alle anderen Kategorien für die Definition der Bauteiltopologie zuständig. Kategorie 4 zeigt beispielsweise den Gehalt an runden und einfachen Strukturen, aus denen das Bauteil aufgebaut ist. Wie in Abbildung 5.1 zu erkennen, werden für die Berechnung der Kategorie insbesondere die Anzahl der Wellen und Einfachheit der Skizzen berücksichtigt. Wenn z.B. in einer Skizze nur ein Kreis enthalten ist, der dann in die dritte Dimension extrudiert wird, so entsteht ein rundes einfaches Gebilde, das stark in diese Kategorie mit einfließt.

Die Kennzahlen aller anderen Kategorien werden mit dem gleichen Prinzip berechnet, wobei immer definiert werden muss, welche Bauteilfeatures in die Bestimmung mit einfließen. Werden etwa Freiformflächen betrachtet, so wird klar, dass alle komplexen Features, wie z.B. Splines, die nicht den anderen Kategorien zugeordnet werden können, dort ihre Verwendung finden.

Weiterhin wird Kategorie 11 aus der Detailgenauigkeit der Bauteile gebildet. Oft entscheidet sich bei der Konstruktion eines neuen Teiles die formbestimmende Gesamterscheinung durch nur wenige Features, vornehmlich Extrusionen und Wellen. Wenn später Details, wie z.B. Bohrungen oder Nuten eingearbeitet werden, so ändert sich die Grundform nur noch geringfügig.

Auch über die Komplexität der Skizzen (Kategorie 14) kann eine Aussage über das Erscheinungsbild des Bauteils getroffen werden. Ist das Bauteil beispielsweise nur aus zwei Skizzen aufgebaut, die jedoch viele geometrische Primitive (Kategorie 12) enthalten, so kann nach einer Analyse der verwendeten Skizzenelemente die endgültige Form geschlussfolgert werden. Auch Kategorie 15, die wiederverwendete Befehle, wie z.B. Spiegelungen, Symmetrien oder musterförmige Anordnungen, untersucht, oder Kategorie 16, in der der Gehalt an zweidimensionalen Flächen, die im Volumenmodell aufgedickt werden, gemessen wird, liefern wichtige Informationen über den Charakterzug eines jeden Bauteiles.

Implementierung

Das Programm attributes.py setzt das oben beschriebene Konzept um und sammelt vor der Berechnung der Kategorien alle Informationen die durch die Programme stl.py, voxel.py, read_tree.py und vrml.py gesammelt und in externen Textdateien

zwischengespeichert wurden. Die Ausgabe des berechneten Fingerprints erfolgt ebenfalls in einer Textdatei, was zur Folge hat, dass mit jedem ursprünglichen Bauteil drei Textdateien und zwei XML-Dateien geliefert werden müssen, aus denen am Ende nur eine Signaturdatei entsteht. Wie im Fließdiagramm zu erkennen, überspringt der Algorithmus die gesamte Bauteilrekonstruktion, wenn die dafür benötigten zweidimensionalen Bilder und technischen Zeichnungen nicht vorhanden sein sollten.

An dieser Stelle sollen für die spätere Fingerprintberechnung zuerst alle verfügbaren Features, die aus dem jeweiligen Bauteil zuvor gefiltert wurden, aufgelistet werden. Die Überschriften zeigen dabei an, aus welchem Programm die jeweiligen Features stammen.

STL

01. Anzahl der Flächen
02. Anzahl der Vertices
03. Anzahl der offenen Kanten
04. Anzahl der geschlossenen Kanten

Voxel

05. Anzahl der Voxel
06. Volumen eines Voxels in mm^3
07. Volumen des gesamten Teils in mm^3
08. Größe in X-Richtung in mm
09. Größe in Y-Richtung in mm
10. Größe in Z-Richtung in mm
11. Massenträgheitsmoment XX in $kg*m^2$
12. Massenträgheitsmoment XY in $kg*m^2$
13. Massenträgheitsmoment XZ in $kg*m^2$
14. Massenträgheitsmoment YX in $kg*m^2$
15. Massenträgheitsmoment YY in $kg*m^2$
16. Massenträgheitsmoment YZ in $kg*m^2$
17. Massenträgheitsmoment ZX in $kg*m^2$
18. Massenträgheitsmoment ZY in $kg*m^2$
19. Massenträgheitsmoment ZZ in $kg*m^2$

Strukturbaum

20. Größe in X-Richtung in mm
21. Größe in Y-Richtung in mm
22. Größe in Z-Richtung in mm
23. Massenträgheitsmoment XX in $kg*m^2$

24. Massenträgheitsmoment XY in kg*m²
25. Massenträgheitsmoment XZ in kg*m²
26. Massenträgheitsmoment YX in kg*m²
27. Massenträgheitsmoment YY in kg*m²
28. Massenträgheitsmoment YZ in kg*m²
29. Massenträgheitsmoment ZX in kg*m²
30. Massenträgheitsmoment ZY in kg*m²
31. Massenträgheitsmoment ZZ in kg*m²
32. Minimum Größe in mm
33. Medium Größe in mm
34. Maximum Größe in mm
35. Volumen in mm³
36. Oberfläche in mm²
37. Volumen/Oberfläche in mm
38. Material
39. Dichte in kg/m³
40. Masse in kg
41. Anzahl an Skizzen
42. Anzahl an Punkten
43. Anzahl an Linien
44. Anzahl an Kreisen
45. Anzahl an Splines
46. Anzahl an Ellipsen
47. Anzahl an Extrusionen
48. Anzahl an Wellen
49. Anzahl an Nuten
50. Anzahl an Taschen
51. Anzahl an Rippen
52. Anzahl an Slots
53. Anzahl an Bohrungen
54. Anzahl an Gewinden
55. Anzahl an Fasen
56. Anzahl an Kantenverrundungen
57. Anzahl an geometrischen Sets
58. Anzahl an Körpern
59. Anzahl an Koinzidenzen
60. Anzahl an Fixierungen
61. Anzahl an Konzentrizitäten

62. Anzahl an Tangentenstetigkeiten
63. Anzahl an Parallelismen
64. Anzahl an Perpendikularitäten
65. Anzahl an Symmetrien
66. Anzahl an Winkeln
67. Anzahl an Längen
68. Anzahl an Distanzen
69. Anzahl an Radien
70. Anzahl an equidistanten Punkten
71. Anzahl an Tranlationen
72. Anzahl an Rotationen
73. Anzahl an aufgedickten Flächen
74. Anzahl an Spiegelungen
75. Anzahl an Mehrfachflächen
76. Anzahl an automatischen Materialfüllungen
77. Anzahl an Bedingungen
78. Anzahl an geometrischen Elementen

Rekonstruktion

79. Rekonstruierte Größe in X-Richtung in mm
80. Rekonstruierte Größe in Y-Richtung in mm
81. Rekonstruierte Größe in Z-Richtung in mm
82. Rekonstruiertes Massenträgheitsmoment XX in kg*m^2
83. Rekonstruiertes Massenträgheitsmoment XY in kg*m^2
84. Rekonstruiertes Massenträgheitsmoment XZ in kg*m^2
85. Rekonstruiertes Massenträgheitsmoment YX in kg*m^2
86. Rekonstruiertes Massenträgheitsmoment YY in kg*m^2
87. Rekonstruiertes Massenträgheitsmoment YZ in kg*m^2
88. Rekonstruiertes Massenträgheitsmoment ZX in kg*m^2
89. Rekonstruiertes Massenträgheitsmoment ZY in kg*m^2
90. Rekonstruiertes Massenträgheitsmoment ZZ in kg*m^2
91. Rekonstruierte minimum Größe in mm
92. Rekonstruierte medium Größe in mm
93. Rekonstruierte maximum Größe in mm
94. Rekonstruiertes Volumen in mm^3
95. Rekonstruierte Oberfläche in mm^2
96. Rekonstruiertes Volumen/Oberfläche in mm
97. Rekonstruiertes Material

98. Rekonstruierte Dichte in kg/m³
99. Rekonstruierte Masse in kg
100. Anzahl der Bohrungen
101. Anzahl der Taschen

VRML

102. Anzahl der Dreiecksflächen
103. Anzahl der Viereckflächen
104. Anzahl der Sechsecksflächen
105. Anzahl der Flächen in einer Ebene
106. Anzahl der Kreisflächen
107. Anzahl der Rundungen
108. Anzahl der Freiformflächen
109. Anzahl der Knoten

Es bleibt zu erwähnen, dass die Umsetzung des Konzeptes nur eine Version darstellt, die jedoch sowohl um weitere Kategorien, als auch um die Berechnungsalgorithmen der Kategorien auf einfache Art und Weise erweitert bzw. verbessert werden kann. Abbildung 5.1 zeigt am Beispiel des Programmquelltextes die Berechnung der Kategorie 'Rund einfach'.

Im oberen Bereich werden vier Sonderfälle für einfache Drehteile behandelt, die alle zu einem Kategoriewert von '1' führen. Im unteren Bereich des Quelltextes wird anhand der entdeckten Features der Wert für die Kategorie mit einzelnen Punkten gefüllt und am Ende so limitiert, dass das Maximum ein Wert von '10' darstellt. Aus der oben dargestellten Auswahl an Features könnten noch viele weitere Berechnungsarten dieser ersten Kategorie mit einfließen. Ein ähnliches Prinzip gilt für alle weiteren Kategorien und soll an dieser Stelle nicht genauer spezifiziert werden.

Evaluation

Zur Bewertung der vorgeschlagenen Methode ist zu Anfang anzumerken, dass eine ähnliche Vorgehensweise in keiner der in Kapitel 3.8 vorgestellten Signaturerstellungen erwähnt wird. Die in Kapitel 2.8 beschriebenen professionellen PDM-Systeme verwenden zwar die im Strukturbaum definierten Features für die Suche der Bauteile, doch auch dort fließen sie nicht in eventuell verwendete Fingerprints mit ein.

Abbildung 5.2 zeigt zwei ähnliche Teile, deren Fingerprints in Tabelle 5.1 abgebildet sind.

```
roundsimple=0
if numsketches==1 and pointcounter==1 and circlecounter==1 and numpads==1
and circleareas==2 and roundareas==1 and linecounter==0:
    roundsimple=1
if numsketches==1 and pointcounter==0 and circlecounter==1 and numpads==1
and circleareas==2 and roundareas==1 and linecounter==0:
    roundsimple=1
if numsketches==1 and pointcounter==4 and circlecounter==0 and numpads==0
and numshafts==1 and circleareas==2 and roundareas==1 and linecounter==4
and parallelcounter==4:
    roundsimple=1
if numsketches==1 and pointcounter==3 and circlecounter==0 and numpads==0
and numshafts==1 and circleareas==2 and roundareas==1 and linecounter==4
and parallelcounter==4:
    roundsimple=1
if roundsimple==0:
    if numsketches>1 and circlecounter>=1 and circlecounter<=4:
        roundsimple=roundsimple+1
    if numsketches>1 and circlecounter>=4:
        roundsimple=roundsimple+2
    if numsketches>2 and circleareas>=1 and circleareas<=4:
        roundsimple=roundsimple+1
    if numsketches>2 and circleareas>4:
        roundsimple=roundsimple+2
    if numsketches>1 and roundareas>=1 and roundareas<=4:
        roundsimple=roundsimple+1
    if numsketches>4 and roundareas>=5 and roundareas<=7:
        roundsimple=roundsimple+2
    if numsketches>4 and roundareas>8:
        roundsimple=roundsimple+3
    if numshafts>=1 and numshafts<=4:
        roundsimple=roundsimple+1
    if numshafts>4:
        roundsimple=roundsimple+2
    if concentcounter>3:
        roundsimple=roundsimple+1
    if tangentcounter>3:
        roundsimple=roundsimple+1
    if radiuscounter>3:
        roundsimple=roundsimple+1
    if roundareas>3 and roundareas<=10:
        roundsimple=roundsimple+1
    if roundareas>10:
        roundsimple=roundsimple+2
    if numfillets>3:
        roundsimple=roundsimple+1
if roundsimple>10:
    roundsimple=10
```

Abbildung 5.1: Berechnung der Kategorie Rund einfach

Da beide Bauteile im CAD-Programm mit genau den gleichen Befehlen erzeugt worden sind, bei denen sich nur der Zahlenwert für die Länge unterschieden hat, besitzen sie bei allen Kategorien außer der Größe in Z-Richtung genau die gleichen Zahlenwerte und somit sich stark ähnelnde Fingerprints. Anhand dieses Beispielergebnisses kann die Methodik gegenüber den konventionellen Signaturerstellungen, wie sie von den PDM-Systemen bei der Ähnlichkeitssuche genutzt werden, als überlegen bewertet werden. Die Tests, die im Zuge studentischer Projekte durchgeführt wurden

(vgl. Kapitel 2.8), haben gezeigt, dass ein Größenunterschied zweier Bauteile mit genau der gleichen Topologie zu sehr schlechten Vergleichswerten führt, da das Konzept der dreidimensionalen Ähnlichkeitssuche an dieser Stelle versagt. Die beiden Hülsen in Abbildung 5.2 unterscheiden sich zwar in Größe und Gewicht maßgeblich, besitzen aber dennoch die gleiche Topologie, was bei einem direkten Fingerprintvergleich sofort deutlich wird.

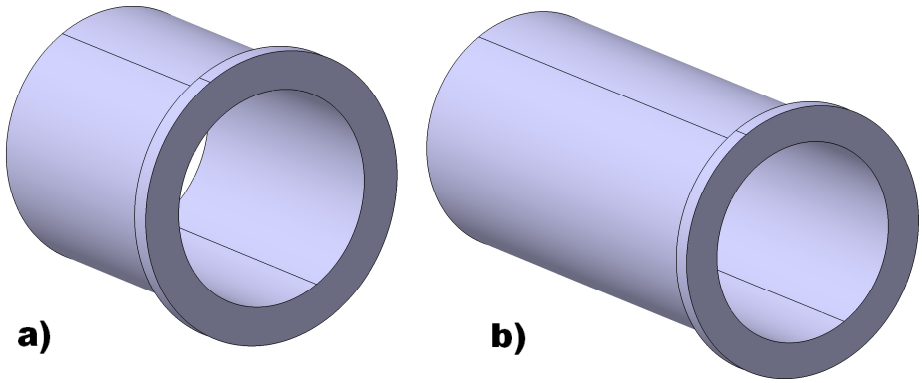

Abbildung 5.2: Zwei Beispielteile zur Fingerprinterstellung

Dennoch lässt sich schlussfolgern, dass die Funktionalität des umgesetzten Konzeptes stark von den zu untersuchenden Bauteilen abhängt und die Features bei der grundlegenden Formgebung der Bauteile eine unterschiedliche Rolle spielen.

Kategorie	a)	b)
1. Größe X in mm	76,3	76,3
2. Größe Y in mm	76,3	76,3
3. Größe Z in mm	70,35	134,4
4. Rund einfach	2	2
5. Rund komplex	1	1
6. Eckig einfach	0	0
7. Eckig komplex	0	0
8. Eben einfach	1	1
9. Eben komplex	0	0
10. Freiform	0	0
11. Details	0	0
12. Primitive	1	1
13. Gewinde	0	0
14. Skizzen	0	0
15. Wiederholungen	0	0
16. 2D	0	0

Tabelle 5.1: Die Fingerprints beider Hülsen

So kann einerseits aus einer einzigen Skizze, deren Kontur extrudiert oder rotiert wird, das maßgebliche Gesamterscheinungsbild entstehen. Andererseits können viele weitere Features, wie etwa Kantenverrundungen, Bohrungen, Fasen oder Nuten einen Großteil des Strukturbaums ausmachen, aber dennoch nicht formgebend sein. Die Teile in Abbildung 5.2 sind ein eindeutiges Beispiel für die Veränderung der Kontur eines Bauteiles durch nur einen einzigen Zahlenwert. Aber da die Topologie und die beteiligten Befehle unverändert bleiben, ist der Attribute-Fingerprint für die Anwendung in einer Suchmaschine oder einem Klassifikationsalgorithmus gut geeignet (vgl. Kapitel 6.1.1 und Kapitel 6.2.1). Als negative Eigenschaft ist die relativ aufwendige Vorbereitungszeit zu erwähnen.

Weiterhin ist anzumerken, dass die hier vorgeschlagene Methode nur eine Möglichkeit darstellt alle gesammelten Informationen zu verwerten. Einerseits können für den Fingerprint weitere Kategorien erarbeitet werden, die die Signatur verfeinern, andererseits ist es auch für den Anwender auf einfache Art und Weise möglich die Berechnungsmethoden der einzelnen Kategorien zu überarbeiten. Mit allen gesammelten Informationen sind zusätzlich weitere Ansätze zur Aufbereitung und Weiterverarbeitung denkbar.

5.2 VRML-Fingerprint

Konzept

Die Vorgehensweise zur Signaturerstellung auf Basis von VRML-Dateien greift die in Kapitel 4.5 beschriebenen eingefärbten Modelle wieder auf, analysiert und verarbeitet sie weiter. Die Grundidee ist es, die unterschiedlichen Flächendefinitionen, die aus der zuvor erfolgten Einfärbung hervorgehen, zur Erstellung eines graphenförmigen Drahtgittermodells zu verwenden, das sowohl die Flächentypen, als auch deren Verbindungen im dreidimensionalen Raum widerspiegelt. Analog zur Entwicklung herkömmlicher Fingerprints, wird auch hier das Bauteil auf seine wichtigsten Eigenschaften, die der Beschreibung dienen und die als entscheidende Merkmale für einen späteren Vergleich verwendet werden können, reduziert. Das so entstehende Gittermodell kann gänzlich in einer Art Adjazenzmatrix dargestellt und somit in Textform abgespeichert werden. Die für die Bauteilsuche bzw. -klassifikation nötige Analyse kann infolgedessen in Form von regulären Ausdrücken ablaufen, womit eine Anwendung des Konzeptes auch auf große Datenmengen ermöglicht wird.

Des Weiteren soll die Idee der Flächenausdünnung vorgestellt werden. Wie bereits in Kapitel 5.1 erwähnt, sind die Features, aus denen das Bauteil aufgebaut ist und die im Strukturbaum abgespeichert werden, in formgebend und nicht formgebend einzuteilen. In der hier vorzustellenden Methode werden auf Basis dieser Idee alle kleineren Flächen, von denen angenommen wird, dass sie für die Bauteilrepräsentation keine große Rolle spielen, aus der Adjazenzmatrix bzw. der Signatur entfernt. Bauteile, die sich bis auf kleine Details gleichen, erhalten so den gleichen oder mindestens einen sehr ähnlichen Fingerprint und können später als übereinstimmend erkannt werden.

Implementierung

Die Methode der normalen Adjazenzmatrixerstellung wird im Programm vrml_matrix.py umgesetzt. Das Programm simple_vrml_matrix.py läuft sehr ähnlich ab und liefert die ausgedünnte Matrix. Beiden Programmen ist gemein, dass sie die kolorierten VRML-Dateien als Eingabe verlangen und pro Bauteil eine Textdatei ausgeben, in der die Adjazenzmatrix definiert ist. Im Folgenden soll der Programmablauf und die Ergebnisse exemplarisch anhand des Beispielbauteils in Abbildung 5.3 demonstriert werden. Auf der linken Seite ist das ursprüngliche CAD-Modell im

CATPart-Format abgebildet, während auf der rechten Seite die verschiedenen Flächeneinfärbungen zu erkennen sind. Die Untersuchung der VRML-Dateien läuft ähnlich wie bei dem in Kapitel 4.5 vorgestellten Programm vrml.py durch eine Analyse des Quelltextes ab. Dieser wird in einer Schleife zeilenweise durchlaufen, um am Anfang alle Flächensets zu extrahieren. Gleichzeitig erfolgt auch die Unterteilung in die verschiedenen Flächentypen, deren Einfärbung zuvor im Quelltext vom Programm vrml.py definiert wurde.

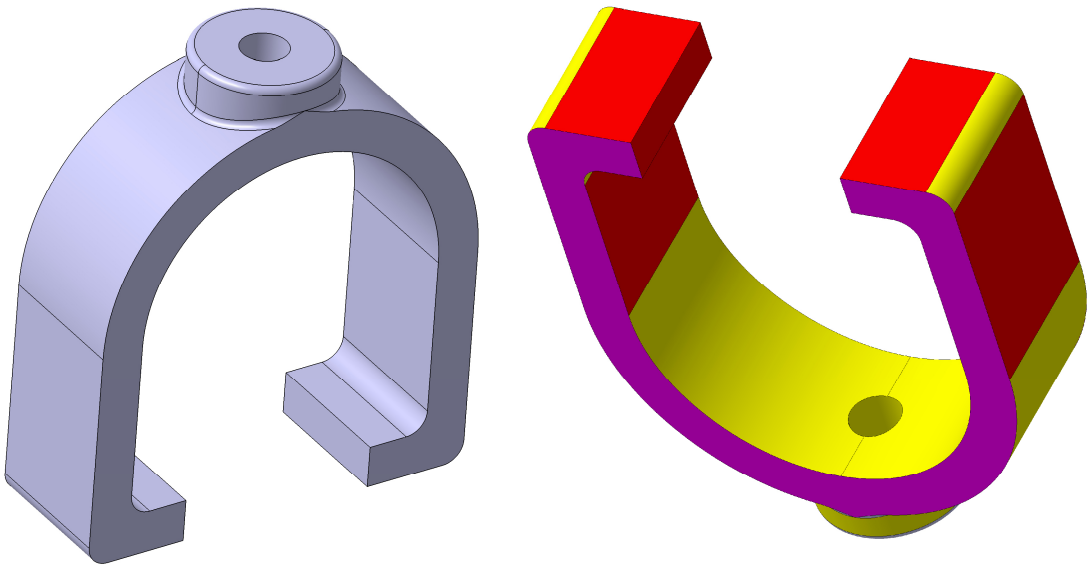

Abbildung 5.3: Beispielbauteil als CATPart und kolorierte VRML-Version

Im nächsten Schritt wird jede Fläche einzeln untersucht und sowohl der Schwerpunkt der Bounding Box als auch die Verbindungen zu allen Nachbarflächen berechnet. Von den Flächen, die als Nachbarflächen identifiziert werden, wird ebenfalls der Mittelpunkt der Bounding Box berechnet, um den Abstand bestimmen zu können. Nachdem in einer Schleife alle Flächen untersucht, auf die Verbindungen zu ihren nächsten Nachbarn getestet und die Abstände gemessen wurden, schreibt das Programm alle gewonnenen Informationen in eine Textdatei und speichert diese im zuvor gewählten Ausgabeverzeichnis ab. Danach erfolgt die Analyse der nächsten VRML-Datei.

Abbildung 5.4 zeigt die Textdatei, die aufgrund der Prüfung des Beispielbauteiles aus Abbildung 5.3 entstanden ist. Wie an den nummerierten Zeilen zu erkennen, besteht das Bauteil aus 30 Flächen, deren Typ entsprechend eingefärbt jeweils im ersten Eintrag der eckigen Klammer angegeben wird. Die darauffolgenden drei Werte bestimmen die X-, Y- und Z-Koordinaten des Schwerpunktes der Bounding Box (violett) und erst danach folgen im Wechsel die eigentlichen Flächenverbindungen (grün) mit den Distanzen (blau). Besonders auffällig sind im Beispielteil die ebenen Flächen 23 und 24, die über besonders viele Verbindungen bzw. Nachbarflächen verfügen.

```
00[four, -14.0, -28.5, 0.0, 13, 6.485, 24, 32.43, 6, 4.069, 23, 32.43]
01[four, 14.0, -28.5, 0.0, 9, 6.485, 23, 32.43, 24, 32.43, 4, 4.069]
02[round, 21.0, -24.0, 0.0, 4, 4.506, 23, 32.71, 3, 12.83, 24, 32.71]
03[four, 22.5, -11.25, 0.0, 2, 12.83, 24, 26.67, 16, 25.15, 23, 26.67]
04[four, 16.75, -25.5, 0.0, 2, 4.506, 24, 31.31, 1, 4.069, 23, 31.31]
05[round, -21.0, -24.0, 0.0, 7, 12.83, 23, 32.71, 6, 4.506, 24, 32.71]
06[four, -16.75, -25.5, 0.0, 5, 4.506, 23, 31.31, 0, 4.069, 24, 31.31]
07[four, -22.5, -11.25, 0.0, 5, 12.83, 24, 26.67, 17, 25.15, 23, 26.67]
08[round, 27.0, -30.0, 0.0, 10, 15.82, 24, 40.86, 9, 7.403, 23, 40.86]
09[four, 19.75, -31.5, 0.0, 8, 7.403, 23, 37.68, 1, 6.485, 24, 37.68]
10[four, 28.5, -14.25, 0.0, 8, 15.82, 24, 32.98, 20, 30.77, 23, 32.98]
11[round, -27.0, -30.0, 0.0, 13, 7.403, 24, 40.86, 12, 15.82, 23, 40.86]
12[four, -28.5, -14.24, 0.0, 11, 15.82, 24, 32.98, 25, 30.77, 23, 32.98]
13[four, -19.75, -31.5, 0.0, 11, 7.403, 24, 37.68, 0, 6.485, 23, 37.68]
14[round, 0.20, 27.87, -0.86, 15, 1.776, 17, 20.20, 16, 19.98, 28, 5.692]
15[round, -0.20, 27.87, 0.86, 14, 1.776, 16, 20.20, 17, 19.98, 28, 5.692]
16[round, 11.25, 11.25, 0.0, 15, 20.20, 14, 19.98, 17, 22.5, 24, 19.40, 3,
25.15, 23, 19.40]
17[round, -11.24, 11.25, 0.0, 14, 20.20, 15, 19.98, 16, 22.5, 23, 19.40, 7,
25.15, 24, 19.40]
18[free, 5.31, 27.97, -4.99, 24, 29.87, 20, 18.50, 19, 9.999, 29, 11.49,
30, 5.664, 21, 10.62]
19[free, 5.31, 27.97, 4.99, 23, 29.87, 22, 10.62, 29, 5.664, 18, 9.999, 30,
11.49, 20, 18.50]
20[round, 16.46, 14.07, 0.0, 19, 18.50, 18, 18.50, 24, 24.46, 10, 30.77,
23, 24.46]
21[free, -5.31, 27.97, -4.99, 24, 29.87, 25, 18.50, 22, 10.0, 29, 11.49,
30, 5.664, 18, 10.62]
22[free, -5.31, 27.97, 5.0, 23, 29.87, 25, 18.50, 21, 10.0, 29, 5.664, 30,
11.49, 19, 10.62]
23[plane, 0.0, -1.0, 10.0, 19, 29.87, 20, 24.46, 10, 32.98, 8, 40.86, 9,
37.68, 1, 32.43, 4, 31.31, 2, 32.71, 3, 26.67, 16, 19.40, 17, 19.40, 7,
26.67, 5, 32.71, 6, 31.31, 0, 32.43, 13, 37.68, 11, 40.86, 12, 32.98, 25,
24.46, 22, 29.87, 29, 31.33]
24[plane, 0.0, -1.0, -10.0, 18, 29.87, 20, 24.46, 10, 32.98, 8, 40.86, 9,
37.68, 1, 32.43, 4, 31.31, 2, 32.71, 3, 26.67, 16, 19.40, 17, 19.40, 7,
26.67, 5, 32.71, 6, 31.31, 0, 32.43, 13, 37.68, 11, 40.86, 12, 32.98, 25,
24.46, 21, 29.87, 30, 31.33]
25[round, -16.46, 14.07, 0.0, 21, 18.50, 22, 18.50, 23, 24.46, 12, 30.77,
24, 24.46]
26[free, 0.0, 33.0, -4.99, 28, 5.024, 27, 9.999, 29, 10.45, 30, 3.063]
27[free, 0.0, 33.0, 4.99, 28, 5.024, 26, 9.999, 29, 3.063, 30, 10.45]
28[round, 0.0, 33.5, 0.0, 14, 5.692, 15, 5.692, 27, 5.024, 26, 5.024]
29[round, 0.0, 29.93, 4.99, 27, 3.063, 26, 10.45, 30, 10.0, 18, 11.49, 19,
5.664, 22, 5.664, 23, 31.33, 21, 11.49]
30[round, 0.0, 29.93, -4.99, 26, 3.063, 27, 10.45, 29, 10.0, 21, 5.664, 22,
11.49, 18, 5.664, 24, 31.33, 19, 11.49]
```

Abbildung 5.4: Textdatei des Beispielbauteils

Mithilfe des Python-Moduls 'Visual Python' besteht weiterhin die Möglichkeit, die erhaltene Textdatei im dreidimensionalen Raum zu visualisieren. In Abbildung 5.5 ist eine solche Verbildlichung der Textdatei aus Abbildung 5.4 zu sehen. Die Flächenschwerpunkte werden anhand kleiner Kugeln in den entsprechenden Farben dargestellt. Alle Verbindungen sind durch weiße Querstreben, die auf die Flächenmittelpunkte gerichtet sind, umgesetzt. Am Beispielbauteil kann demonstriert werden, dass die Form des Originalmodells auch in der Visualisierung noch immer hinreichend zu

erkennen ist. Die beiden bereits erwähnten magentafarbenen Flächen bilden mit ihren vielen Verbindungen die Basis, während die kurzen Abstände zwischen den grauen Freiformflächen im oberen Bereich auf relativ kleine für die Formgebung unbedeutende Flächen hindeuten.

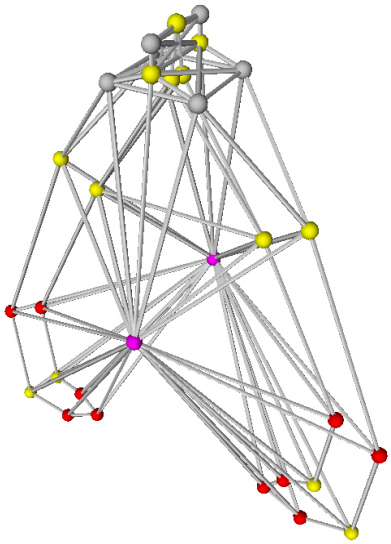

Abbildung 5.5: Visualisierung der Textdatei

Das Programm simple_vrml.py läuft im Prinzip auf die gleiche Art und Weise ab, wie vrml_matrix.py. Da nur die größten formgebenden Flächen berücksichtigt werden sollen, untersucht das Programm auch die Flächenmaße. Diese werden durch die Diagonale der Bounding Box angenähert und nachdem die größte Fläche im Modell identifiziert wurde, werden alle Flächen und Verbindungen gelöscht, die kleiner als ein zuvor bestimmter Prozentwert sind. Abbildung 5.6 zeigt die fertige Textdatei nach der Ausdünnung mit einem Schwellwert von 40%. Das bedeutet, dass alle Flächen entfernt wurden, deren Größe weniger als 40% der größten Fläche betrug. Da auch die Verbindungen zu den gelöschten Flächen entfernt wurden, kann es geschehen, dass in der Visualisierung Bereiche entstehen, die nicht zusammenhängen.

```
10[four, 28.5, -14.25, 0.0, 24, 32.98, 20, 30.77, 23, 32.98]
12[four, -28.5, -14.24, 0.0, 24, 32.98, 25, 30.77, 23, 32.98]
16[round, 11.25, 11.25, 0.0, 17, 22.5, 24, 19.40, 23, 19.40]
17[round, -11.24, 11.25, 0.0, 16, 22.5, 23, 19.40, 24, 19.40]
20[round, 16.46, 14.077, 0.0, 24, 24.46, 10, 30.77, 23, 24.46]
23[plane, 0.0, -1.0, 10.0, 20, 24.46, 10, 32.98, 16, 19.40, 17, 19.40, 12,
32.98, 25, 24.46]
24[plane, 0.0, -1.0, -10.0, 20, 24.46, 10, 32.98, 16, 19.40, 17, 19.40, 12,
32.98, 25, 24.46]
25[round, -16.46, 14.07, 0.0, 23, 24.46, 12, 30.77, 24, 24.46]
```

Abbildung 5.6: Ausgedünnte Textdatei

In Abbildung 5.7 ist noch einmal das dreidimensionale Drahtgittermodell zu sehen, das mithilfe von Visual Python aus der ausgedünnten Textdatei entstanden ist. Die Ausdünnung des Beispielteils führt an dieser Stelle zu einem einzigen zusammenhängenden Drahtgittermodell.

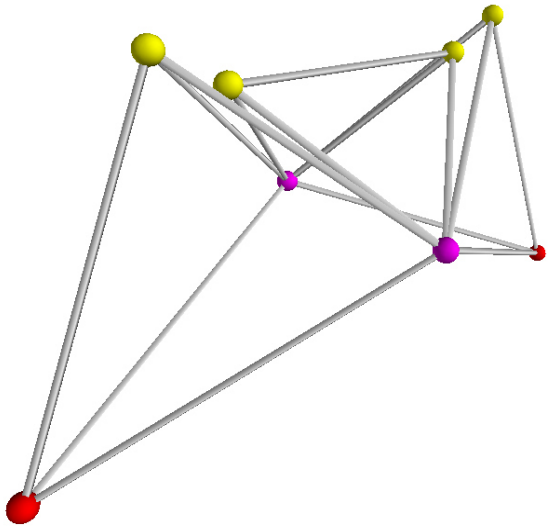

Abbildung 5.7: Visualisierung der ausgedünnten Textdatei

Nach der Erstellung sowohl der konventionellen als auch der ausgedünnten Textdatei, sind die Signaturen letztlich von der Geometrie unabhängig und liegen in Textform zur Weiterverarbeitung im Ausgabeverzeichnis bereit (vgl. Kapitel 6.1.2 und 6.2.2). Die Analyse dieser Fingerprints kann dann möglichst schnell erfolgen, was zu einer komfortablen Anwendung führt.

Evaluation

Da die VRML-Dateien durch ihre vorherige Einfärbung des Programmes vrml.py für die hier beschriebene Informationsaufbereitung schon präpariert sind und sowohl die Untersuchung der kolorierten Teile als auch die Programmausgabe ausschließlich textbasiert ablaufen, gestalten sich die Laufzeiten beider Programme auch für sehr große Datenmengen als praktikabel. Da das Programm simple_vrml_matrix.py neben dem Ablauf, der auch in vrml_matrix.py enthalten ist, noch weitere Berechnungen durchführen muss, entsteht eine etwas längere Ausführzeit.

Auch der Speicherplatzbedarf kann als positiv bewertet werden. Abhängig von der Komplexität und somit der Flächenzahl der Bauteile entstehen lediglich Textdateien, die verglichen mit der ursprünglichen Datenmenge, die zur Beschreibung des VRML-

bzw. CATPart-Modells nötig war, nur einen sehr geringen Anteil vom Festplattenspeicher ausmachen.

Allgemein ist anzumerken, dass die Signaturerstellung in dieser Form eine gute Lösung des Problems darstellt, komplexe Geometrien in verkürzter Form wiederzugeben. In Kapitel 6 werden die hier erstellten Fingerprints auf ihre Leistungsfähigkeit beim Formenvergleich sowie der Ähnlichkeitssuche und dem Clustering geprüft.

5.3 IGES-Fingerprint

Konzept

Das Prinzip der Signaturerstellung von VRML-Dateien soll an dieser Stelle auf das IGES-Format übertragen werden, wo es sehr ähnlich abläuft, aber zu unterschiedlichen Ergebnissen führt. Als Ausgangsmaterial werden die in Kapitel 4.6 vorbereiteten IGES-Dateien verwendet, die zuvor automatisch aus dem proprietären CATPart-Format erzeugt wurden. Vergleichbar mit der Analyse der VRML-Dateien läuft dann ebenfalls eine Untersuchung der Flächen und ihrer Verbindungen zueinander ab, mit dem Unterschied, dass die einzelnen Flächentypen vorher nicht spezifiziert werden. Auch die Werte werden nicht über den Quelltext der IGES-Dateien bestimmt, sondern mithilfe der Makroprogrammierung in CATIA gemessen.

Der Bauteilfingerprint liegt am Ende in Form einer textbasierten Matrix vor, die sowohl die Flächenverbindungen, als auch weitere Informationen, wie z.B. die Flächengröße und die Flächenschwerpunkte, enthält.

Des Weiteren läuft der Ansatz der Flächenausdünnung analog zu dem in Kapitel 5.2 ab. Die Idee ist es, alle Flächen zu entfernen, die eine kleinere Größe als ein anfangs vom Benutzer definierter Grenzwert besitzen. Auch dabei entsteht eine ausgedünnte Adjazenzmatrix, die am Ende in einer separaten Textdatei zur Weiterverarbeitung vorliegt.

Die gesamte Methodik wurde verbunden mit einer automatisierten Klassifikation von IGES-Dateien für eine Veröffentlichung in drei Bereiche eingeteilt [*RoW-15]. Neben der Erzeugung einer konventionellen Adjazenzmatrix, wurden sowohl die Eigenschaften der ausgedünnten Matrix, als auch einer weiteren Distanzmatrix im Hinblick auf das Clustering im nächsten Schritt untersucht.

Implementierung

Die Umsetzung des erklärten Konzeptes erfolgt in den Programmen iges_matrix.py und simple_iges_matrix.py, deren Quelltext sehr ähnlich aufgebaut ist. Beide Programme erstellen für jede zu untersuchende IGES-Datei jeweils eine Textdatei, die die Adjazenzmatrix in schriftlicher Form enthält. Diese kann allerdings auch in einem Tabellenkalkulationsprogramm geöffnet werden, wenn eine Darstellung in einer rasterförmigen Struktur gewünscht ist. Im Folgenden soll der Programmablauf anhand des Beispielteils in Abbildung 5.8 (untere Fläche ausgeblendet) vorges-

tellt werden. In den vorbereitenden Schritten, die im Programm ablaufen, wird als erstes eine Verbindung mit CATIA hergestellt, da für die Handhabung und die Analyse der IGES-Dateien die dem Benutzer zur Verfügung gestellten Messfunktionen automatisiert werden. Als erstes werden die Maße aller Flächen geprüft, so dass eine Sortierung nach der Größe vollzogen werden kann. In den ursprünglichen IGES-Dateien sind alle Flächen mit der gleichen Benennung versehen. Das Programm vergibt jedoch für jede Fläche einen neuen Namen, der aufgrund einer Nummerierung entsteht, was zur Folge hat, dass am Ende die Flächen der Größe nach sortiert in aufsteigender Reihenfolge vorliegen.

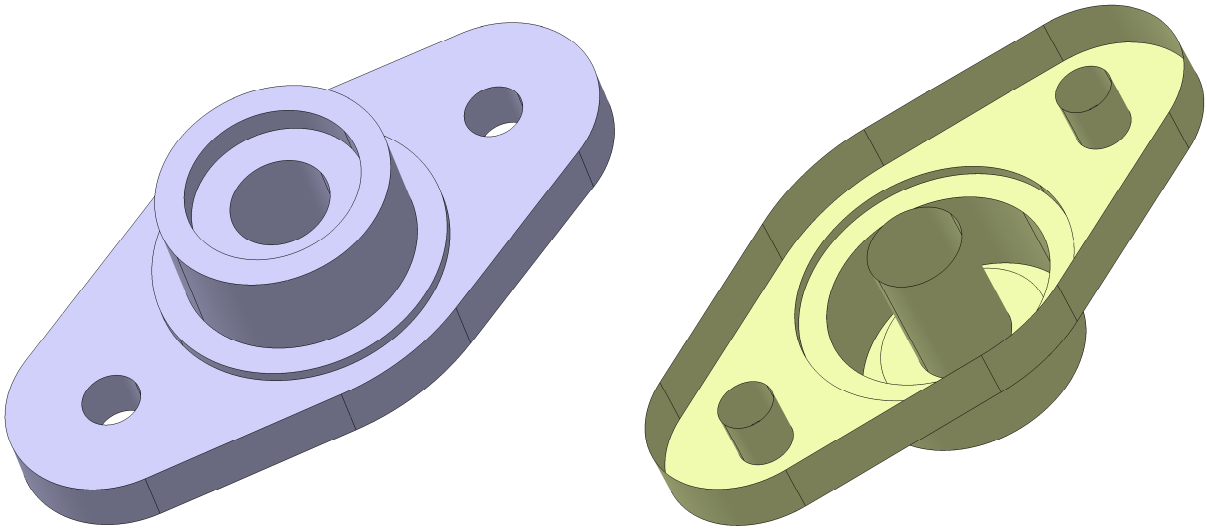

Abbildung 5.8: Beispielbauteil im CATPart- und IGES-Format

In mehreren darauffolgenden Schleifendurchläufen werden als nächstes mit der CATIA-eigenen 'SPAWorkbench' die Flächenschwerpunkte berechnet und die Verbindungen sowie Distanzen gemessen. Dabei unterscheiden sich die entstehenden Werte in einem wesentlichen Punkt von den vorangegangenen VRML-Textdateien. Während in Kapitel 5.2 ausschließlich die Distanzen zwischen den einzelnen benachbarten Flächenschwerpunkten ermittelt wurden, erfolgt hier eine Messung der Mindestabstände nicht benachbarter Flächen. Wie in Tabelle 5.2, die die Adjazenzmatrix des Beispielteils aus Abbildung 5.8 in gekürzter Form beinhaltet, zu erkennen ist, entsteht aus der automatischen Abstandmessung in CATIA ein Wert von Null, wenn zwei Flächen benachbart sind. Sollte keine direkte Verbindung zwischen zwei Flächen bestehen, wird ihr Mindestabstand in die Matrix eingetragen. So entsteht eine matrixförmige Struktur, die symmetrisch ist und da eine Distanz nicht zwischen der eigenen Fläche ermittelt werden kann, auf ihrer Diagonalen den unbestimmten Wert 'None' aufweist.

Als Überschrift sind die Flächennummern auf der Horizontalen sowie Vertikalen angegeben, so dass die Verbindungswerte auf den ersten Blick abgelesen werden können. In der viertletzten Spalte stehen zusätzlich die von CATIA ermittelten Größen in mm^2 sowie die Koordinaten der Flächenschwerpunkte.

Nachdem das Programm für ein Bauteil alle Schleifen durchlaufen hat, wird die Matrix in eine Ausgabetextdatei abgespeichert und kann ggf. visualisiert werden. Die Visualisierung des Beispielteils, die allgemein ebenfalls mit dem Modul 'Visual Python' umgesetzt wird, ist in Abbildung 5.9 zu erkennen. Da keine Unterscheidung zwischen den verschiedenen Flächentypen erfolgt, sind alle Schwerpunkte mithilfe von roten Kugeln dargestellt, die je nach gemessener Flächengröße über unterschiedliche Durchmesser verfügen. Hier besitzt die größte Fläche auf der Unterseite des Bauteils auch die meisten Verbindungen und stellt somit die Referenzfläche für die Ausdünnung in den folgenden Schritten dar.

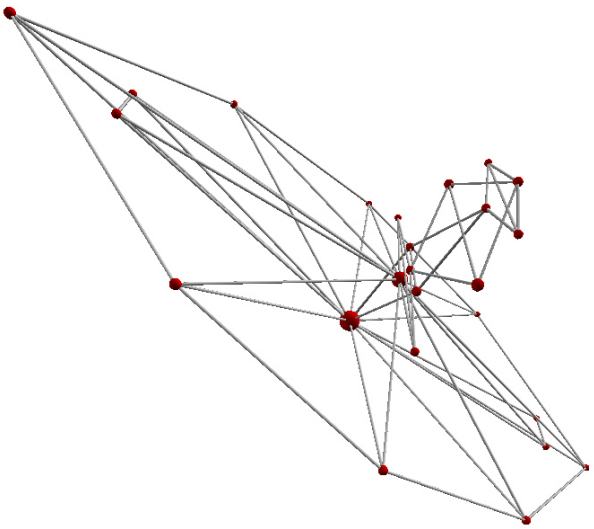

Abbildung 5.9: Visualisierung des Beispielteils

Im Programm simple_iges_matrix.py wird das gleiche Prinzip der Erstellung einer Adjazenzmatrix verfolgt. Ebenso wie das in Kapitel 5.2 vorgestellte Programm zur Simplifizierung der ursprünglichen Textdatei vorgeht, werden auch hier alle Flächen entfernt, die einen zuvor definierten Schwellwert, der aus empirischen Gründen normalerweise auf 40% gesetzt wird, unterschreiten. Da im Beispiel die bereits erwähnte Grundfläche viel größer ist, als alle anderen, wurde der Schwellwert hier nach einigen Tests auf 15% festgelegt. In Tabelle 5.3 ist die Matrix mit den übrig gebliebenen Flächen abgebildet. Diese sind außerdem nach ihrer Größe sortiert, was den späteren Vergleich im Zuge der Klassifikation vereinfacht (vgl. Kapitel 6.2.3).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Größe	X	Y	Z
0	None	0.0	8.20	11.5	11.5	6.5	6.5	11.5	16.5	16.5	16.5	40.03	27.45	28.62	27.45	0.0	667.59	0.00	5.41	12.50
1	0.0	None	5.0	5.0	5.0	0.0	0.0	13.92	16.40	16.40	18.02	36.74	21.21	21.21	21.21	25.0	479.87	0.00	0.00	25.00
2	8.20	5.0	None	0.0	0.0	0.0	0.0	18.0	18.68	18.68	20.61	34.84	22.36	22.36	22.36	30.0	549.77	0.00	0.00	30.00
3	11.5	5.0	0.0	None	0.0	5.0	5.0	0.0	5.0	5.0	5.38	28.60	24.08	30.72	24.08	12.0	1130.97	0.00	12.73	21.00
4	11.5	5.0	0.0	0.0	None	5.0	5.0	0.0	5.0	5.0	5.38	28.60	10.19	10.19	10.19	12.0	1130.97	0.00	-12.73	21.00
5	6.5	0.0	0.0	5.0	5.0	None	0.0	13.92	16.40	16.40	18.02	36.74	21.21	21.21	21.21	25.0	235.62	0.00	-9.55	27.50
6	6.5	0.0	0.0	5.0	5.0	0.0	None	13.92	16.40	16.40	18.02	36.74	29.58	32.86	29.58	25.0	235.62	0.00	9.55	27.50
7	11.5	13.92	18.0	0.0	0.0	13.92	13.92	None	0.0	0.0	2.0	23.62	5.38	5.38	5.38	12.0	706.86	0.00	0.00	12.00
8	16.5	16.40	18.68	5.0	5.0	16.40	16.40	0.0	None	0.0	0.0	23.53	22.5	32.78	22.5	10.0	157.08	0.00	15.92	11.00
9	16.5	16.40	18.68	5.0	5.0	16.40	16.40	0.0	0.0	None	0.0	23.53	5.0	5.0	5.0	10.0	157.08	0.00	-15.92	11.00
10	16.5	18.02	20.61	5.38	5.38	18.02	18.02	2.0	0.0	0.0	None	0.0	0.0	0.0	0.0	10.0	2904.58	0.00	-0.07	10.00
11	40.03	36.74	34.84	28.60	28.60	36.74	36.74	23.62	23.53	23.53	0.0	None	0.0	38.15	55.59	0.0	455.80	-53.56	0.00	5.00
12	27.45	21.21	22.36	24.08	10.19	21.21	29.58	5.38	22.5	5.0	0.0	0.0	None	0.0	18.0	0.0	381.58	-27.20	-22.89	5.00
13	28.62	21.21	22.36	30.72	10.19	21.21	32.86	5.38	32.78	5.0	0.0	38.15	0.0	None	0.0	0.0	182.82	0.00	-29.54	5.00
14	27.45	21.21	22.36	24.08	10.19	21.21	29.58	5.38	22.5	5.0	0.0	55.59	18.0	0.0	None	0.0	381.58	27.20	-22.89	5.00
15	0.0	25.0	30.0	12.0	12.0	25.0	25.0	12.0	10.0	10.0	10.0	0.0	0.0	0.0	0.0	None	4641.09	0.00	-0.04	0.00

Tabelle 5.2: Adjazenzmatrix des Beispielteils

Obwohl im Quelltext auch eine konventionelle Visualisierung mit Visual Python ermöglicht wird, zeigt Abbildung 5.10 die ausgedünnte IGES-Datei, aus der alle kleineren Flächen gelöscht wurden.

	15	10	4	3	7	0	Größe	X	Y	Z
15	None	10.0	12.0	12.0	12.0	0.0	4641.09	0.00	-0.04	0.00
10	10.0	None	5.385	5.385	2.0	16.5	2904.58	0.00	-0.07	10.00
4	12.0	5.385	None	0.0	0.0	11.5	1130.97	0.00	-12.73	21.00
3	12.0	5.385	0.0	None	0.0	11.5	1130.97	0.00	12.73	21.00
7	12.0	2.0	0.0	0.0	None	11.5	706.86	0.00	0.00	12.00
0	0.0	16.5	11.5	11.5	11.5	None	667.59	0.00	5.41	12.50

Tabelle 5.3: Ausgedünnte Adjazenzmatrix des Beispielteils

Die Möglichkeit einer Entstehung von mehreren Flächenbereichen, die nicht zwangsläufig miteinander verbunden sein müssen, hat sich an dieser Stelle entwickelt. Aus der Matrix ist abzulesen, dass beispielsweise Fläche 10 keinerlei Verbindungen mehr zu den anderen übrig gebliebenen Flächen aufweist.

Evaluation

Im direkten Vergleich mit dem in Kapitel 5.2 beschriebenen VRML-Fingerprint fällt eine große Ähnlichkeit der beiden Methoden auf. Dennoch unterscheiden sich die Vorgehensweisen bei der Erstellung beider Signaturen maßgeblich. Während die VRML-Dateien von den CAE-Programmen losgelöst betrachtet und dann nur noch textbasiert untersucht, koloriert und aufbereitet werden, ist für die Umsetzung des hier vorgestellten IGES-Ansatzes eine Software vonnöten, die die Flächenmessungen und Distanzberechnungen übernimmt.

Im direkten Vergleich ist folglich der VRML-Ansatz zu bevorzugen, da die gesamte Methodik schneller abläuft und ohne ggf. kostspielige professionelle Software auskommt.

Obwohl sich der Inhalt der beiden Adjazenzmatrizen aus dem VRML- und dem IGES-Fingerprint stark ähnelt, ist an dieser Stelle aufgrund der besseren Übersicht die IGES-Version zu bevorzugen. Es ist jedoch anzumerken, dass die genaue Formatierung in der folgenden Informationsweiterverarbeitung kaum eine Rolle spielt, da auch die Textdateianalyse automatisch abläuft.

Die bereits erwähnte Veröffentlichung, in der auch die Kapazitäten des IGES-Ansatzes zur Ähnlichkeitssuche und zur Klassifikation ausgewertet werden, hat ge-

zeigt, dass die Methode besonders für Bauteile, die nach der Ausdünnung nur noch über sehr wenige Flächen verfügen, geeignet ist. Am Beispiel verschiedener Schrauben und Drehteile, von denen nach der Simplifizierung nur noch der Schaft übrig bleibt, wird der Klassifikationsalgorithmus dort ausgetestet.

Weiterhin ist zu berücksichtigen, dass der Quelltext von IGES-Dateien neben der Definition von Flächen, auch über Volumenelemente verfügt. Im Vergleich zum VRML-Format, entstehen so eventuelle Möglichkeiten bei einer tiefergreifenden Analyse.

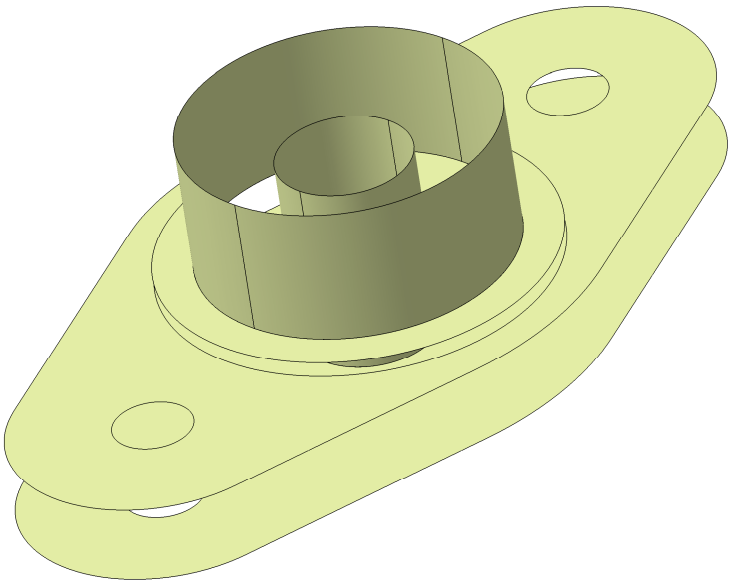

Abbildung 5.10: Ausgedünnte IGES-Datei

Kapitel 6. Informationsweiterverarbeitung

6.1 Suchmaschinen

Die Fingerprints bzw. Bauteilsignaturen, die in Kapitel 5 aus den zuvor gesammelten Informationen zusammengestellt wurden, sollen in diesem sechsten Kapitel weiterverarbeitet und sowohl zum Bauteilvergleich in Suchmaschinen, als auch zur Einteilung der Bauteile in Kategorien im Zuge der Klassifikation zur Anwendung kommen. Verglichen mit den wissenschaftlichen Veröffentlichungen in den Kapiteln 3.10 und 3.11, müssen auch hier Methoden gefunden werden, mit denen es möglich ist zwei Bauteile bzw. zwei Signaturen miteinander zu vergleichen.

In diesem Kapitel 6.1, das verschiedene Ansätze einer Suchmaschine vorstellt, soll im Detail auf den Attribute- und auf den VRML-, nicht jedoch auf den IGES-Fingerprint eingegangen werden. Dieser wird erst wieder in Kapitel 6.2 erwähnt, in dem die Möglichkeiten zur Klassifikation erörtert werden.

Wie bereits in Kapitel 1.4 erwähnt, wurden alle hier beschriebenen Algorithmen an einer Grundgesamtheit von ca. 100 Bauteilen getestet (vgl. Abbildung 1.1). Insbesondere die Klassenbildung in Kapitel 6.2 lässt sowohl die Übereinstimmung, als auch die Varianz der enthaltenen Bauteile erkennen.

6.1.1 Attribute-Suchmaschine

Konzept

Für die Erstellung einer Suchmaschine, die die Eigenschaften zweier Bauteile miteinander vergleicht und Ähnlichkeiten feststellt, sollen die Fingerprints, die die in Kapitel 5.1 beschriebenen Kategorien enthalten, wieder aufgegriffen und weiterverarbeitet werden. Ähnlich wie bei anderen Suchmaschinen, die beispielsweise Internetseiten durchsuchen oder Produkte aus einem Lager herausfiltern, sollen auch hier zunächst Suchbegriffe vom Benutzer angegeben werden. Im hier vorgestellten Konzept geschieht dieser Schritt durch die Definition eines Referenzteils, dessen Fingerprint in der Datenbasis, in der die Signaturen in Textform vorliegen, mit allen vorhandenen Einträgen verglichen wird. Sollten Bauteile gefunden werden, die in die zuvor definierte Suchtoleranz passen, so müssen diese Treffer dem Anwender lediglich angezeigt werden.

Für den Vergleich der Signaturen, die in Form von mehreren Zahlenwerten vorliegen, werden zwei mathematischen Formeln zur Berechnung der euklidischen Distanz und der sogenannten Pearson-Korrelation angewendet, die in dem Buch zum Thema kollektive Intelligenz von Segaran vorgeschlagen werden [Sega-07].

Die euklidische Distanz kann als Abstand zweier Punkte visualisiert werden und wird bei dem Vergleich von mehreren Werten in folgender Formel aufsummiert. Der Wert n stellt im vorliegenden Fall die Anzahl der im Fingerprint enthaltenen Kategorien dar, während mit p und q die Zahlenwerte zweier zu vergleichender Bauteile vorliegen.

$$Euklidische\ Distanz = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Sollten die Fingerprints zweier Bauteile identisch sein, ergibt dies eine euklidische Distanz von Null. Allgemein ist anzumerken, dass je höher die Ähnlichkeit zweier zu vergleichender Signaturen ist, desto geringer entwickelt sich die euklidische Distanz. Bein direkten Vergleich im Zuge einer Suchmaschinenentwicklung muss somit eine Toleranz festgelegt werden, die bestimmt, ab wann ein Bauteil im Vergleich zur Referenz als Treffer zu bewerten ist.

Die zweite Formel zur Ähnlichkeitsbestimmung wird als Pearson-Korrelation bezeichnet und ergibt beim direkten Vergleich einen Wertebereich zwischen -1 und +1. Auch hier stellen p und q die Kategoriewerte dar, während n die Anzahl der Katego-

rien beschreibt. Bei einer hundertprozentigen Übereinstimmung entsteht eine Pearson-Korrelation von +1.

$$Pearson - Korrelation = \frac{\sum_{i=1}^{n} p_i q_i - \frac{\sum_{i=1}^{n} p_i \sum_{i=1}^{n} q_i}{n}}{\sqrt{\left(\sum_{i=1}^{n} p_i^2 - \frac{(\sum_{i=1}^{n} p_i)^2}{n}\right)\left(\sum_{i=1}^{n} q_i^2 - \frac{(\sum_{i=1}^{n} q_i)^2}{n}\right)}}$$

Je nachdem welche der beiden Berechnungsarten beim Vergleich zweier Signaturen angewendet wird, müssen die Suchtoleranzen zuvor unterschiedlich bestimmt werden. Für die endgültige Anwendung sollten so viele Einstellungsmöglichkeiten wie möglich zur Verfügung gestellt werden, so dass der Anwender seine Suche spezifizieren kann und so schnellstmöglich zu einem Ergebnis gelangt.

Implementierung

Die beiden Ansätze zur Berechnung der euklidischen Distanz sowie zur Pearson-Korrelation werden in den Programmen attributes_euklid.py und attributes_pearson.py realisiert. Dabei müssen im zuvor definierten Eingabepfad mindestens die ursprünglichen CATPart-Dateien sowie die Fingerprints als Textdateien vorliegen. Die Ergebnisse des Programmes werden ausschließlich in Form eines Textes in der Python-Konsole ausgegeben.

Nachdem der Benutzer zu Anfang das Referenzteil bestimmt und das Programm die zugehörige Signaturdatei ausgelesen hat, werden nacheinander alle anderen Fingerprints im Eingabeverzeichnis aufgerufen, analysiert und mit der Referenz verglichen. Die beiden Programme unterscheiden sich dabei nur in der Berechnungsmethode der Ähnlichkeiten.

Abbildung 6.1 zeigt zwei Beispielteile, die sowohl in ihrer Geometrie, als auch in ihrer Signatur eine große Ähnlichkeit aufweisen.

In Tabelle 6.1 sind die Werte aller Signaturkategorien und im unteren Teil die euklidische Distanz und die Pearson-Korrelation beider Teile aufgelistet. Dem in Abbildung 6.1 gezeigten linken Bauteil wird hier der Buchstabe (a) zugewiesen, während (b) auf der rechten Seite zu sehen ist.

Da beide Teile sich nur um einen Punkt in der Kategorie 'Eckig komplex' unterscheiden, ergibt sich in der euklidischen Distanz ein Wert von 1 und in der Pearson-Korrelation 0,9954.

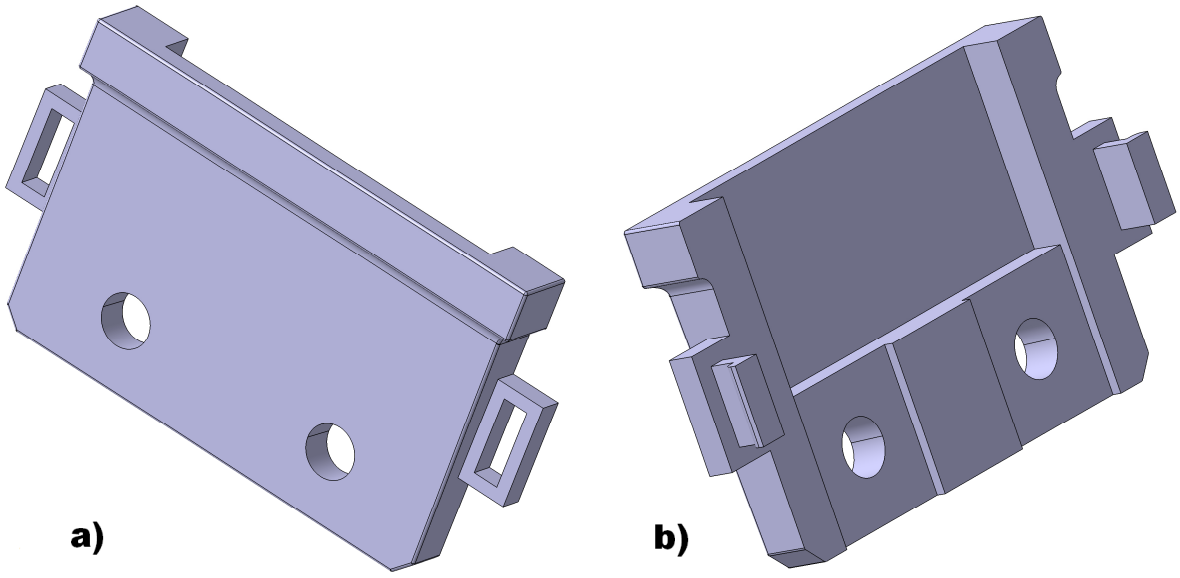

Abbildung 6.1: Zwei Beispielbauteile mit ähnlicher Signatur

Es ist jedoch anzumerken, dass die geometriebegrenzenden Bounding Boxen nicht in die Berechnungen mit einfließen. Die grundlegende Idee bei der Erstellung der Fingerprints ist es die Bauteiltopologie strikt von der Größe zu trennen (vgl. Kapitel 5.1).

Kategorie	a)	b)
1. Größe X in mm	2,975	4,375
2. Größe Y in mm	12,6	12,6
3. Größe Z in mm	22,4	22,4
4. Rund einfach	5	5
5. Rund komplex	2	2
6. Eckig einfach	6	6
7. Eckig komplex	5	4
8. Eben einfach	7	7
9. Eben komplex	8	8
10. Freiform	6	6
11. Details	7	7
12. Primitive	2	2
13. Gewinde	0	0
14. Skizzen	3	3
15. Wiederholungen	0	0
16. 2D	0	0
Euklid	1	
Pearson	0,9954	

Tabelle 6.1: Die Fingerprints und Vergleichswerte beider Beispielteile

Bei der Ähnlichkeitssuche entsteht somit die Tatsache, dass das Hauptaugenmerk beim Vergleich zweier Teile auf die Topologie gelegt wird. Für die spätere ausgereifte Anwendung in der Praxis sollte die Software einen Größenfilter zur Verfügung stellen, der es dem Anwender erlaubt aufgrund der Maße der Bounding Box alle Bauteile, die nicht in die Suchergebnisse passen, herauszufiltern.

Obwohl hier beide Bauteile insbesondere Unterschiede in der Bounding Box und in der Geometrie der Außenseiten aufweisen, erlaubt der direkte Vergleich der Fingerprints in Tabelle 6.1 die Schlussfolgerung, dass eine große Ähnlichkeit besteht und dass bei der Suche die Teile als zugehörig identifiziert werden.

Abbildung 6.2 und die dazugehörige Tabelle 6.2 zeigen zwei weitere Beispielteile, die einerseits eine ähnliche Silhouette, aber bei genauerer Betrachtung eine grundverschiedene Topologie aufweisen. Dies ist sowohl in den Fingerprints, als auch in den beiden Vergleichsmaßen zu erkennen.

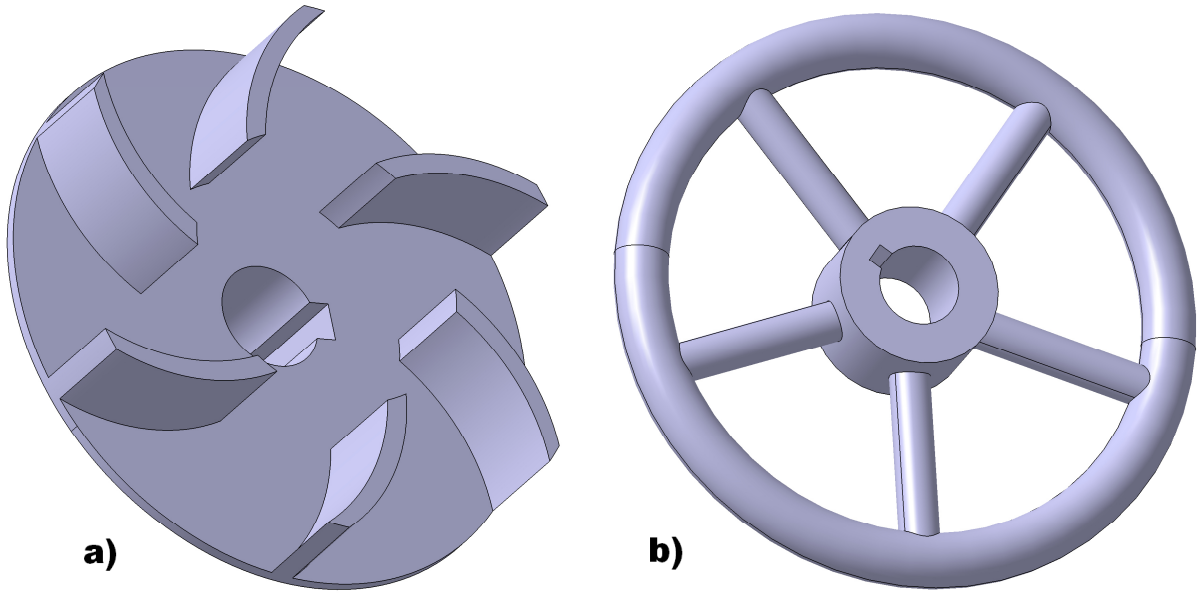

Abbildung 6.2: Zwei Beispielbauteile mit unterschiedlicher Signatur

Obwohl sich beide Teile in einigen Kategorien stark unterscheiden und folglich von der Suchmaschine als nicht ähnlich eingestuft werden, existieren dennoch in einigen Kategorien exakte Übereinstimmungen. Dabei entsteht jedoch die Frage, ob alle Kategorien mit der gleichen Wichtigkeit zu bewerten sind.

Auch die verschiedenen Größen, die aufgrund der unterschiedlichen Maßstäbe in der Abbildung nicht zu erkennen sind, werden nach der Fingerprintanalyse deutlich und der Benutzer kann wählen, ob die Größe bei der Ähnlichkeitssuche miteinbezogen werden soll.

Evaluation

Wie bereits in Kapitel 5.1 erwähnt, ist die Methodik der Fingerprinterstellung auf Basis von berechneten Kategorien generell als innovativ und praktikabel einzustufen. Mit den beiden hier vorgestellten Ansätzen, die es erlauben die Ähnlichkeiten auf einen einzigen Zahlenwert zu reduzieren, wird auch eine schnelle Anwendung in Form einer Suchmaschine ermöglicht. Da nach der Erstellung der Fingerprints, die relativ aufwendig ist (vgl. Kapitel 4.1 bis 4.5), zum Vergleich nur einige wenige Zahlenwerte aus den Textdateien herangezogen werden müssen, kann die Suche auch in großen Datenmengen in angemessener Geschwindigkeit ablaufen.

Kategorie	a)	b)
1. Größe X in mm	33,425	2,1
2. Größe Y in mm	75,425	11,375
3. Größe Z in mm	75,425	11,375
4. Rund einfach	1	4
5. Rund komplex	2	0
6. Eckig einfach	1	1
7. Eckig komplex	6	1
8. Eben einfach	4	4
9. Eben komplex	6	2
10. Freiform	5	5
11. Details	5	5
12. Primitive	1	1
13. Gewinde	0	0
14. Skizzen	1	1
15. Wiederholungen	0	0
16. 2D	0	0
Euklid	14	
Pearson	0.5759	

Tabelle 6.2: Die Fingerprints beider Teile mit geringer Ähnlichkeit

Weiterhin ist zu erwähnen, dass die gesamte Methodik auf einfache Art und Weise erweitert werden kann. Die Berechnungsmethoden der 16 Kategorien können für genauere Ergebnisse weiter verfeinert werden. Auch das Hinzufügen von weiteren Kategorien, die ggf. andere Features berücksichtigen, oder eine benutzerdefinierte Einstellung der Suchtoleranzen ist denkbar.

Wie bereits in Kapitel 5.1 erwähnt, entstehen bei der Anwendung der hier vorgestellten Fingerprint-Methode dennoch Probleme, wenn die Features, die insbesondere aus dem Strukturbaum extrahiert werden, in die Kategorien 'formgebend' und 'de-

tailorientiert' eingeteilt werden können. So definieren einige wenige Features meist am Anfang die grundlegende Form des Bauteils und mit weiteren Befehlen wird danach der Detailgehalt erhöht. Die Feinheiten könnten demnach bei der Ähnlichkeitssuche außer Acht gelassen werden.

6.1.2 VRML-Suchmaschine

Konzept

Der in Kapitel 5.2 beschriebene Fingerprint, der auf Basis der eingefärbten VRML-Bauteile entstanden ist und die verschiedenen Flächenarten in Form von Adjazenzmatrizen darstellt, bildet aufgrund der Genauigkeit, die trotz Informationsreduktion zu einer Signatur bestehen bleibt, den leistungsstärksten Ansatz. In diesem Kapitel sollen die Möglichkeiten erklärt werden, die bei der Entwicklung einer Suchmaschine zum Bauteilähnlichkeitsvergleich entstehen. Das bereits in Kapitel 6.1.1 beschriebene Prinzip bleibt unverändert. Der Anwender muss die Kriterien in Form eines Referenzteils vor dem eigentlichen Suchprozess angeben. Die beinhaltete Ausgangssignatur wird daraufhin mit jedem anderen Fingerprint in der Datenbasis verglichen um bei großen Ähnlichkeiten einen Treffer anzuzeigen.

Hier sollen neben der konventionellen Ähnlichkeitssuche, die die Adjazenzmatrix als Grundlage verwendet, auch weitere Möglichkeiten erläutert werden, die bei speziellen Anwendungsfällen besonders gute Ergebnisse liefern. An dieser Stelle soll jedoch zuerst auf das grundlegende Prinzip bei der Unterscheidung zweier Adjazenzmatrizen nach Quantität und Qualität eingegangen werden.

Beim direkten Vergleich des Referenzteils mit einer beliebigen anderen Signatur aus der Datenbasis werden zu Beginn alle Flächen, aus denen das Referenzteil aufgebaut ist und die in Form des Flächentyps, des Schwerpunktes und der Verbindungen in der Fingerprint-Textdatei definiert sind, einzeln betrachtet. Um im Vergleichsteil ähnliche Flächen zu finden, wird am Anfang die erste Fläche der Referenz untersucht und mit allen Flächen im Vergleichsteil abgeglichen. Sollte eine Fläche, die die gleichen Verbindungen aufweist, gefunden werden, so wird dies für die Referenzfläche registriert. Die Verbindungen einer Fläche zu ihren nächsten Nachbarn werden dabei als eindeutig angesehen. Das bedeutet, dass beispielsweise eine Kreisfläche, die am Ende eines Schaftes zu finden ist, bei VRML-Dateien, die mit CATIA erzeugt wurden, immer mit zwei zylinderförmigen Halbschalen verbunden ist, die der Kategorie der runden Flächen zuzuordnen sind. Aufgrund aller Verbindungen einer Fläche, die insbesondere bei großen Bauteilen sehr komplex werden können, entsteht so eine Eindeutigkeit, die mit dem gesamten Verbund aller Flächen eines Bauteils das gesamte Modell eindeutig beschreibt und sich somit gut als Vergleichswert bei der Ähnlichkeitssuche verwenden lässt.

Des Weiteren sind die Suchergebnisse in Quantität und Qualität aufgeteilt. Wenn nur eine Fläche mit identischen Verbindungen enthalten ist, so wird dies als ein quantitativer Treffer gewertet. Sollten neben den Verbindungen sogar die Distanzen

zwischen den Schwerpunkten der Flächen stimmen, so zählt dies zur Qualität und wird für den Vergleich gewertet. In dieser Art und Weise wird jeweils jede Fläche des Referenzteils mit allen Flächen, aus denen das Vergleichsteil aufgebaut ist, verglichen, was dazu führt, dass die Ergebnisse am Ende in Quantität und Qualität aufgeteilt sind und ggf. interpretiert werden müssen.

Bei dieser Auswertung kann eine Statistik helfen, die nicht alle Ergebnisse, die beim Vergleich entstanden sind, anzeigt, sondern nur die Anzahl der Flächentypen mit ihren zugehörigen Quantitäten sowie Qualitäten.

Diese Vorgehensweise des mathematischen Vergleichs, der in der Implementierung bei allen Programmen gleich abläuft, bietet dem Benutzer neben der Auffindung ähnlicher Bauteile auch die Möglichkeit, ein Referenzteil zu wählen, das im Gegensatz zu den Modellen in der Datenbasis weniger Flächen besitzt und somit auch eine relativ einfache Struktur aufweist. Die Idee ist es somit Subgraphen, die ggf. in einem komplexen Modell enthalten sind und sogar diverse Male auftauchen, zu identifizieren. Als Subgraph wird eine Flächenstruktur angesehen, die Teil eines größeren Konstruktes ist. Wenn vom Anwender überprüft werden soll, ob eine Substruktur in einem komplexen Bauteil mindestens ein Mal vorhanden ist, so muss als Referenz ein Bauteil angegeben werden, das genau diese Substruktur beinhaltet. Anhand der Ergebnisse, die die Suchmaschine liefert, kann bei einem Treffer dann entschieden werden, ob das Vergleichsteil den gesuchten Subgraphen ein Mal oder mehrere Male beinhaltet. Es wird jedoch immer vorausgesetzt, dass die Referenz kleiner ist, als das Vergleichsmodell.

Des Weiteren bietet der automatisierte Matrizenvergleich die Möglichkeit, mit der hier entwickelten Suchmaschine skalierte Bauteile in der Datenbasis aufzufinden und gleichzeitig den Vergrößerungs- bzw. Verkleinerungsfaktor angezeigt zu bekommen. Das zugrunde liegende Prinzip ist simpel. Wenn anhand der Referenz ein Vergleichsteil gefunden wird, das in der Quantität perfekt, in der Qualität aber gar nicht mit dem angegebenen Teil übereinstimmt, so kann davon ausgegangen werden, dass es sich um den gleichen Flächenverbund in einer unterschiedlichen Größe handelt. Wenn als nächstes die Distanzen überprüft werden und ein identischer Skalierungsfaktor für alle Flächen identifiziert wird, so kann dieser berechnet und als Ergebnis angezeigt werden, wodurch es sich beim Vergleichsteil um das gleiche Modell wie bei der Referenz, nur in einer anderen Größe handeln muss.

Auch ein Vergleich zwischen den bereits beschriebenen vereinfachten Matrizen wird durch das gleiche Prinzip ermöglicht. Anstatt die gesamten Adjazenzmatrizen als Eingabeparameter für die Suchmaschine zu verwenden, können auch die ausgedünnten Signaturen analysiert und ggf. als ähnlich gekennzeichnet werden. Dabei ändert sich das Vergleichsprinzip, das in Quantität und Qualität aufgeteilt ist, nicht.

Besonders ein Vergleich von Bauteilen, die das in Kapitel 6.1.1 beschriebene Problem der formgebenden und detailorientierten Features verdeutlichen, soll durch die vorangegangene Ausdünnung ermöglicht werden, indem alle kleinen Flächen und somit unwichtigen Features vor dem Vergleich zu entfernen sind.

Implementierung

Die Implementierung der verschiedenen Ansätze erfolgt in den Programmen vrml_matrix_compare.py zum konventionellen Vergleich der Adjazenzmatrizen in Form einer Suchmaschine, vrml_matrix_statistic.py zur Erstellung der zugehörigen Vergleichsstatistik, vrml_matrix_subgraph.py zur Auffindung von Substrukturen in den Bauteilen im Eingabeverzeichnis, vrml_matrix_scaled.py zur Auffindung skalierter Modelle und zur Berechnung des Vergrößerungs- bzw. Verkleinerungsfaktors sowie simple_vrml_matrix_compare.py für die Verarbeitung der ausgedünnten Matrizen. Alle Programme verlangen als Eingabeinformation lediglich die VRML-Fingerprints bzw. die ausgedünnten Signaturen sowie den Namen des Referenzteils. Die Ausgabe erfolgt immer über die Python-Konsole in Textform. Das Programm vrml_matrix_subgraph.py erstellt zusätzlich einen Ordner mit dem Namen 'Match', in den die Textdateien aller Treffer kopiert werden.

Alle Programme stimmen bei der Umsetzung des eigentlichen Vergleichs im Quelltext überein. Dafür werden zwei geschachtelte Schleifen verwendet, von denen die äußere die Flächen des Referenzteils eintragsweise durchläuft und die innere überprüft, ob diese mit allen Flächen des aktuellen Vergleichsteils kongruieren. Die Untersuchung beinhaltet dabei sowohl die Überprüfung im Hinblick auf die Quantität, bei der ausschließlich die Flächenverbindungen analysiert werden und im Vergleichsteil nach Flächen mit identischen Verbindungen zu suchen ist, als auch die Analyse der qualitativen Verbindungen, die nur stattfindet, wenn zwei Flächen quantitativ übereinstimmen, um dann zu überprüfen, ob außerdem auch die Distanzen zwischen den Flächenschwerpunkten identisch sind.

Abbildung 6.3 zeigt zwei Beispielteile, die die gleichen Größen und somit Bounding Boxen besitzen. Die Abbildung beinhaltet im unteren Teil außerdem die eingefärbten VRML-Modelle, die eine quantitative Ähnlichkeit aufgrund ihrer Färbung erkennen lassen. Die qualitative Übereinstimmung wird in Abbildung 6.4 verdeutlicht, die die Konsolenausgabe des Programmes vrml_matrix_compare.py darstellt. Dabei wurde das Bauteil mit den vier Durchgangsbohrungen und insgesamt 25 Flächen (a) als Referenz gewählt. Die ersten acht Flächen, die im Vergleich angezeigt werden, zeigen sowohl quantitativ als auch qualitativ eine viel schlechtere Übereinstimmung als alle

anderen Flächen, wodurch darauf zu schließen ist, dass es sich dabei um die Flächen handelt, die im Vergleichsteil mit nur zwei Bohrungen (b) nicht enthalten sind. Die zwei Zusatzbohrungen bestehen aus vier Flächen, die jeweils aus zwei zylinderförmigen Halbschalen zusammengesetzt sind. Durch ihre spezielle Lage verändern sie auch die Verknüpfung der benachbarten magentafarbenen Flächen in einer Ebene. All dies spiegelt sich in der Konsolenausgabe in Abbildung 6.4 wider. Da alle anderen Flächen hohe Werte in beiden Kategorien aufweisen, bleibt eine sehr hohe Ähnlichkeit beider Teile bestehen und das Programm zählt das Vergleichsteil als Treffer.

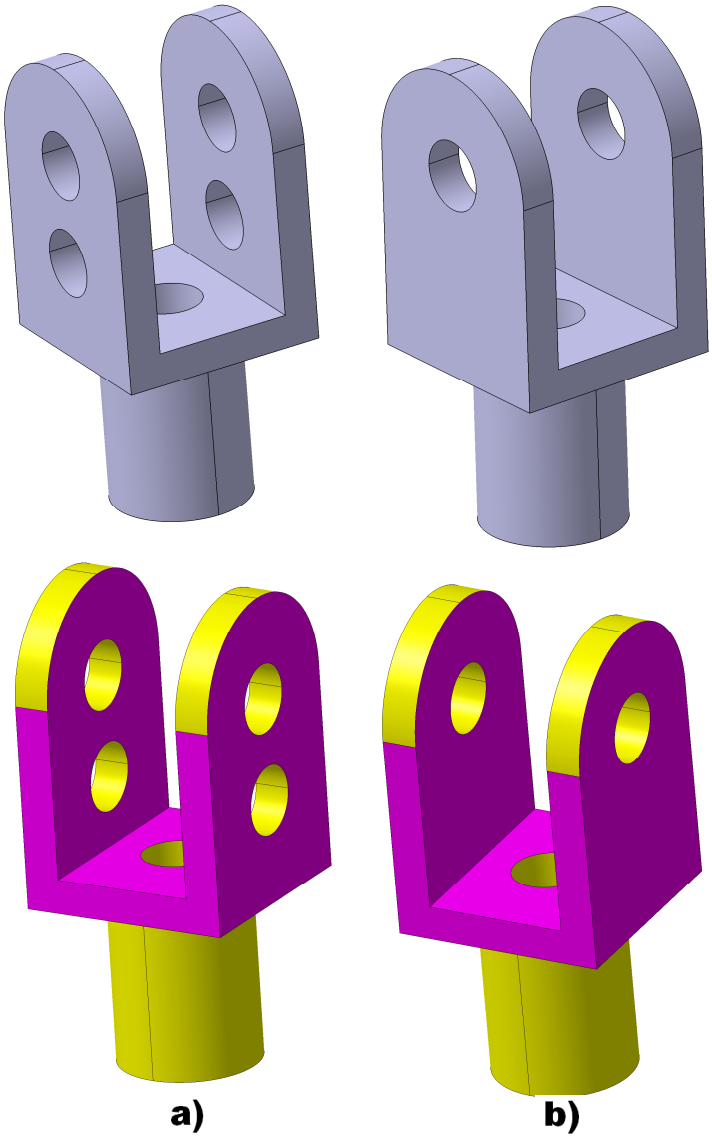

Abbildung 6.3: Zwei ähnliche Bauteile im VRML-Format

Zusätzlich zur Konsolenausgabe des konventionellen Vergleichs kann das Ergebnis des Programmes vrml_matrix_statistic.py in Abbildung 6.5 einer besseren Übersicht dienend hinzugezogen werden. Wie dort zu erkennen ist, werden alle Treffer der einzelnen Flächen in Quantität und Qualität aufsummiert, was zu einer Ausgabe in der

immer gleichen Länge führt und eine Bewertung der Verhältnisse auf den ersten Blick erlaubt. Da die beiden Beispielteile in Abbildung 6.3 ausschließlich aus runden (gelb) und ebenen (magenta) Flächen aufgebaut sind, entstehen Zahlenwerte ungleich Null nur in diesen beiden Kategorien.

```
Reference part:  a)
Current part:  b)
Number of surfaces of the reference part:  25
Number of surfaces of the current part:  21
0 : plane  is found:    0  (quantity)     0  (quality)
1 : plane  is found:    0  (quantity)     0  (quality)
2 : plane  is found:    0  (quantity)     0  (quality)
3 : plane  is found:    0  (quantity)     0  (quality)
4 : round  is found:    4  (quantity)     0  (quality)
5 : round  is found:    4  (quantity)     0  (quality)
6 : round  is found:    4  (quantity)     0  (quality)
7 : round  is found:    4  (quantity)     0  (quality)
8 : round  is found:    4  (quantity)     2  (quality)
9 : round  is found:    4  (quantity)     2  (quality)
10 : round  is found:   4  (quantity)     2  (quality)
11 : round  is found:   4  (quantity)     2  (quality)
12 : plane  is found:   2  (quantity)     2  (quality)
13 : round  is found:   4  (quantity)     4  (quality)
14 : round  is found:   4  (quantity)     4  (quality)
15 : plane  is found:   2  (quantity)     2  (quality)
16 : round  is found:   4  (quantity)     4  (quality)
17 : round  is found:   4  (quantity)     4  (quality)
18 : round  is found:   1  (quantity)     1  (quality)
19 : plane  is found:   2  (quantity)     1  (quality)
20 : round  is found:   4  (quantity)     2  (quality)
21 : round  is found:   4  (quantity)     2  (quality)
22 : plane  is found:   2  (quantity)     1  (quality)
23 : round  is found:   4  (quantity)     2  (quality)
24 : round  is found:   4  (quantity)     2  (quality)
```

Abbildung 6.4: Konsolenausgabe des konventionellen Vergleichs

Auch wenn das Programm bei der Ähnlichkeitssuche während der Anwendung in Suchmaschinenform die Ergebnisse erkennt und alle Treffer für den Anwender herausfiltert, ist dennoch anzumerken, dass zum Teil undurchsichtige Ergebnisse entstehen können, die einer menschlichen Interpretation bedürfen.

Zur Verdeutlichung der Programmergebnisse, die den Skalierungsfaktor anzeigen und die die Subgraphen identifizieren, sollen die Abbildungen 6.6 und 6.7 dienen. In Abbildung 6.6 ist auf der rechten Seite ein relativ komplexes Bauteil in Originalgröße abgebildet (a), dessen eingefärbtes VRML-Modell auf der linken Seite von Abbildung 6.7 zu erkennen ist (a). Eine kleinere Version des Bauteiles, die quantitativ perfekt, qualitativ jedoch gar nicht mit dem Original übereinstimmt, zeigt Abbildung 6.6 auf der rechten Seite (b). Das Programm vrml_matrix_scaled.py, das den Skalierungsfaktor ermittelt, vergleicht nach der Eingabe der Referenz zunächst nur die Größe der Adjazenzmatrix des aktuellen Vergleichsteils mit der Größe der Referenzmatrix. Da die Anzahl der Flächen identisch sein muss, fährt das Programm mit einem eingehen-

deren Vergleich nur fort, wenn beide Matrizen gleich groß sind. Wenn im weiteren Verlauf erkannt wird, dass es sich um identische Teile handelt, die sich lediglich in ihrer Skalierung unterscheiden, muss im Zuge einer Distanzanalyse aller Flächen der genaue Faktor berechnet werden, so dass die gewünschte Information am Ende in der Konsole ausgegeben werden kann.

```
planes in quantity: 8
planes in quality: 6
tris in quantity: 0
tris in quality: 0
fours in quantity: 0
fours in quality: 0
sixes in quantity: 0
sixes in quality: 0
frees in quantity: 0
frees in quality: 0
circles in quantity: 0
circles in quality: 0
rounds in quantity: 85
rounds in quality: 33
```

Abbildung 6.5: Konsolenausgabe in Form einer Statistik

Das Programm berechnet dabei zunächst den Faktor aufgrund der Distanzen zwischen zwei gleichen Flächen und nimmt diesen ersten Wert als Skalierungsfaktor an. In der darauffolgenden Überprüfung, die die Skalierungsfaktoren aller anderen Flächen mit dieser ersten Berechnung vergleicht, stellt sich dann heraus, ob alle Abstände im gesamten Vergleichsteil mit dem angenommenen Faktor vergrößert bzw. verkleinert wurden.

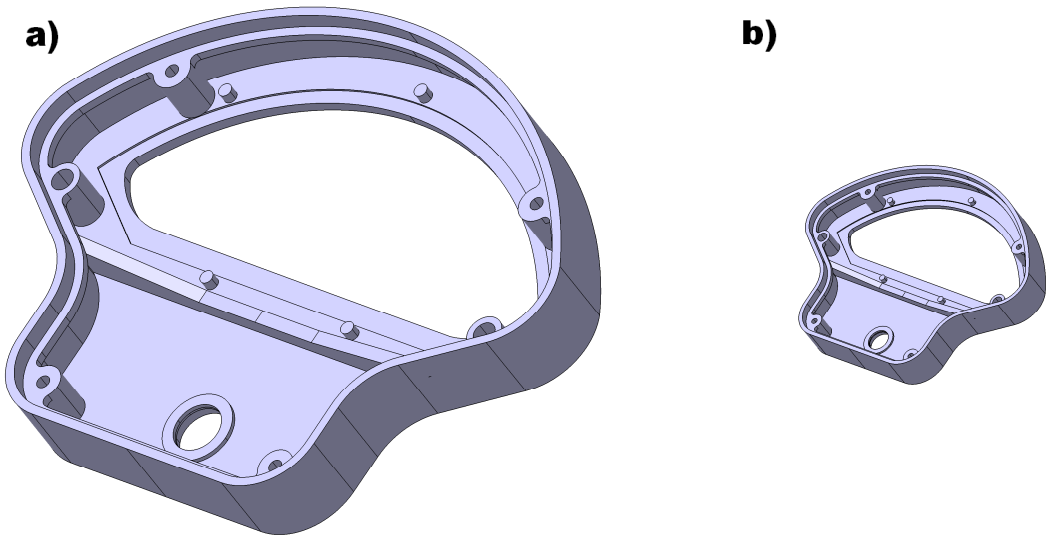

Abbildung 6.6: Komplexes Beispielteil mit skalierter Version

Schon bei der Entdeckung einer Unstimmigkeit kann geschlussfolgert werden, dass es sich beim Vergleichsteil doch nicht um eine skalierte Referenz handelt und das Programm kann in einer Schleife zur Überprüfung des nächsten Teiles übergehen. Die Beispielteile in Abbildung 6.6 besitzen ein Verhältnis von 3:1.

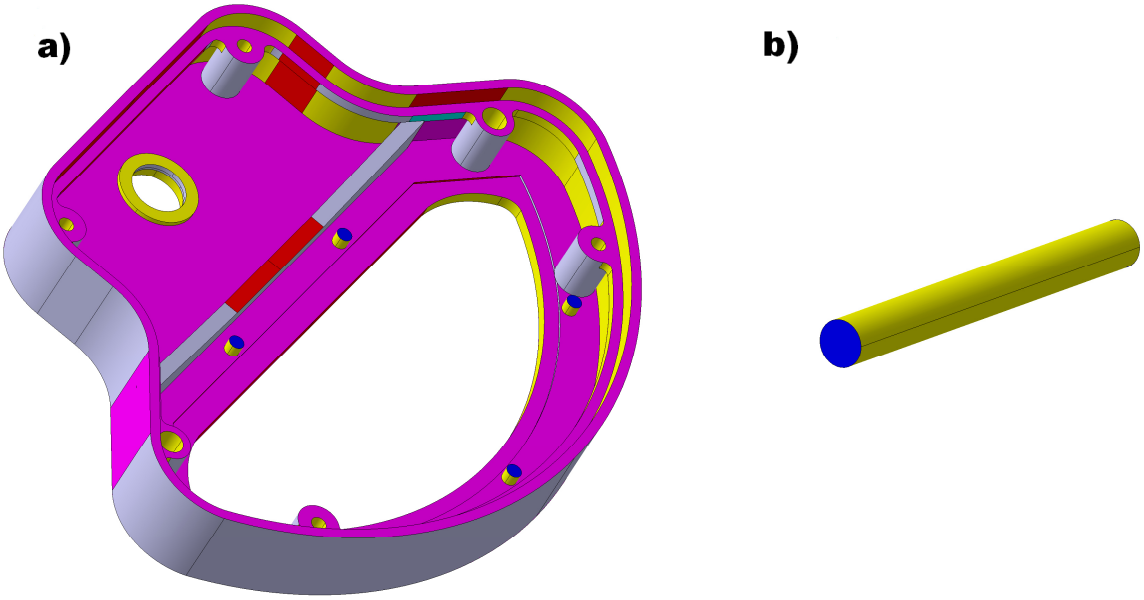

Abbildung 6.7: Eingefärbtes Beispielteil mit Subgraphenstruktur

Das Prinzip der Erkennung von Subgraphen soll anhand von Abbildung 6.7 verdeutlicht werden. Wie bereits im Konzept beschrieben, besteht die Grundidee aus der Konstruktion eines einfachen Bauteiles, das die gesuchte Struktur enthält. Ein solches Teil ist auf der rechten Seite der Abbildung in Form eines Schaftes zu erkennen (b), der lediglich aus zwei Kreisflächen und zwei Zylinderhalbschalen aufgebaut ist. Der Benutzer sollte in der Anwendung des Programmes vrml_matrix_subgraph.py diesen Schaft als Referenz angeben, wenn im komplexen Bauteil auf der linken Seite der Abbildung nach derartigen Substrukturen gesucht werden soll.

Der eigentliche Adjazenzmatrizenvergleich läuft im Programm genauso ab wie bei den oben beschriebenen konventionellen Bauteilen. Der Unterschied besteht darin, dass hier am Anfang überprüft wird, ob das aktuelle Vergleichsteil mindestens 30% größer ist bzw. 30% mehr Flächen aufweist als die Referenz (Prozentwert variabel). Sollte dies der Fall sein, so erfolgt der Vergleich beider Matrizen und die Ergebnisse werden in der Konsole angezeigt.

Die Beispielteile in Abbildung 6.7 bilden einen interessanten Sonderfall. Da das Programm beim Flächenverbund des Referenzteils aus Kreis und Zylinderhalbschalen nicht zwischen innen liegenden und außen liegenden Flächen bzw. Bohrungen und Schäften unterscheiden kann, werden im großen Vergleichsteil alle Flächenverbunde

entdeckt, bei denen ein Kreis mit zwei runden Flächen verbunden ist. Hier sind fünf Schäfte und sechs Bohrungen enthalten, die dazu führen dass die quantitativen Werte in den Suchergebnissen insbesondere bei den Kreisflächen stark vertreten sind. Außerdem muss berücksichtigt werden, dass zwei Kreisflächen mit jeweils den gleichen Verbindungen in der Referenz vorkommen.

Im letzten Ansatz zum Thema VRML-Suchmaschinen soll das Programm simple_vrml_matrix_compare.py, das einen Bauteilvergleich aufgrund der vereinfachten Matrizen durchführt, anhand der Bauteile in Abbildung 6.8 erklärt werden. Der Unterschied zu den anderen Suchmaschinenprogrammen besteht lediglich in den Eingabeparametern. Anstelle einer Untersuchung der ursprünglichen Adjazenzmatrizen werden die ausgedünnten Matrizen zugrunde gelegt.

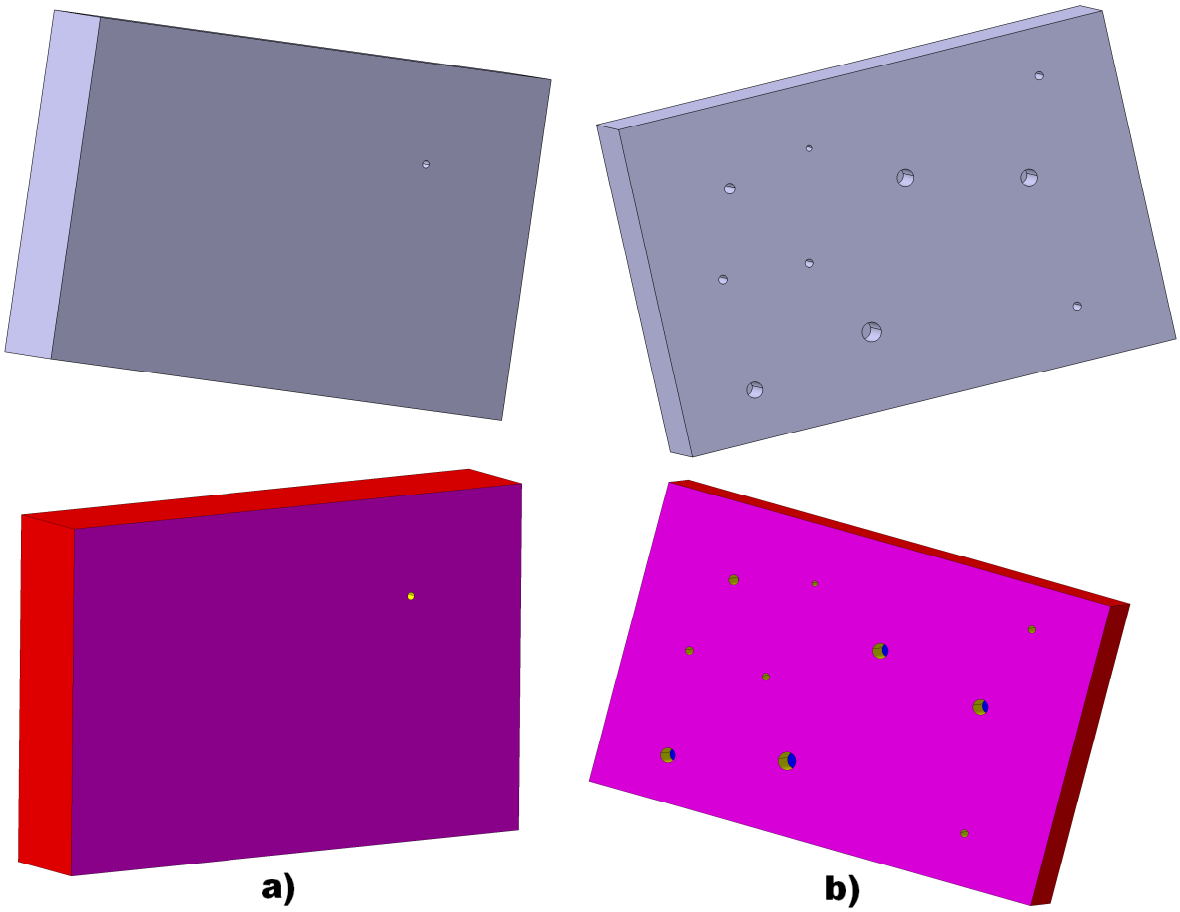

Abbildung 6.8: Zwei Teile zur Demonstration der ausgedünnten Matrizen

Auch in dieser Abbildung sind im oberen Bereich zwei Beispielteile zu sehen, deren ausgedünnte Matrizen eine große Ähnlichkeit aufweisen. Die eingefärbten VRML-Modelle im unteren Bereich lassen aufgrund ihrer ähnlichen Kolorierung bereits zu Anfang erahnen, dass eine große Übereinstimmung vom Programm berechnet wird.

Dennoch ist an dieser Stelle zu erwähnen, dass ein Vergleich zwischen den beiden ursprünglichen Matrizen mit dem Programm vrml_matrix_compare.py zu sehr schlechten Ergebnissen führen würde. Da das rechte Bauteil viele Sacklochbohrungen aufweist (b), was zu einer starken Vergrößerung der Gesamtflächenzahl führt, würde das Programm nur bei den großen formgebenden Flächen eine quantitative sowie qualitative Übereinstimmung feststellen, die jedoch aufgrund ihrer geringen Zahl nicht stark ins Gewicht fallen würde.

Eine vorangegangene Ausdünnung der Matrizen führt aufgrund ihrer geringen Größe zu einer Entfernung aller Bohrungen in beiden Bauteilen. Dies bewirkt, dass dann beide Modelle die in Abbildung 6.9 gezeigte Adjazenzmatrix besitzen und der direkte Vergleich in der Konsolenausgabe in Abbildung 6.10 sehr gute Ergebnisse liefert.

```
00[plane, -26.96, 0.0, 23.92, 32, 105.31, 33, 71.84, 34, 105.31, 35, 71.84]
31[four, -26.96, 38.0, 23.92, 32, 105.31, 35, 71.84, 34, 105.31, 33, 71.84]
32[four, 76.61, 19.0, 23.92, 31, 105.31, 33, 124.61, 0, 105.31, 35, 124.61]
33[four, -26.96, 19.0, 93.21, 31, 71.84, 34, 124.61, 0, 71.84, 32, 124.61]
34[four, -130.55, 19.0, 23.92, 31, 105.31, 35, 124.61, 0, 105.31, 33,
124.61]
35[four, -26.96, 19.0, -45.36, 31, 71.84, 32, 124.61, 0, 71.84, 34, 124.61]
```

Abbildung 6.9: Ausgedünnte Matrix des rechten Beispielteils

Abbildung 6.9 zeigt die ausgedünnte Adjazenzmatrix des rechten Bauteiles, was an der Nummerierung der übrig gebliebenen Flächen zu erkennen ist. Die ausgedünnte Matrix des linken Teils (a) besitzt die gleichen Zahlenwerte im Hinblick auf die Flächenschwerpunkte (violett) und Distanzen (blau), jedoch eine andere Nummerierung und soll an dieser Stelle nicht zusätzlich abgebildet werden.

```
Reference part:  a)
Current part:  b)
Number of surfaces of the reference part:  6
Number of surfaces of the current part:  6
0 : plane  is found:    1  (quantity)     1  (quality)
4 : four  is found:     1  (quantity)     1  (quality)
5 : four  is found:     4  (quantity)     2  (quality)
6 : four  is found:     4  (quantity)     2  (quality)
7 : four  is found:     4  (quantity)     2  (quality)
8 : four  is found:     4  (quantity)     2  (quality)
```

Abbildung 6.10: Ergebnis des Vergleichs beider ausgedünnter Matrizen

In Abbildung 6.10 ist die ursprüngliche Nummerierung des linken Bauteils jedoch zu erkennen, da es hier als Referenz ausgewählt wurde und mit dem Teil auf der rechten Seite verglichen wird. Im Ergebnis sind somit sowohl viele quantitative, als auch qualitative Übereinstimmungen zu erkennen.

Alle vorgestellten Programme, denen das Grundprinzip des Adjazenzmatrizenvergleichs zugrunde liegt, müssen je nach Anwendungsfall vom Benutzer ausgewählt werden und liefern zum Teil Ergebnisse, die einer Interpretation bedürfen.

Evaluation

Die hier vorgestellten Programme zeigen, dass der gesamte Ansatz der Adjazenzmatrizen, die aus den eingefärbten VRML-Modellen hergeleitet werden, eine große Vielfalt aufweist und für verschiedene Anwendungsfälle geeignet ist. Obwohl auch hier der textbasierte Vergleich sehr schnell ablaufen kann und alle Programme ausnahmslos ohne weitere Bedienung auskommen, muss der Anwender beispielsweise bei der Erstellung eines Modells, das zur Subgraphensuche verwendet wird, dieses Bauteil so geschickt gestalten, dass aussagekräftige Ergebnisse bei dem darauffolgenden Vergleich entstehen. Auch die Aufteilung der Konsolenergebnisse in quantitative sowie qualitative Übereinstimmungen bedarf an einigen Stellen einer Interpretation durch den Benutzer. Dabei ist eine Kenntnis der internen Programmabläufe von großem Vorteil.

Da hier nur das Grundkonzept vorgestellt und implementiert, nicht jedoch zur Marktreife gebracht wurde, soll der Schwerpunkt auf die ablaufenden Algorithmen gelegt werden. Um eine komfortable Anwendungspraxis zu erschaffen, könnte der gesamte Ansatz noch mit Hilfe einer GUI, die den Benutzer bei der Interpretation unterstützt, in einem umfassenden Softwarepaket umgesetzt werden.

6.2 Klassifikation

Das hier folgende Kapitel ist in drei Unterkapitel aufgeteilt, die die drei Signaturansätze aus Kapitel 5 wieder aufgreifen und die im Zuge der Informationsweiterverarbeitung in Form einer Klassifikation bzw. eines Clustering der zugrunde liegenden Bauteile die Modelle in der zu untersuchenden Datenbasis in ähnliche Kategorien einteilen. Obwohl die drei Fingerprintformen der Attribute, der VRML- sowie der IGES-Dateien in unterschiedlicher Form vorliegen, laufen die Klassifikationsalgorithmen sehr ähnlich ab und bilden bei gefundenen Übereinstimmungen Gruppierungen, die zu einer besser sortierten Datenbasis führen.

6.2.1 Attribute-Klassifikation

Konzept

Der Vergleich der Attribute-Fingerprints, der in Kapitel 6.1.1 im Zuge der Suchmaschinenentwicklung in Form der euklidischen Distanz sowie der Pearson-Korrelation abgelaufen ist, soll im vorliegenden Kapitel für die Klassifikation bzw. die Gruppierung von Bauteilen in verschiedene Cluster verwendet werden. Im Gegensatz zur Anwendung von Suchmaschinen verlangen die Clusteralgorithmen nicht nach einer anzugebenden Referenz, sondern untersuchen alle Teile in einer Datenbasis automatisch und bilden bei detektierten Ähnlichkeiten zwischen mindestens zwei Modellen entsprechende Bauteilgruppen.

Ähnlich wie bei den in Kapitel 3.11 erwähnten Klassifizieralgorithmen, muss die Schärfe der Klassengrenzen jedoch vor Beginn der Untersuchung definiert werden. Der Algorithmus, der hier vorgestellt wird, wendet eine spezielle Form der euklidischen Distanz auf die 13 Attribute-Fingerprintkategorien an, die nur die Topologie, nicht jedoch die Abmaße der Bauteile berücksichtigen. Die Idee ist es automatisch alle Bauteile nacheinander im Eingabeverzeichnis einzeln aufzurufen, als Referenz zu verwenden und mit allen anderen Bauteilen in der Datenbasis zu vergleichen. Bei der Berechnung der euklidischen Distanz dürfen jedoch die Werte jeder Kategorie um nur zwei Punkte abweichen (Wert einstellbar). Folglich werden zwei Teile als nicht ähnlich erkannt, sobald die Werte sich in mindestens einer Kategorie um mehr als zwei Punkte unterscheiden. Diese Grenze ist allerdings vom Benutzer zu variieren, so dass auch Abweichungen, die größer oder kleiner sind, berücksichtigt werden können.

Dennoch ist im Großen und Ganzen davon auszugehen, dass zwei Bauteile für die Einsortierung in die gleiche Klasse schon nicht mehr ähnlich genug sind, wenn die Abweichung in einer einzigen Kategorie mehr als zwei Punkte beträgt. Andererseits erlaubt diese Vorgehensweise theoretisch eine gesamte euklidische Distanz von maximal 26 Punkten. Wenn sich zwei Bauteile in jeder der 13 Kategorien um zwei Punkte unterscheiden, gelten sie immer noch als ähnlich und werden am Ende dem gleichen Cluster zugewiesen.

Implementierung

Im Programm attributes_cluster.py wird die im Konzept beschriebene Methodik umgesetzt. Nachdem der Benutzer lediglich die Eingabe- und Ausgabeverzeichnisse bestimmt hat, untersucht das Programm die Textdateien, die die Fingerprints enthal-

ten automatisch und erstellt für jede gefundene Klasse einen Ordner, in den sowohl die Textdateien, als auch die ursprünglichen CATPart-Dateien hereinkopiert werden. Alle Bauteildateien, die keinem Cluster angehören, verbleiben im Eingabeverzeichnis.

Abbildung 6.11 zeigt den Quelltext des Programmes und verdeutlicht die Umsetzung in mehreren Schleifen. Alle Bauteile, die zu Klassen zusammengefasst werden, werden direkt nach ihrer Entdeckung zuerst in eine geschachtelte Liste geschrieben, die am Ende noch einmal untersucht und ausgewertet wird. Für die einzelnen Cluster werden daraufhin Ordner erstellt, die mithilfe eines internen Zählers durchnummeriert werden und in die alle nötigen Dateien kopiert werden. Die verschiedenen Klassen besitzen folglich keine speziellen Benennungen und müssen durch eine eingehendere Untersuchung der beinhalteten Teile unterschieden werden.

In Abbildung 6.12 sind sechs Bauteile eines gleichen Clusters abgebildet (aus der Grundgesamtheit von ca. 100 Teilen). Da es sich dabei um relativ einfache Drehteile handelt, weisen auch die zugehörigen Fingerprints, die in Tabelle 6.3 in durchnummerierter Form zu sehen sind, eine große Ähnlichkeit auf, was zu sehr geringen euklidischen Distanzen führt. Da während dieses Tests die Größenverhältnisse bei dem Vergleich der Fingerprints nicht berücksichtigt wurden, sind große Unterschiede in den Abmaßen im oberen Bereich der Tabelle zu erkennen.

Die in den Kapiteln 6.2.2 und 6.2.3 vorgestellten Klassifikationsalgorithmen besitzen programmtechnisch den gleichen Ablauf und sollen nicht noch einmal im Detail erklärt werden. Nur die Fingerprints und ihre zugrunde liegende bewertende Analyseart wird variiert.

Evaluation

Das grundlegende Prinzip bei der Erstellung von Klassen und der Sortierung von Bauteilen in entsprechende Cluster unterscheidet sich in dem hier vorgestellten Konzept nicht sonderlich von den wissenschaftlichen Ansätzen aus Kapitel 3.11.

Als Nachteil wäre die Tatsache zu erwähnen, dass die Klassen keine automatischen Benennungen, wie z.B. 'Einfache Drehteile' oder 'Blechteile', aufweisen. Für den Anwender, der ggf. nicht weiß welche Art von Modelle in der zu sortierenden Datenbasis enthalten sind, entsteht so die Option einer weiteren Untersuchung der fertig klassifizierten Datenbank.

Ein Algorithmus der eine Einteilung nach der Bauteilfunktion leisten könnte, wäre im Hinblick auf die automatisierte Klassifikation ohne manuelle Informationseingabe von großer Hilfe.

```
copylist=[]
for ij in range(len(files)):
    if "_fingerprint.txt" in files[ij]:
        tempname=files[ij].replace("_fingerprint.txt","")
        booltest=0
        for i in range(len(copylist)):
            for j in range(len(copylist[i])):
                if tempname==copylist[i][j]:
                    booltest=1
        if booltest==0:
            dat=open(str(inputpath)+"\\"+tempname+"_fingerprint.txt",
            "r").readlines()
            ref=[]
            for i in range(16):
                temp=dat[i]
                temp=re.split(r'\n',temp)
                temp=temp[0]
                ref.append(temp)
            checklist=[]
            for kl in range(len(files)):
                if "_fingerprint.txt" in files[kl]:
                    tempname2=files[kl].replace("_fingerprint.txt","")
                    if tempname!=tempname2:
                        dat2=open(str(inputpath)+"\\"+
                        tempname2+"_fingerprint.txt", "r").readlines()
                        if len(dat)==len(dat2):
                            cur=[]
                            for i in range(16):
                                temp=dat2[i]
                                temp=re.split(r"\n",temp)
                                temp=temp[0]
                                cur.append(temp)
                            counter=0
                            for i in range(13):
                                if int(ref[i])-int(cur[i])==0:
                                    counter=counter+1
                                if int(ref[i])-int(cur[i])>=2:
                                    counter=counter-100
                                if int(ref[i])-int(cur[i])<=-2:
                                    counter=counter-100
                            if counter>=10:
                                checklist.append(tempname2)
            if len(checklist)>0:
                checklist.append(tempname)
            copylist.append(checklist)
counter=0
for i in range(len(copylist)):
    if len(copylist[i])>0:
        os.mkdir(outputpath+"\\"+str(counter))
        counter=counter+1
        for j in range(len(copylist[i])):
            shutil.copy(inputpath+"/"+
            str(copylist[i][j])+".CATPart",
            inputpath+"/"+str(counter-1)
            +"/"+str(copylist[i][j])+".CATPart")
            shutil.copy(inputpath+"/"+str(copylist[i][j])+
            "_fingerprint.txt",inputpath+'/'+str(counter-1)
            +"/"+str(copylist[i][j])+"_fingerprint.txt")
```

Abbildung 6.11: Quelltext der Attribute-Klassifikation

Die Ergebnisse, die durch die hier vorgestellte Methode geliefert werden, können insbesondere bei einfachen Bauteilen mit wenigen Features und simpler Geometrie als funktionell bewertet werden.

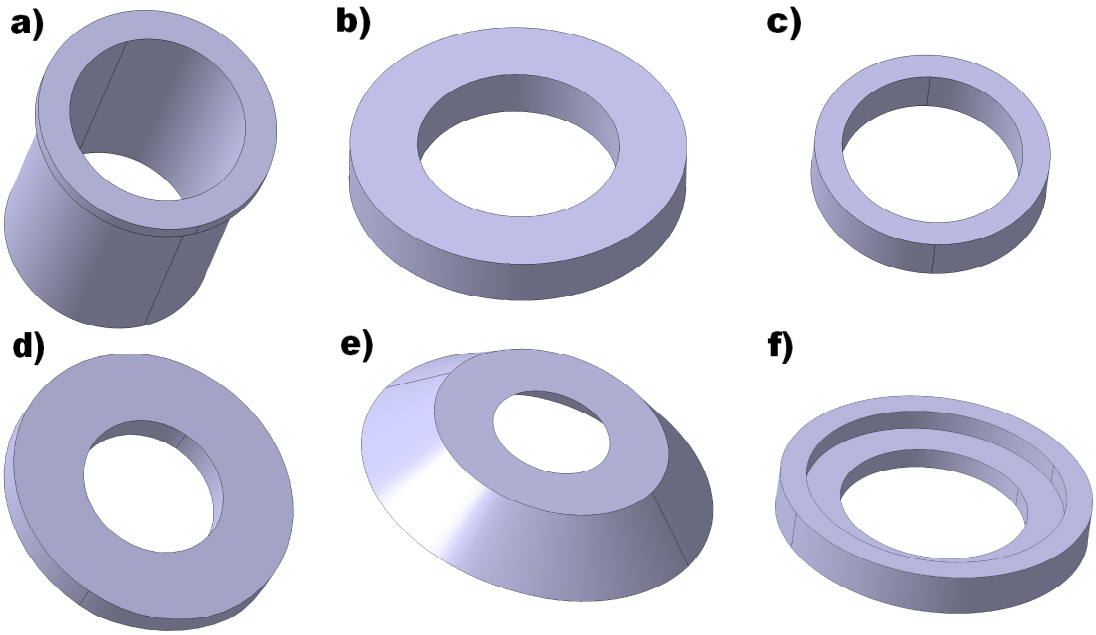

Abbildung 6.12: Einfache Rotationsteile im gleichen Cluster

Aber auch komplexere Teile können nahezu identische Fingerprints aufweisen und somit der gleichen Klasse angehören.

Kategorie	a)	b)	c)	d)	e)	f)
1. Größe X in mm	76,3	3,325	3,325	1,4	5,425	4,375
2. Größe Y in mm	76,3	20,3	15,4	10,325	30,275	30,275
3. Größe Z in mm	77,35	20,3	15,4	10,325	30,275	30,275
4. Rund einfach	1	1	0	0	1	1
5. Rund komplex	1	1	1	1	1	1
6. Eckig einfach	0	0	0	0	0	0
7. Eckig komplex	0	0	0	0	0	0
8. Eben einfach	2	2	2	2	2	2
9. Eben komplex	0	0	0	0	0	0
10. Freiform	1	1	1	1	1	1
11. Details	1	1	1	1	1	1
12. Primitive	0	0	0	0	0	0
13. Gewinde	0	0	0	0	0	0
14. Skizzen	0	0	0	0	0	0
15. Wiederholungen	0	0	0	0	0	0
16. 2D	0	0	0	0	0	0

Tabelle 6.3: Die Fingerprints der sechs Rotationsteile

Da die Klassengrenzen durch den Benutzer variiert und auf die speziellen Anwendungsgegebenheiten eingestellt werden können, lassen sich bei Datenbanken, die schon am Anfang ausschließlich ähnliche Teile beinhalten, die Klassifikationsparameter ändern und die Ergebnisse so beeinflussen. Auch die Suchtoleranzen für die euklidischen Distanzen bzw. die Pearson Korrelation sind vom Benutzer einstellbar und können somit je nach Anwendungsfall variiert werden. Je nachdem welche Art von Datenbasis untersucht werden soll, sind für eine sinnvolle Feinjustierung eventuell Erfahrungswerte nötig. Sollte eine Datenbank klassifiziert werden, die mit einer Vielzahl an sehr unterschiedlichen Modellen gefüllt ist, so können die Klassengrenzen bzw. Toleranzen großzügig definiert werden (z.B. eine euklidische Distanz von maximal 7 Punkten und eine Pearson-Korrelation größer 0,8).

Wenn aber zwischen ähnlichen Teilen engere Klassengrenzen gezogen werden sollen, so sind auch die Grenzwerte der euklidischen Distanz (z.B. maximal 3 Punkte insgesamt) bzw. die Pearson-Korrelation (z.B. zwischen 1 und 0,9) entsprechend streng auszurichten.

Ebenso wie bei den Bauteilvergleichen bei der Anwendung der entwickelten Suchmaschine, lässt sich die Anwendbarkeit auf große Datenmengen als positiv bewerten. Auch in diesem Ansatz läuft die gesamte Klassifikation nur noch textbasiert und somit für den Benutzer in geeigneter Geschwindigkeit ab (keine langen Wartezeiten).

6.2.2 VRML-Klassifikation

Konzept

Die in Kapitel 6.1.2 erwähnte Unterteilung der Vergleichsergebnisse in Quantität und Qualität wird auch in den Clusteralgorithmen, deren Prinzipien hier vorgestellt werden, fortgeführt. Da sich die Vorgehensweisen bzw. internen Abläufe der Programme stark mit denen in Kapitel 6.2.1 ähneln, sollen sie an dieser Stelle nicht noch einmal ausführlich beschrieben werden. Dennoch ist zu erwähnen, dass für verschiedene Anwendungsfälle unterschiedliche Konzepte bei der Klassifikation der in der Datenbasis befindlichen Bauteile benötigt werden. Aus diesem Grund werden hier fünf Ansätze erläutert, die sowohl die konventionellen VRML-Fingerprints, als auch die Vergleichsergebnisse der Subgraphen bei der Klassifikation in Quantität und Qualität aufteilen. Des Weiteren werden auch die simplifizierten VRML-Signaturen zur Einteilung der Bauteile in Gruppen verwendet, was aufgrund der vorangegangenen Ausdünnung dazu führt, dass kleine zuvor gelöschte Details nicht mit einbezogen werden und sich das Clustering auf die formgebenden Features der Modelle und somit auf ihr Gesamterscheinungsbild konzentriert.

Der Unterschied zur Klassifikationsmethode der Attribute-Fingerprints ist die Tatsache, dass hier ausschließlich die vollständigen bzw. ausgedünnten Adjazenzmatrizen verwendet werden. Ebenso wie in der in Kapitel 6.1.2 beschriebenen Suchmaschine, die in mehreren Programmen für verschiedene Anwendungen ausgelegt wurde, werden in allen fünf Versionen, die hier behandelt werden, die Teile miteinander verglichen und bei großen Ähnlichkeiten, die bei der Suchmaschine mit einem Treffer gleichzusetzen sind, die Bauteile zur gleichen Klasse gezählt. Dadurch dass nacheinander alle Teile als Referenz gewählt werden, kann es vorkommen, dass ein Modell in mehrere Klassen eingeordnet wird.

Die Unterteilung in Quantität und Qualität ist mit der Berücksichtigung der Bounding Boxen im Attribute-Fingerprint zu vergleichen. Wenn die Klassifikation lediglich im Hinblick auf die Quantität vollzogen wird, so ist davon auszugehen, dass es sich bei den Ergebnissen um eine topologische Sortierung der Bauteile handelt. Wird die Qualität mit berücksichtigt, so erfolgt eine Einteilung aufgrund der Topologie und der Maße. Wie bereits erwähnt, sind beide Anwendungsfälle in der Realität vorstellbar und sollten dem Benutzer zur Verfügung gestellt werden. Die Klassifizierung anhand der ausgedünnten Matrizen führt zu gröberen Klassengrenzen, so dass bei der Sortierung weniger Gruppen mit jeweils mehr enthaltenen Teilen entstehen.

Implementierung

In diesem Kapitel sollen die Ergebnisse der Implementierung anhand einiger Beispiele angedeutet werden. Die im Konzept beschriebenen Ansätze werden in den Programmen vrml_cluster_quan.py, vrml_cluster_qual.py, subgraph_vrml_cluster_quan.py, subgraph_vrml_cluster_qual.py und simple_vrml_cluster.py umgesetzt. Alle Programme verarbeiten die normalen bzw. ausgedünnten VRML-Fingerprints im Eingabeverzeichnis und liefern im zuvor gewählten Ausgabepfad ein Ordnersystem, mit nummerierten Klassen.

Der Quelltext aller Programme ist ebenso wie die Clusteralgorithmen aus Kapitel 6.2.1 immer in der gleichen Form aufgebaut. Sobald bei dem Vergleich eine Ähnlichkeit zwischen der Referenz und dem aktuellen Teil auffällt, werden beide Teile an eine Liste angehängt, die am Ende aller Durchgänge die Informationen, die für die Klassenerstellung nötig sind, beinhaltet.

Das Ordnersystem aller Klassen entsteht im Ausgabepfad, nachdem der Algorithmus die Liste ausgelesen hat und die zugrunde liegenden Informationen für die Kopiervorgänge der ursprünglichen CATPart-Dateien und ihre entsprechenden Fingerprints verwendet hat.

Abbildung 6.13 zeigt zwei Beispielteile, die bei der quantitativen Klassifikation dem gleichen Cluster zugewiesen wurden.

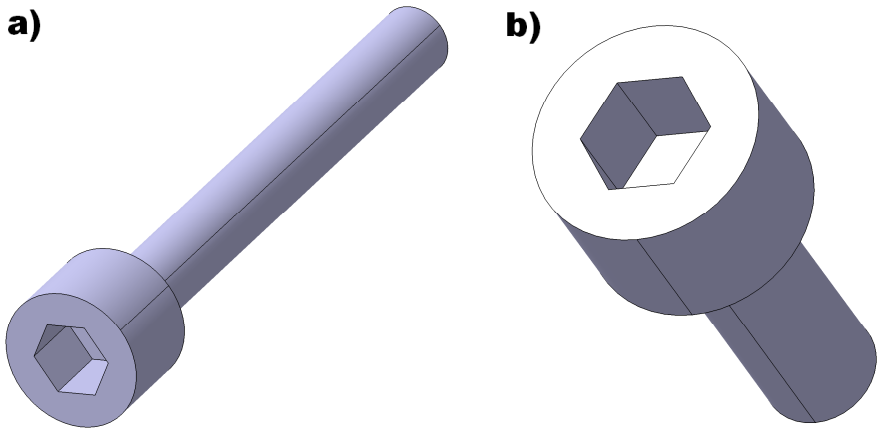

Abbildung 6.13: Zwei Zylinderkopfschrauben im gleichen Cluster

Die beiden Schrauben in Abbildung 6.14 wurden aufgrund ihrer anderen Kopfform im gleichen Durchgang mit denselben Parametern in eine andere Klasse sortiert.

Für beide Schraubenarten gilt die Tatsache, dass die Schraubenköpfe im Verhältnis zum gesamten Bauteil relativ viele Flächen besitzen und somit im Fingerprint stark vertreten sind. Obwohl der Schaft bei allen vier Teilen aus den gleichen fünf Flächen (eine Kreisfläche, zwei Zylinderhalbschalen und zwei Fasen) aufgebaut ist,

erkennt das Programm den Unterschied der Schraubenköpfe, die aufgrund ihrer Flächenzahl stark ins Gewicht fallen und erstellt für beide Schraubenarten unterschiedliche Cluster. In der Grundgesamtheit waren von beiden Schraubentypen (Zylinderkopfschraube mit Innensechskant und Sechskantschraube) jeweils nur diese beiden Exemplare vorhanden.

Eine Anwendung der qualitativen Klassifikation würde zu den gleichen Ergebnissen führen, wenn vorausgesetzt wird, dass beispielsweise die Schraubenköpfe identische Größen aufweisen. Eine Bauteilsortierung, die neben der Topologie auch die Verhältnisse der Maximalmaße berücksichtigt, würde insbesondere am Beispiel von Schrauben zu guten Ergebnissen führen, da sich die Kopfformen sowie -größen, nicht aber die Schaftlängen bei Schrauben mit den gleichen Schlüsselweiten gleichen. Wenn sich in der Datenbasis verschiedene Schraubentypen mit den jeweils gleichen Kopfgrößen befinden, so entsteht bei der qualitativen Klassifikation die gleiche Einteilung wie sie in den Abbildungen 6.13 und 6.14 zu sehen ist. Neben einer Klassifikation in die Schraubentypen findet allerdings zusätzlich noch eine Unterscheidung im Hinblick auf die Kopfgröße statt.

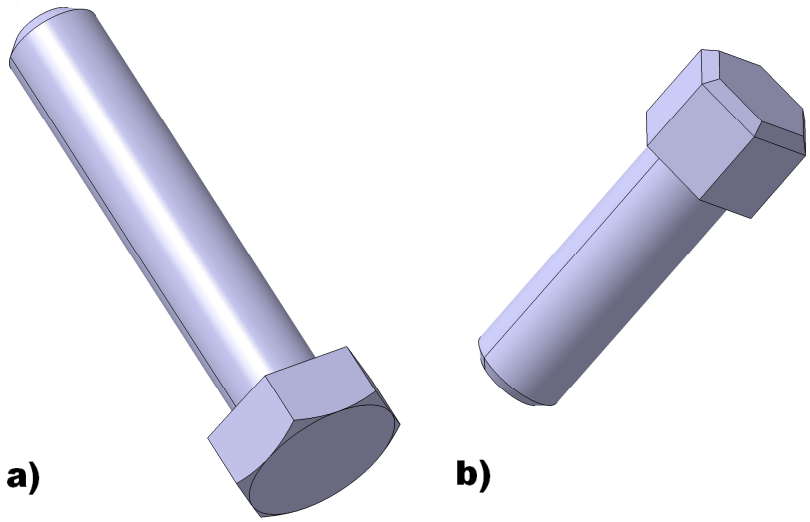

Abbildung 6.14: Zwei Sechskantschrauben im gleichen Cluster

Ein ähnliches Prinzip entsteht bei den beiden Programmen, die für die Klassifikation der Subgraphen zuständig sind. In Abbildung 6.15 sind zwei Kurbelwellen abgebildet, die zu einem Vierzylinder- (a) und einem Sechszylindermotor (b) gehören.

Die Vierzylinderkurbelwelle erfüllt dabei alle Vorgaben, die für die Erkennung als Subgraph der Sechszylinderkurbelwelle nötig sind, was auch in der Suchmaschine zur Detektion von Subgraphen zu einem Treffer führt.

In den beiden Clusterprogrammen zur Sortierung der Bauteile mit Subgraphen werden beide Wellen vom quantitativen Algorithmus der gleichen Klasse zugewiesen.

Der qualitative Algorithmus verhält sich ähnlich, wenn sich auch die Distanzen zwischen den Flächen gleichen bzw. beide Bauteile die gleichen Größenverhältnisse aufweisen. Lediglich die Vorgabe, dass das Vergleichsteil 30% (Wert einstellbar, empirisch hergeleitet) mehr Flächen aufweisen muss, als die Referenz, muss erfüllt sein.

Zur Verdeutlichung der algorithmischen Abläufe des Programmes simple_vrml_cluster.py sollen die Ergebnisse in Abbildung 6.16 beschrieben werden. Alle sechs dort zu sehenden Teile wurden ebenfalls dem gleichen Cluster zugewiesen.

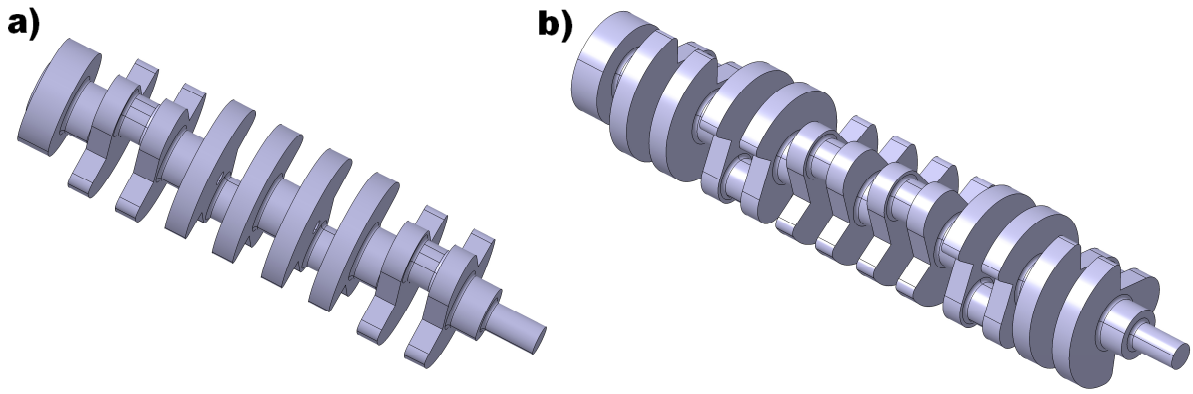

Abbildung 6.15: Zwei Kurbelwellen zur Anwendung im Subgraphencluster

Dadurch dass zuvor die Ausdünnung der Fingerprints stattgefunden hat, bleiben bei allen sechs Bauteilen die großen formgebenden Flächen übrig, während die Bohrungen und Taschen gelöscht werden. Dies hat zur Folge, dass das Programm die würfelförmige Gesamterscheinung erkennt und alle Modelle dem gleichen Cluster zuweist. Auch wenn kleine Teile der verbleibenden Strukturen sich unterscheiden, so gleichen sich doch eine Vielzahl der bestehen bleibenden Flächen und die Teile werden als ähnlich identifiziert. Hier spielen insbesondere die großen Viereckflächen an den Seiten eine große Rolle. Diese bleiben unverändert und machen einen Großteil der ausgedünnten Adjazenzmatrix aus.

Evaluation

Die Ergebnisse in den Abbildungen verdeutlichen, dass es sich bei dem gesamten VRML-Ansatz im Vergleich zum Attribute- und IGES-Fingerprint um die leistungsstärkste Methode handelt. Die Verkürzung der Bauteileigenschaften in eine Signatur, die eine graphenförmige Adjazenzmatrix aufweist, besitzt den Vorteil, dass die Bauteilstrukturen erhalten bleiben und beim Vergleich bzw. bei der Klassifikation in die Auswertung mit einfließen können. Die Einteilung in Quantität und Qualität ist bei

der Anwendung ebenfalls von Vorteil, da die Bauteilmaße nicht immer von Interesse sind.

Das Programm zur Klassifikation der ausgedünnten Bauteilsignaturen bietet den Vorteil, dass eine noch gröbere Einteilung in Cluster stattfinden kann, bei der die Bauteile auf ihr Gesamterscheinungsbild reduziert werden. Somit sind mit den hier vorgestellten Algorithmen viele Bereiche der Klassifikation bzw. des Vergleichs abgedeckt.

Wie schon in den Kapiteln der Suchmaschinen beschrieben, eignen sich die Methoden auch zur Anwendung auf große Datenmengen, da die Klassifikation textbasiert ablaufen kann und nur die Informationen aus den Dateien extrahiert werden müssen. Im Gegensatz zum Attribute-Fingerprint bietet der VRML-Ansatz eine andersartige Grundidee, die auch gute Ergebnisse liefert, wenn die Ausgangsmodelle eine hohe Komplexität aufweisen.

Des Weiteren besteht nicht das Problem, dass die Methodik featurebasiert abläuft und somit formgebende Features von den detailorientierten unterschieden werden müssen. Im VRML-Ansatz wird dieses Problem durch die Ausdünnung gelöst, die dafür sorgt, dass bei Bedarf alle kleinen Flächen gelöscht und somit nicht mehr berücksichtigt werden.

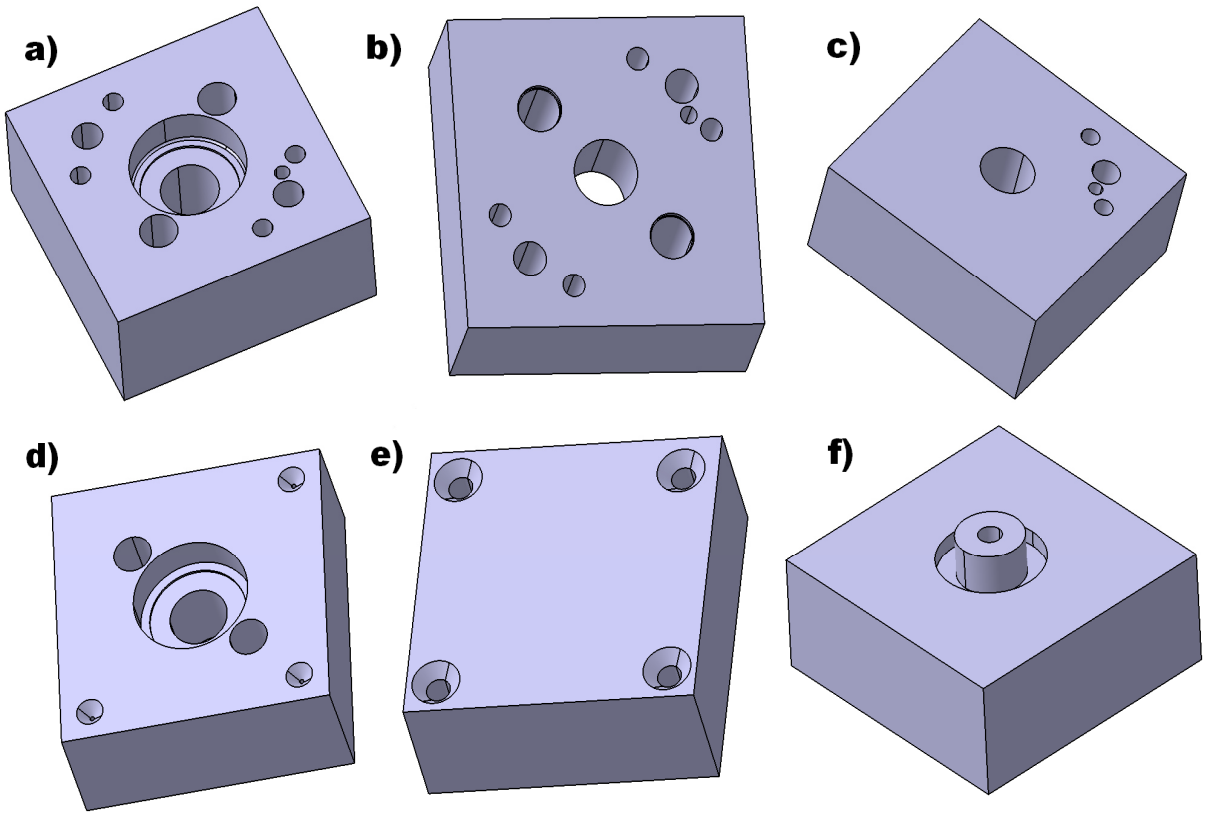

Abbildung 6.16: Sechs eckige Beispielteile im gleichen Cluster

Als Nachteil bleibt zu erwähnen, dass die Flächenverbindungen nicht immer aussagekräftig sind. Insbesondere Fasen oder Verrundungen können die Struktur der Adjazenzmatrizen verändern. Wenn zwei Bauteile mit ähnlicher Form verglichen werden, von denen eines mit Fasen oder Verrundungen versehen ist, so fällt die automatisierte Entdeckung einer ähnlichen Grundform schwer. Auch wenn zuvor das Werkzeug der Ausdünnung angewendet wird, so bedeutet dies nicht zwangsläufig eine Verbesserung des Vergleichspotenzials. Wenn etwa mehrere Kanten von Bauteil (a) aus Abbildung 6.16 abgerundet oder gefast wären, so könnte das Bauteil nicht dem gleichen Cluster wie die Teile (b) bis (f) zugeordnet werden. Da bei der Ausdünnung alle kleineren Flächen entfernt werden, könnte es vorkommen, dass einige größere verbliebene Flächen zwar noch in der Matrix vorhanden sind, aber sowohl ihre Form bzw. Kategorie ändern, als auch keinerlei Nachbarflächen mehr aufweisen. Kleine Fasen oder Verrundungen können folglich nicht das Gesamterscheinungsbild des Bauteils ändern, wohl aber das der ausgedünnten Adjazenzmatrix.

6.2.3 IGES-Klassifikation

Konzept

Obwohl die Signaturen der VRML-Modelle ebenso wie die Signaturen der IGES-Modelle eine Form von Adjazenzmatrizen enthalten, so läuft der Vergleich zwischen zwei Bauteilen doch unterschiedlich ab. Wie in den Kapiteln 6.1.2 und 6.2.2 erwähnt, werden bei der Analyse zweier VRML-Signaturen alle Flächen zusammen mit ihren direkten Nachbarn und den zugehörigen Distanzen einzeln betrachtet. Da der hier beschriebene IGES-Ansatz keine Unterteilung in die einzelnen Flächentypen beinhaltet, wird der Schwerpunkt insbesondere auf die Verbindungen gelegt.

Um zwei Bauteile anhand ihrer Adjazenzmatrizen aussagekräftig miteinander vergleichen zu können, findet zu Beginn eine in Kapitel 5.3 beschriebene Sortierung der Flächen nach ihrer Größe statt. Dabei wird davon ausgegangen, dass die großen wichtigen Flächen sowohl bei der vollständigen, als auch bei der ausgedünnten Matrix formgebend sind und beim Vergleich zwischen zwei ähnlichen Bauteilen auf der gleichen Position der Flächenrangfolge stehen. Nach einer solchen Sortierung muss der Algorithmus nur noch überprüfen, ob alle Matrixeinträge und somit die Flächenverbindungen miteinander übereinstimmen. Die Distanzen werden dabei in der Regel nicht berücksichtigt, da zuerst die topologische Struktur zweier Teile aussagekräftiger und bei dem Vergleich der IGES-Dateien zu bevorzugen ist. Sollten auch die Größenverhältnisse eine Rolle spielen, so können die Abstände beim Vergleich mit hinzugezogen werden.

Die Klassifizieralgorithmen, die hier für die Gruppierung der vollständigen und der ausgedünnten Modelle eingeteilt sind, laufen im Prinzip ähnlich wie die anderen in Kapitel 6.2.1 und 6.2.2 beschriebenen Programme ab. Die Dateien in einer Datenbasis werden bei mindestens zwei ähnlichen Teilen in zusammengehörende Klassen sortiert, so dass bei der Anwendung des Algorithmus eine überarbeitete übersichtlichere Datenbank entsteht.

Implementierung

Die Ansätze werden in den beiden Programmen iges_cluster.py sowie simple_iges_cluster.py umgesetzt. Vergleichbar mit den anderen Klassifikationsprogrammen werden auch hier vom Benutzer lediglich die Eingabepfade, die die unsortierten Textdateien mit den IGES-Signaturen bzw. den vereinfachten Matrizen enthalten,

und die Ausgabeverzeichnisse definiert, in die die beiden Programme jeweils die Ordnersysteme abspeichern.

Der eigentliche Matrizenvergleich wird von beiden Programmen in mehreren Schleifen selbstständig durchgeführt. Die Matrizen der Referenz und des aktuellen Vergleichsteils werden dafür sowohl zeilen-, als auch spaltenweise durchlaufen und die Parameter alle einzeln miteinander verglichen. Auch hier werden zunächst alle zu einer Klasse gehörigen Bauteile bzw. deren Namen an eine Liste angehängt, die nach der Untersuchung aller Modelle im Eingabeverzeichnis analysiert wird, so dass ihre Struktur in Form von nummerierten Ordnersystemen umgesetzt werden kann.

In Abbildung 6.17 sind zwei verschiedene Bolzen zu erkennen, die exakt die gleiche Flächenstruktur aufweisen und topologisch identisch sind. Das Programm iges_cluster.py weist somit beide Bauteile dem gleichen Cluster zu, wenn vorausgesetzt wird, dass die Distanzen zwischen den Flächenschwerpunkten bzw. die Qualität nicht mitberücksichtigt werden.

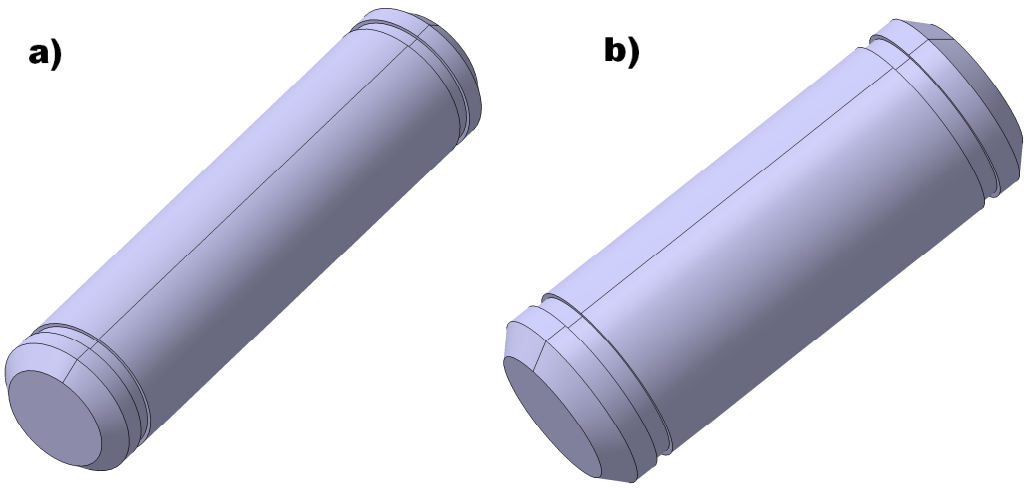

Abbildung 6.17: Zwei verschieden große Bolzen im gleichen Cluster

Ein weiteres Ergebnis der Klassifikation auf Basis der ausgedünnten Matrizen ist in Abbildung 6.18 zu erkennen. Dabei sind die abgebildeten simplen rotationssymmetrischen Modelle ausnahmslos soweit reduziert worden, dass bei allen Teilen jeweils nur die zwei Mantelflächen übrig geblieben sind. Da die ausgedünnten Matrizen folglich nur noch aus den zwei Zylinderhalbschalen bestehen, werden alle sechs Teile ins gleiche Cluster verschoben. Bei der Umwandlung vom proprietären CAD-Modell zur herstellerunabhängigen IGES-Datei findet gleichzeitig auch eine Veränderung vom Volumen- zum Flächenmodell statt.

Evaluation

Wenn der gesamte hier beschriebene Ansatz mit der VRML-Methodik aus den Kapiteln 5.1, 6.1.2 und 6.2.2 verglichen wird, so wird deutlich, dass der IGES-Ansatz aufgrund der fehlenden Flächenformanalyse weniger leistungsfähig ist. Während die Anwendung der ausgedünnten IGES-Fingerprints dazu führt, dass alle Bauteile in Abbildung 6.18 dem gleichen Cluster zugewiesen werden, so leistet die VRML-Klassifikation noch eine feinere Unterscheidung zwischen Sechskant- und Zylinderkopfschrauben. Dennoch sind beide Ansätze individuell einstellbar und ihre Anwendung hängt sowohl von den Gegebenheiten in der Datenbank, als auch von den gewünschten Ergebnissen der Benutzer ab.

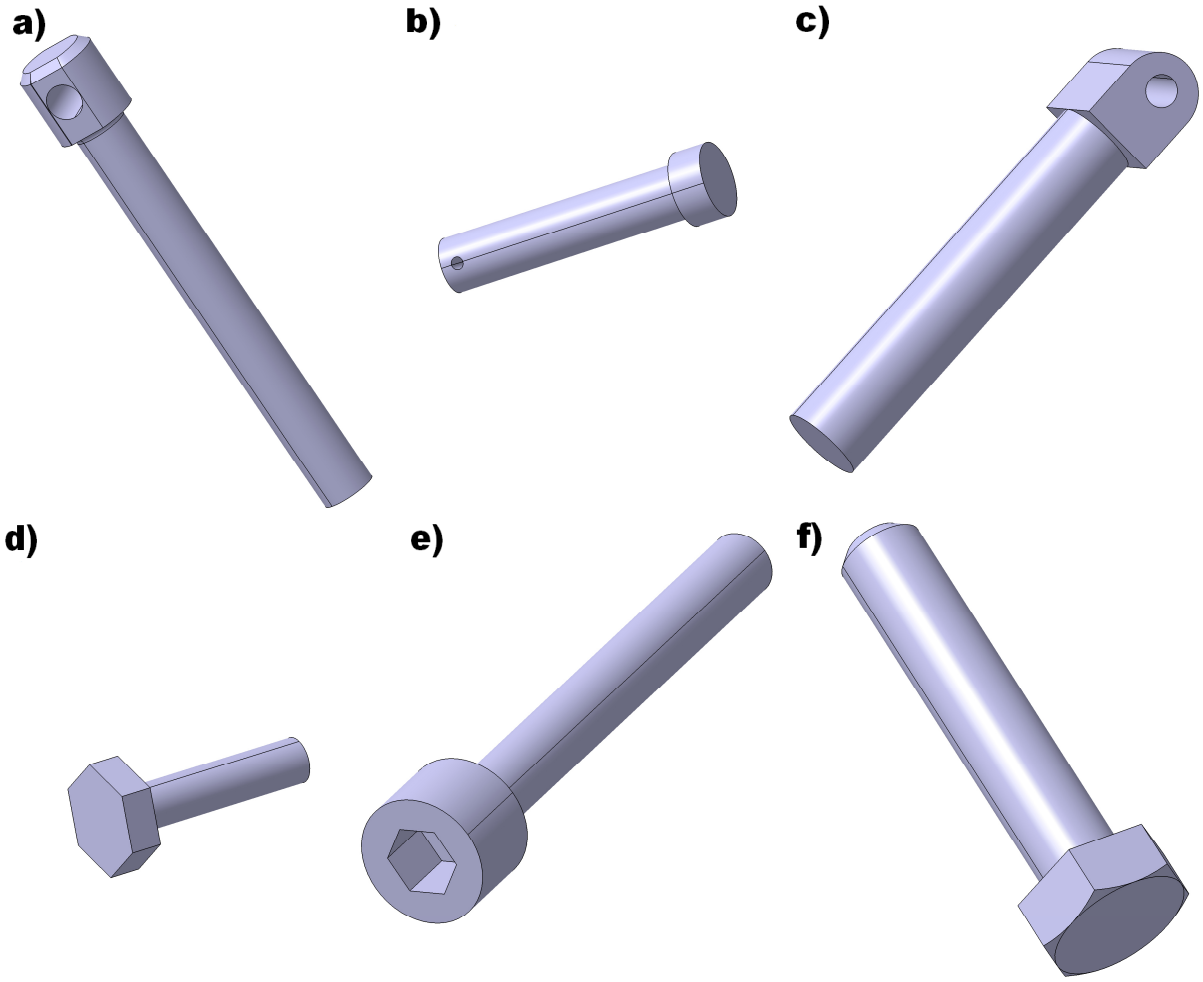

Abbildung 6.18: Sechs rotationssymmetrische Teile im gleichen Cluster

Dennoch ist anzumerken, dass CAD-Modelle im IGES-Format in der ingenieurstechnischen Anwendung gängiger sind und das vorgestellte Konzept somit in der Praxis zur Anwendung kommen könnte.

Da sowohl der IGES- als auch der VRML-Ansatz automatisch ablaufen und beide im Hinblick auf den Vorbereitungsaufwand im Gegensatz zum Attribute-Fingerprint wenig Zeit und keine zusätzlichen Dateiformate benötigen, kann die Anwendbarkeit auf große Datenmengen als praktikabel erachtet werden. Dennoch ist die Anwendung der VRML-Dateien aufgrund der Identifikation der verschiedenen Flächentypen aussagekräftiger und effektiver.

Kapitel 7. Fazit und zukünftige Arbeit

7.1 Schlussfolgerungen

Die im zweiten Teil der vorliegenden Arbeit entwickelte Methodik zur Informationsextraktion aus großen CAD-Datenbeständen zur Aufbereitung und Weiterverarbeitung soll in diesem Kapitel in Relation mit den industriellen Anwendungen in der Praxis und den wissenschaftlichen Konzepten in der Forschung gesetzt werden. Zunächst ist zu erwähnen, dass in der gesamten Entwicklung alle Schritte untersucht werden, die für eine Verbesserung der Zugänglichkeit zu den Bauteilen, die in einer gegebenen unsortierten Datenbasis enthalten sind, nötig sind. Die Einteilung in die drei Bereiche 'Informationsextraktion', 'Informationsaufbereitung' und 'Informationsweiterverarbeitung' lässt sich aus dem Gesamtkonzept schlussfolgern, das, neben einer Beschleunigung der Handhabung von großen Datenbeständen, zum Ziel hat, die Möglichkeiten, die sich durch den Aufbau der Bauteile bzw. Datenbanken bieten, aufzudecken.

Bei einer eingehenderen einzelnen Betrachtung der drei Bereiche werden die Unterschiede zu den in der Industrie in der Anwendung befindlichen Systemen und den Forschungsthemen deutlich. So bietet zwar die in Kapitel 2.1 und 2.8 untersuchte Software viele praktikable und ausgereifte Ansätze, nützliche Informationen effektiv aus großen Datenbeständen zu filtern, doch geschieht dies dort zumeist im Hinblick auf eine kommerzielle Anwendung mit dem Fokus auf ein bestimmtes proprietäres Dateiformat. Der Unterschied wird hier in Kapitel 4 verdeutlicht, in dem diverse Dateitypen untersucht werden, um sie auf ihre Kapazitäten bei der Informationsextraktion im Hinblick auf die spätere Weiterverarbeitung zu testen. Insbesondere die Ergebnisse der VRML-Anwendungen zeigen, dass sich ein leistungsstarkes Gesamtkonzept für eine Anwendung, die in geeigneter Geschwindigkeit ablaufen kann, auch aus herstellerunabhängigen Datentypen entwickeln lässt.

Die Informationsaufbereitung, die sich in Kapitel 5 unter Verwendung der gewonnenen Informationen auf die Erstellung von Bauteilsignaturen bzw. Fingerprints konzentriert, kann mit den internen Abläufen der professionellen PDM-Programme verglichen werden, auch wenn die Algorithmen nicht zugänglich sind. Insbesondere bei der Ähnlichkeitssuche, die auch bei dreidimensionalen Modellen in den PDM-Programmen in anwendungsfreundlicher Geschwindigkeit abläuft, kommen zuvor automatisch erstellte Fingerprints, die als Informationsextrakt anzusehen sind, zur Anwendung.

Die hier entwickelten Signaturen können folglich besser mit den in Kapitel 3.8 beschriebenen theoretischen Ansätzen verglichen werden, bei denen die Abläufe der Algorithmen bekannt sind. Die drei verschiedenen Fingerprints auf Basis unterschiedlicher Inputinformationen zeigen, dass ein gewisses Potential beispielsweise im Vergleich zu den Signaturen, die ausschließlich die geometrische Erscheinung der Bauteile einbeziehen, enthalten ist. Die Idee, gesamte Features bzw. Bauteileigenschaften für die Erstellung von Fingerprints zu verwenden, führt in den folgenden Schritten der Informationsweiterverarbeitung zu guten Ergebnisse, die insbesondere von den PDM-Systemen, die die Bauteiltopologien nicht von den Maximalabmessungen trennen, nicht geleistet werden können.

Der Bauteil- bzw. Signaturvergleich, der im Zuge der Informationsweiterverarbeitung bei den Suchmaschinen und Klassifikationsalgorithmen angewendet wird, führt bei den vorgestellten Ergebnissen neben einer Verwendung der konventionellen Such- und Klassifikationsaufgaben auch zu den erläuterten spezialisierten Fällen, in denen beispielsweise die Vergrößerungs- oder Verkleinerungsfaktoren skalierter Bauteile berechnet sowie die Detektion von Subgraphen ermöglicht werden. Hierbei handelt es sich um innovative Sonderfälle, die von den untersuchten PDM-Programmen außer Acht gelassen werden und auch von den wissenschaftlichen Ansätzen in Kapitel 3 nur zum Teil angesprochen werden.

Die Tatsache, dass alle Programme nach der Definition der Eingabeparameter automatisch ablaufen und keine weitere Beaufsichtigung benötigen, bekräftigt das Potential für eine Anwendung auf große Datenmengen in der Praxis. Die im Intermezzo definierten Forschungsziele wurden im Zuge der Kapitel 4 bis 6 analysiert und umgesetzt. Als Fazit ist zu erwähnen, dass in Kapitel 5 drei grundlegend verschiedene Ansätze zur Erstellung von Fingerprints entwickelt wurden. Auf Basis der in Kapitel 4 gesammelten Informationen ist es jedoch auf einfache Art und Weise möglich sowohl die vorhandenen Methoden zu erweitern, als auch neuartige Konzepte umzusetzen. Ähnliches gilt für Kapitel 6. Die Algorithmen zur Entwicklung von Suchmaschinen und Klassifikationsprogrammen liefern neuartige funktionstüchtige Herangehensweisen und können dennoch bei der Implementierung weiterentwickelt oder auf spezielle Anwendungsfälle zugeschnitten werden.

In Kapitel 7.2 werden noch weitere zukünftige Forschungsaufgaben angesprochen, die für eine marktreife Umsetzung bzw. Ausreifung des entwickelten Konzeptes durchzuführen sind.

7.2 Zukünftige Forschungsaufgaben

Die Ergebnisse der Bauteilsuche und Klassifikation der drei verschiedenen Methoden zur Bestimmung von Fingerprints haben gezeigt, dass alle Ansätze sowohl Stärken als auch Schwächen besitzen und ihre Verwendungsmöglichkeiten somit von den gegebenen Umständen sowie den gewünschten Ergebnissen abhängen. Im Idealfall sollten somit die Stärken der unterschiedlichen Konzepte zu einem Ansatz vereint werden, der sowohl die positiven Eigenschaften der Attribute-Fingerprints als auch die der Adjazenzmatrizen beinhaltet und in einer einzigen Signatur miteinander verbindet.

Sollte aus dem Gesamtkonzept eine Software entwickelt werden, die zur Marktreife gebracht wird und in der industriellen Anwendung auf große Datenmengen zugeschnitten werden muss, so sollte zum einen für eine komfortable Bedienung eine GUI entwickelt werden, die dem Benutzer neben einer übersichtlichen Anwendung auch die Möglichkeit bietet, Suchergebnisse oder Klassifikationsvorschläge abzuspeichern und wieder aufzurufen.

Generell ist davon auszugehen, dass alle Programme noch erweitert und verbessert werden können. Um die Ausführgeschwindigkeiten zu erhöhen, sollten die Möglichkeiten einer Implementierung in anderen Programmiersprachen sowie eine Verarbeitung mithilfe paralleler Algorithmen in Betracht gezogen werden. Auch durch die Umsetzung der entwickelten Konzepte auf verschiedenen Plattformen mit unterschiedlichen Betriebssystemen könnten neue Herausforderungen entstehen.

Zudem besteht insbesondere bei den Attribute-Fingerprints noch ein großes Potential zur Erweiterung und Verbesserung der Berechnungsalgorithmen. Die Vorgehensweise zur Bestimmung der Zahlenwerte in den einzelnen Kategorien könnte unter Miteinbeziehung weiterer Dateiformate noch weiter ausgereift werden. Neben einer eingehenden Untersuchung anderer Datentypen, wie z.B. dem STEP-Format, das hier nicht behandelt wurde, könnten außerdem weitere Kategorien hinzugefügt werden, die den Attribute-Fingerprint weiter detaillieren.

Auch die Einfärbung der VRML-Dateien besitzt das Potential, um weitere Typen erweitert zu werden, so dass eine genauere Unterscheidung zwischen den einzelnen Flächen und ihren Verbindungen zueinander stattfinden kann.

Eine solche Flächenunterscheidung und die darauffolgende Übertragung in eine Adjazenzmatrix ist ggf. auch mit anderen Dateiformaten, wie z.B. dem IGES- oder dem STEP-Format realisierbar. Wenn neben den Flächentypen, ihren Abmaßen sowie Verbindungen und Distanzen noch weitere Informationen im Fingerprint untergebracht werden könnten, so würde eine Unterscheidung im Zuge des Bauteilvergleichs

bei der Klassifikation noch weiter ins Detail gehen und die Möglichkeit einer Bildung noch schärferer Klassengrenzen entstehen.

Da hier, bei der Strukturbaumanalyse und Featureextraktion eine Konzentration auf die CAE-Software CATIA stattgefunden hat, liegt eine Übertragung auf die Programme anderer Hersteller und Softwareapplikationen nahe. Aus den Gegebenheiten zur Makroprogrammierung, die von den meisten professionellen CAE-Systemen zur Verfügung gestellt werden, könnten neben den hier extrahierten Features und Bauteileigenschaften noch weitere Alternativen entstehen, die am Ende in die Fingerprintentwicklung und somit in die Bauteilsuche und -klassifikation mit einfließen. Weiterhin sind auch theoretische Entwicklungen von neuartigen Ansätzen denkbar, die denen in Kapitel 3 ähneln. Nach der Entwicklung derartiger Konzepte ist eine sofortige technische Umsetzung nicht immer sinnvoll oder praxistauglich. Je nachdem wie leistungsfähig die benötigte Technik ist, könnten beispielsweise lange Rechenzeiten eine praktikable Anwendung einschränken bzw. verhindern. In einem solchen Fall ist der Schwerpunkt auf die Weiterentwicklung der Technik zu legen, so dass die wissenschaftlichen Konzepte letztendlich dennoch realisiert werden können.

Anhang

A: Programmübersicht

Programm	Eingabe	Ausgabe
stl.py	CATParts	pro CATPart eine STL-Datei und eine Textdatei
voxel.py	STL-Dateien	pro STL-Datei eine MSH-Datei und eine Textdatei
read_tree.py	CATParts	pro CATPart eine XML-Datei
create_stl.py	CATParts, pro CATPart drei SVG- und drei TIF-Dateien	pro CATPart drei STL-Dateien
stl_to_bool.py	CATParts, pro CATPart drei SVG- und drei STL-Dateien	pro CATPart ein rekonstruiertes Bool-CATPart
drilling.py	CATParts, pro CATPart ein Bool-CATPart, sechs SVG- und sechs TIF-Dateien	pro CATPart ein Bool-CATPart mit Bohrungen
pocket.py	CATParts, pro CATPart ein Bool-CATPart mit Bohrungen, sechs SVG- und sechs TIF-Dateien	pro CATPart ein Bool-CATPart mit Taschen
vrml.py	CATParts	pro CATPart eine VRML-Datei, eine eingefärbte VRML-Datei und eine Textdatei
iges.py	CATParts	pro CATPart eine IGES-Datei
attributes.py	CATParts, pro CATPart je eine STL-, Voxel- und VRML-Textdatei sowie zwei XML-Dateien	pro CATPart eine Fingerprint-Textdatei
vrml_matrix.py	eingefärbte VRML-Dateien	pro eingefärbte VRML-Datei eine Matrix-Textdatei und evtl. Visualisierung
simple_vrml_matrix.py	eingefärbte VRML-Dateien	pro eingefärbte VRML-Datei eine ausgedünnte Matrix-Textdatei und evtl. Visualisierung
iges_matrix.py	IGES-Dateien	pro IGES-Datei eine Matrix-Textdatei und evtl. Visualisierung

simple_iges_matrix.py	IGES-Dateien	pro IGES-Datei eine ausgedünnte Matrix-Textdatei und evtl. Visualisierung
attributes_euklid.py	CATParts, pro CATPart eine Fingerprint-Textdatei	Konsolenausgabe
attributes_pearson.py	CATParts, pro CATPart eine Fingerprint-Textdatei	Konsolenausgabe
vrml_matrix_compare.py	VRML-Matrix-Textdateien	Konsolenausgabe
vrml_matrix_statistic.py	VRML-Matrix-Textdateien	Konsolenausgabe
vrml_matrix_subgraph.py	VRML-Matrix-Textdateien	Ordner 'Match' mit Matrix-Kopien
vrml_matrix_scaled.py	VRML-Matrix-Textdateien	Konsolenausgabe
simple_vrml_matrix_compare.py	ausgedünnte VRML-Matrix-Textdateien	Konsolenausgabe
attributes_cluster.py	Fingerprint-Textdateien	Ordnersystem mit Fingerprint-Kopien
vrml_cluster_quan.py	VRML-Matrix-Textdateien	Ordnersystem mit VRML-Matrix-Kopien
vrml_cluster_qual.py	VRML-Matrix-Textdateien	Ordnersystem mit VRML-Matrix-Kopien
subgraph_vrml_cluster_quan.py	VRML-Matrix-Textdateien	Ordnersystem mit VRML-Matrix-Kopien
subgraph_vrml_cluster_qual.py	VRML-Matrix-Textdateien	Ordnersystem mit VRML-Matrix-Kopien
simple_vrml_cluster.py	ausgedünnte VRML-Matrix-Textdateien	Ordnersystem mit ausgedünnten VRML-Matrix-Kopien
iges_cluster.py	IGES-Matrix-Textdateien	Ordnersystem mit IGES-Matrix-Kopien
simple_iges_cluster.py	ausgedünnte IGES-Matrix-Textdateien	Ordnersystem mit ausgedünnten IGES-Matrix-Kopien

Tabelle A.1: Programmübersicht

B: Flussdiagramme

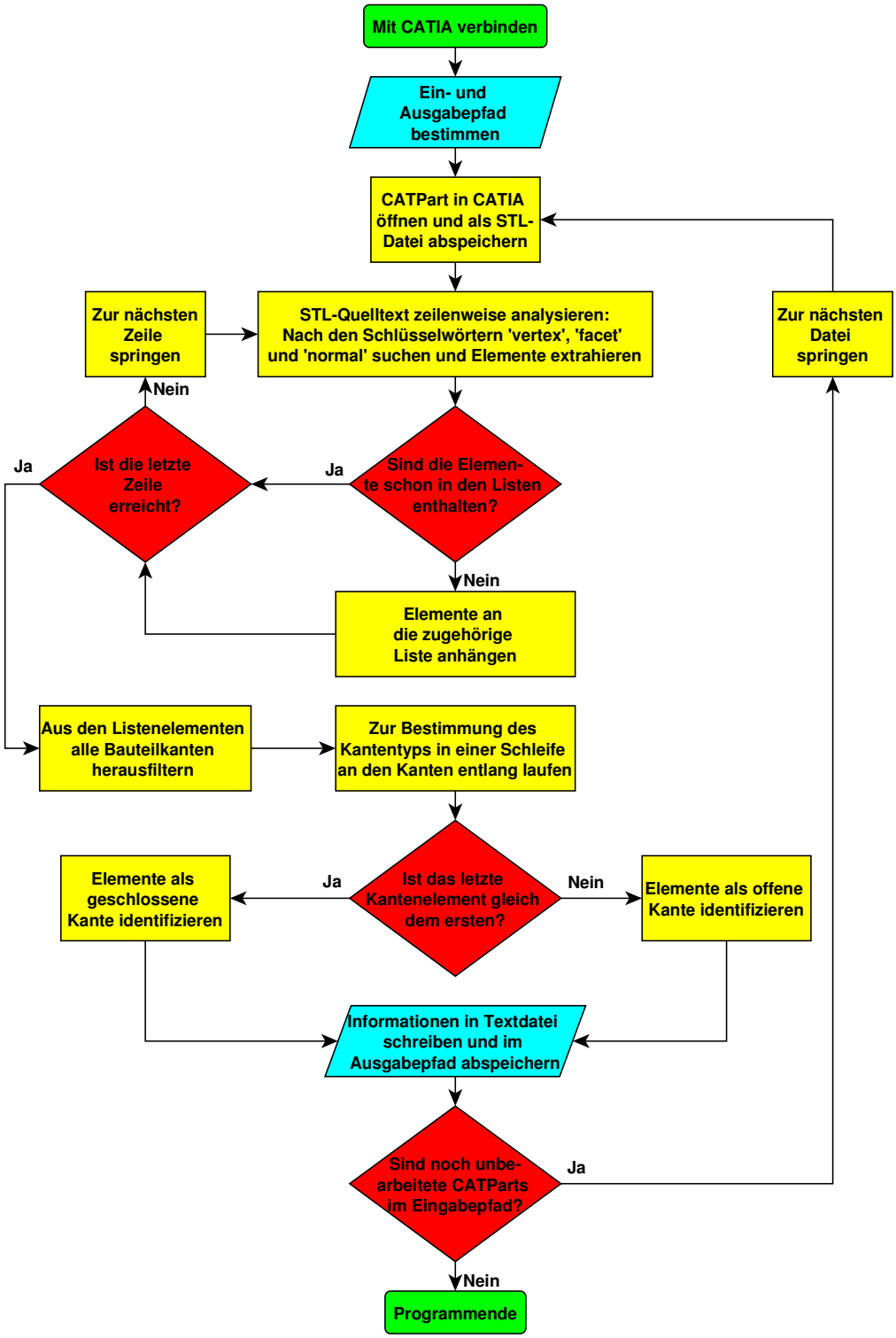

Abbildung B.1: stl.py

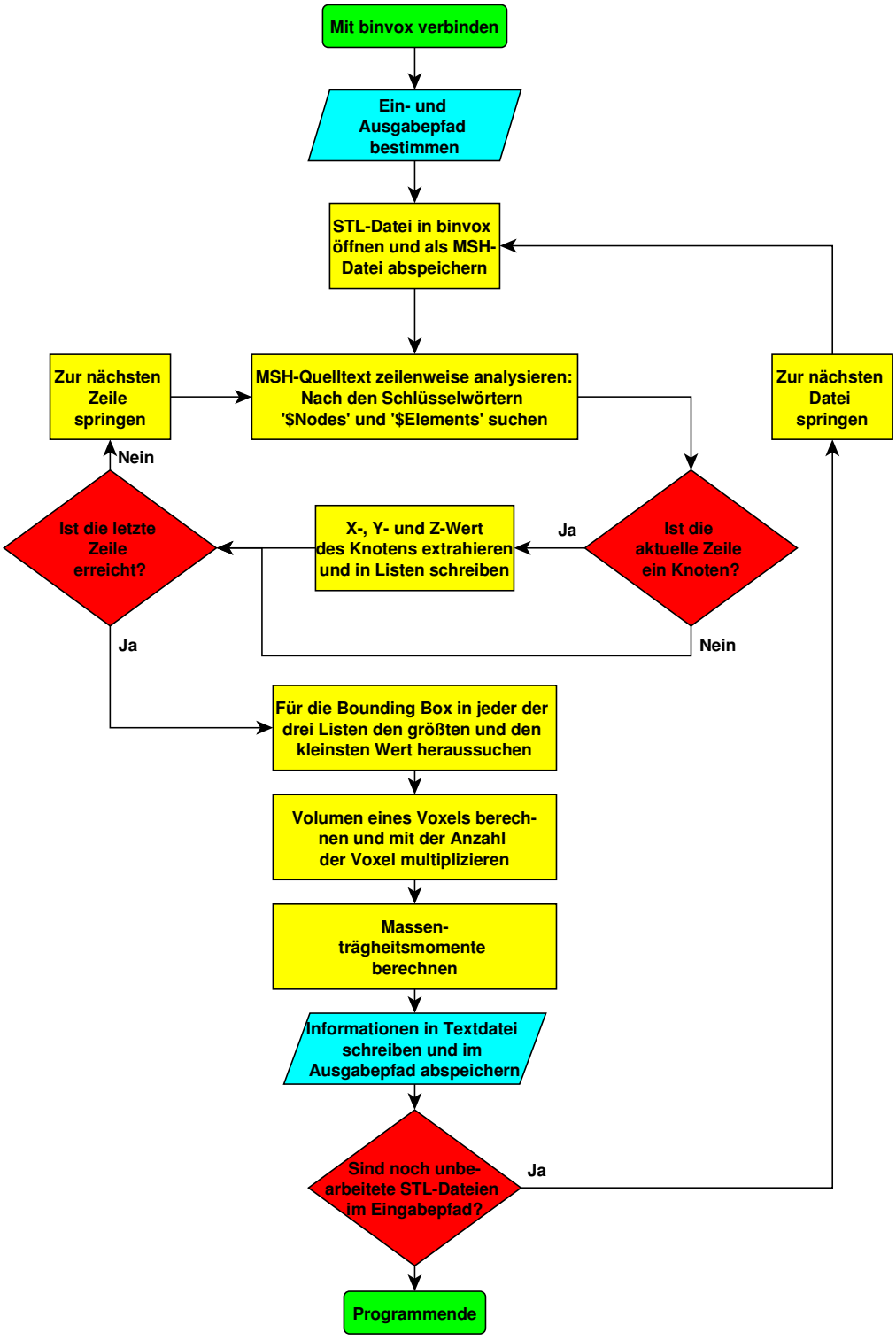

Abbildung B.2: voxel.py

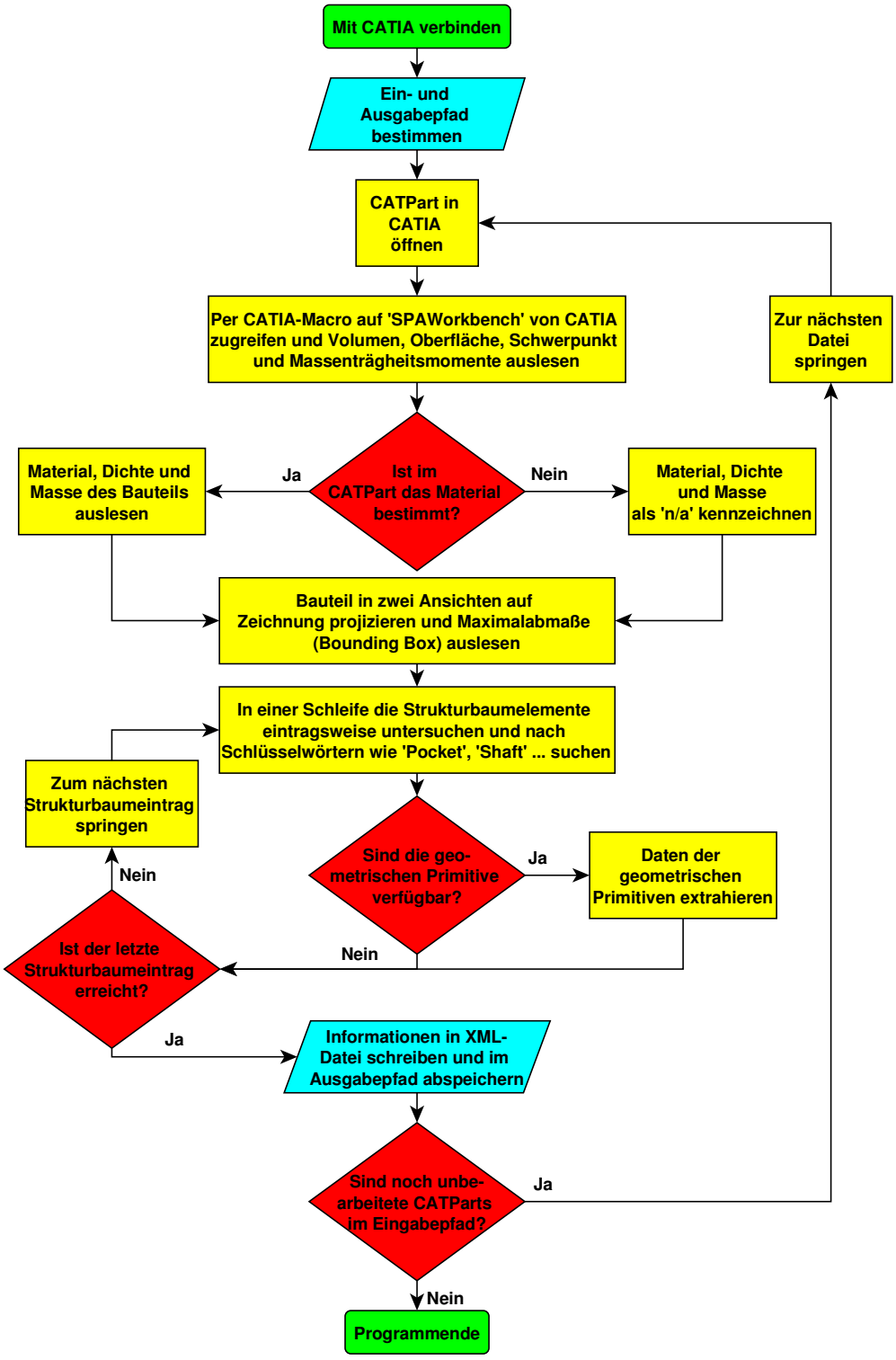

Abbildung B.3: read_tree.py

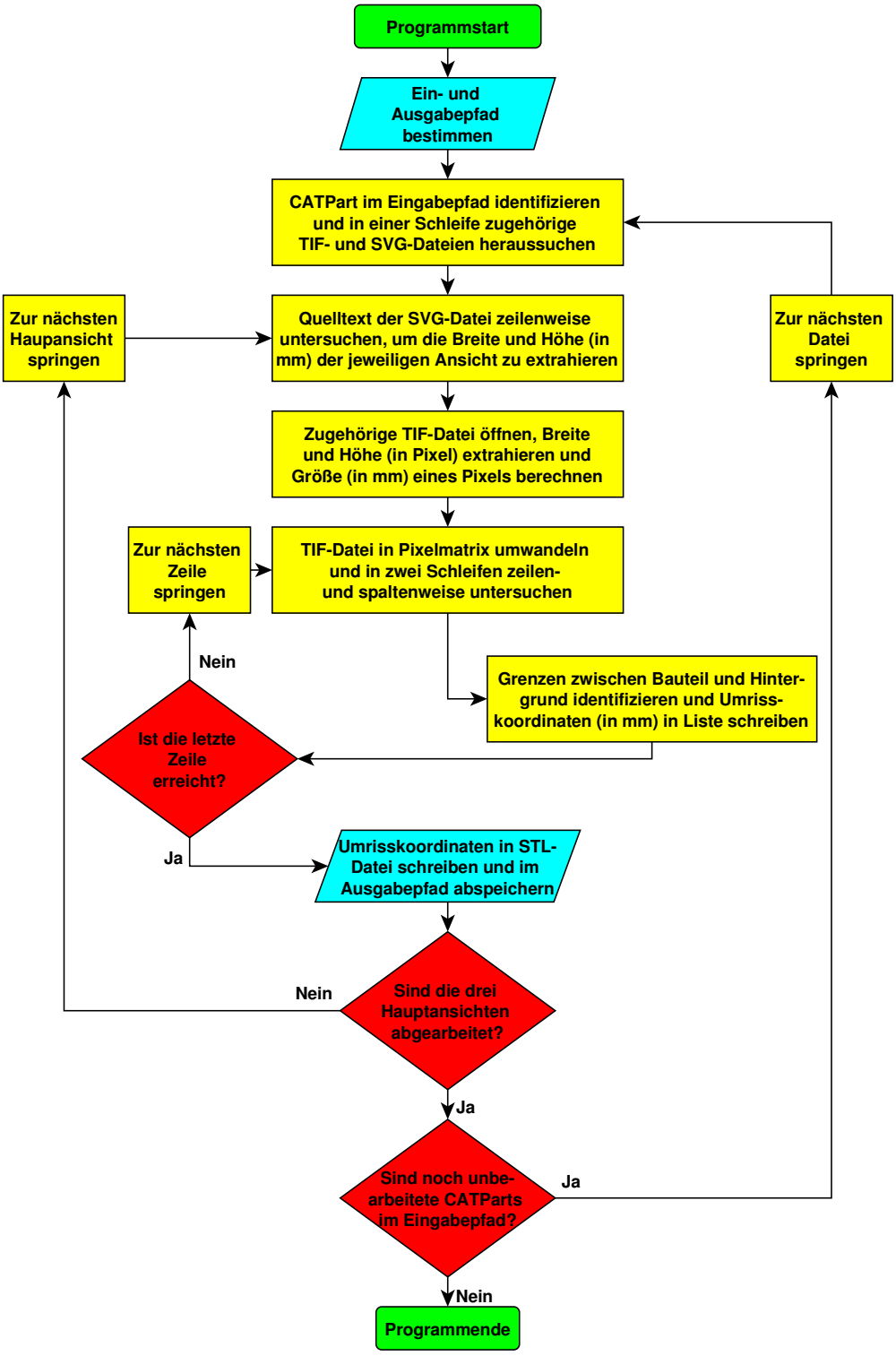

Abbildung B.4: create_stl.py

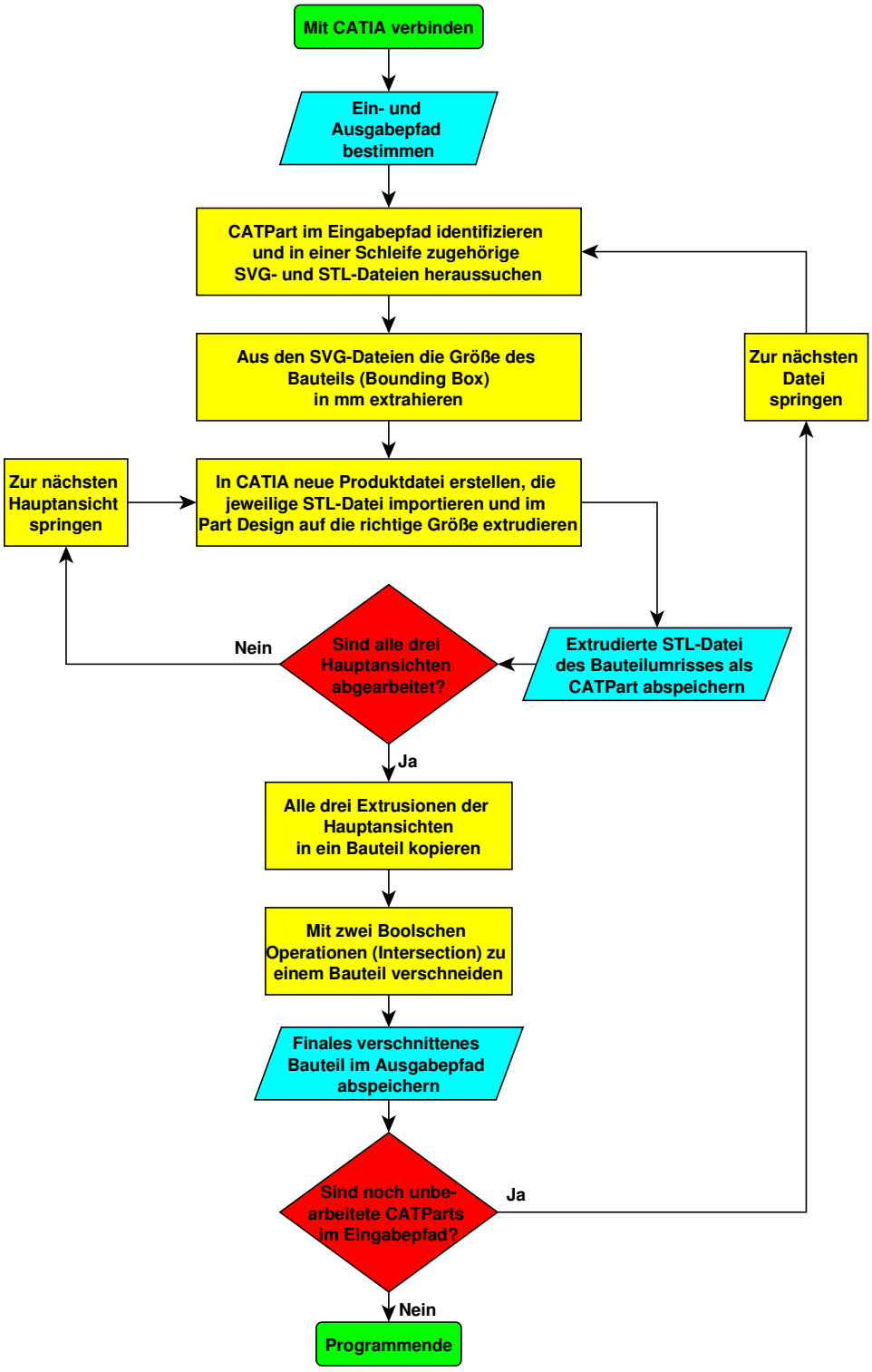

Abbildung B.5: stl_to_bool.py

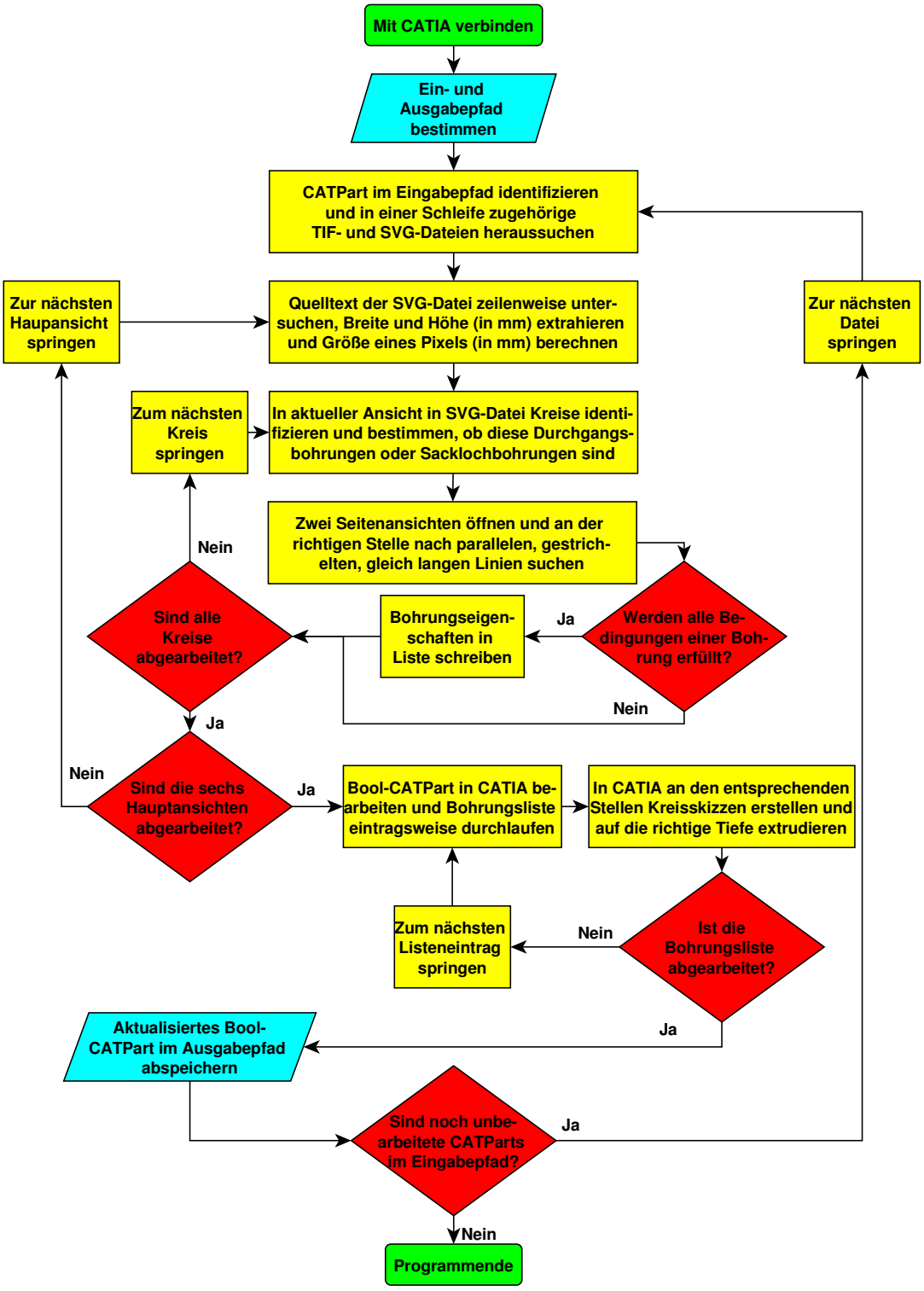

Abbildung B.6: drilling.py

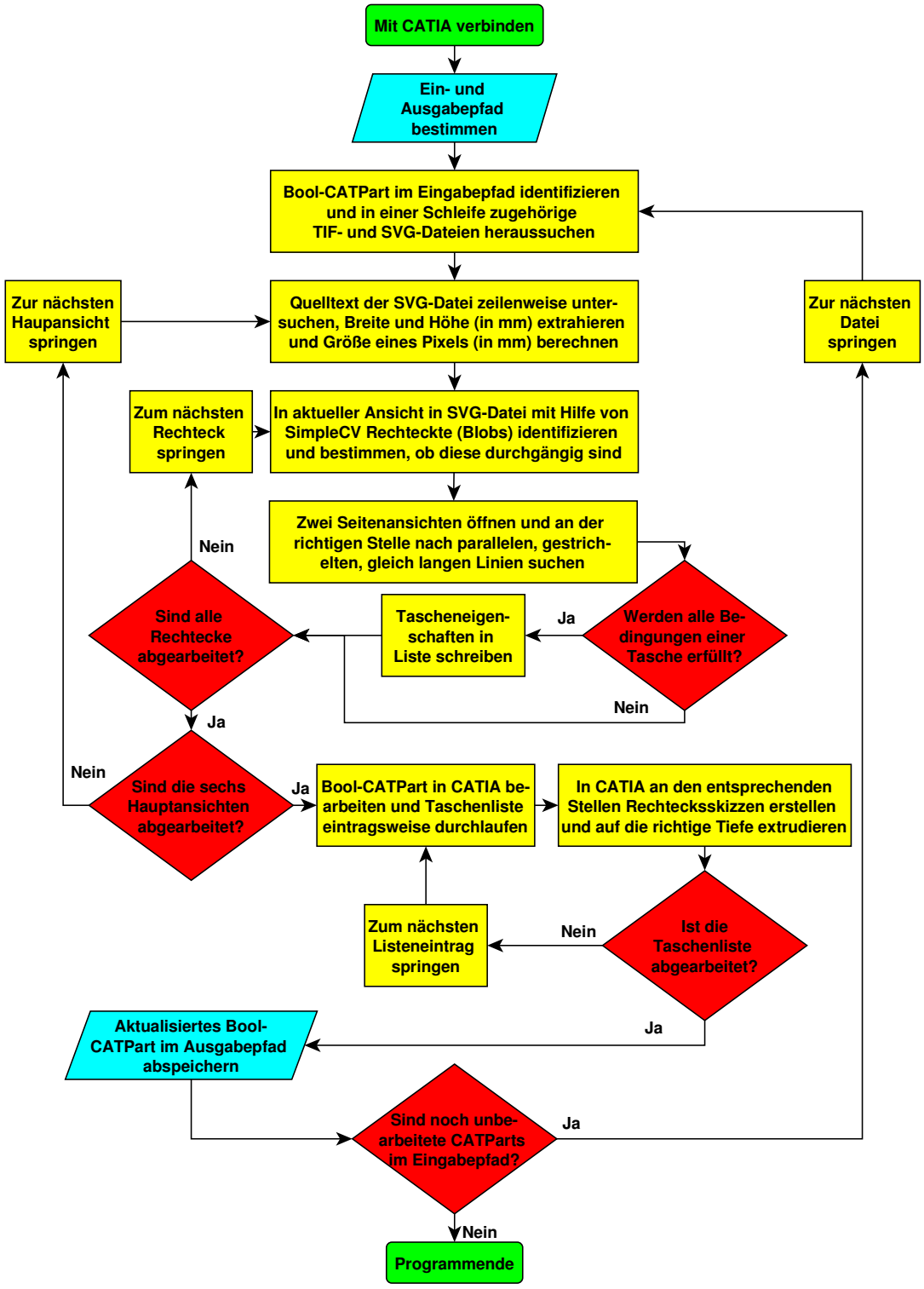

Abbildung B.7: pocket.py

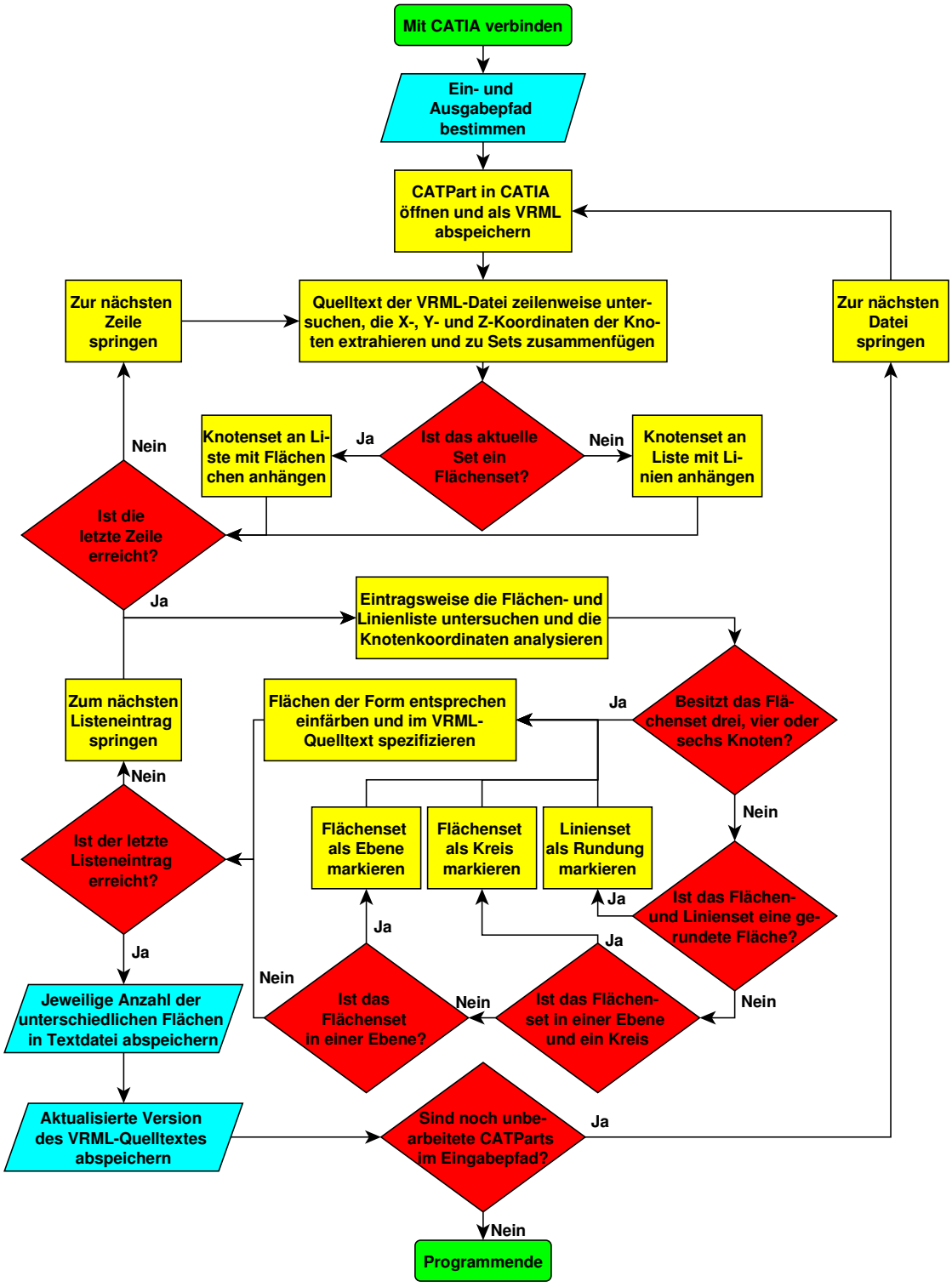

Abbildung B.8: vrml.py

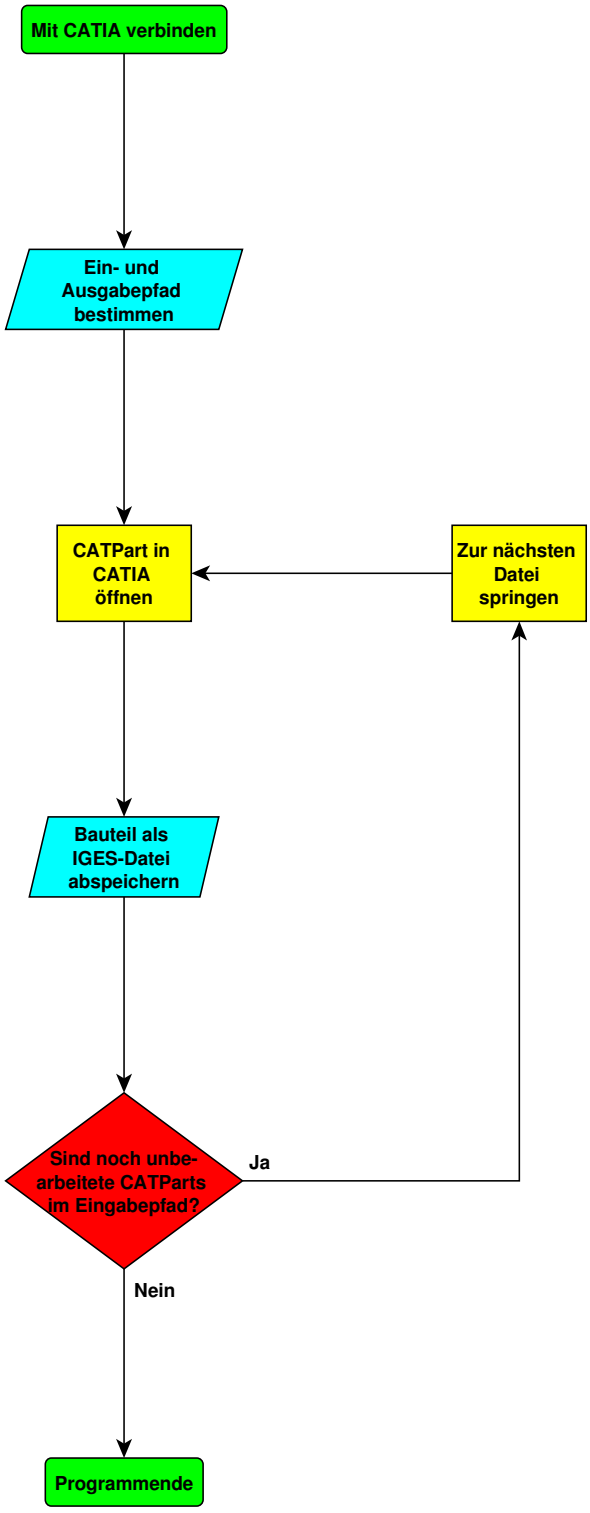

Abbildung B.9: iges.py

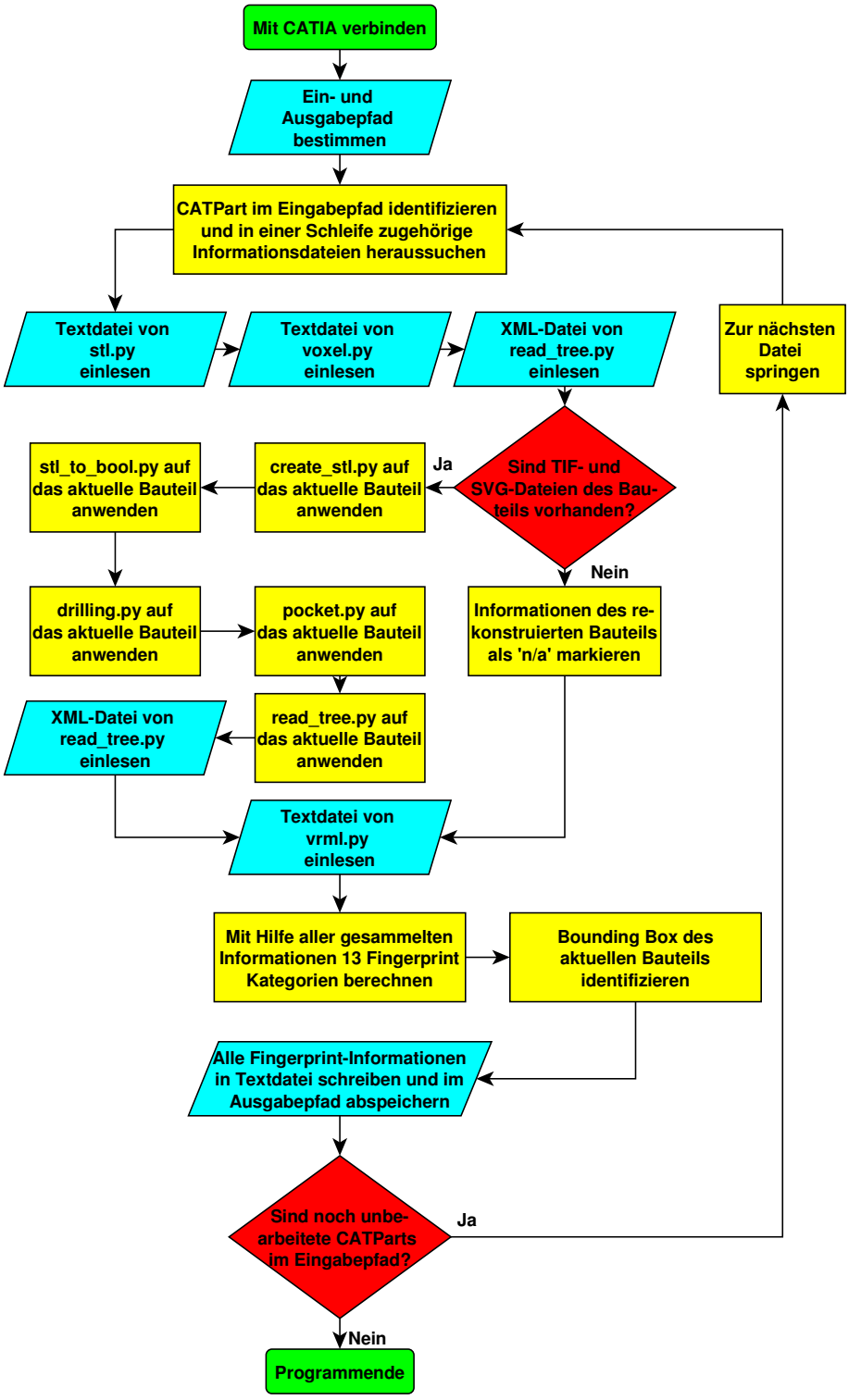

Abbildung B.10: attributes.py

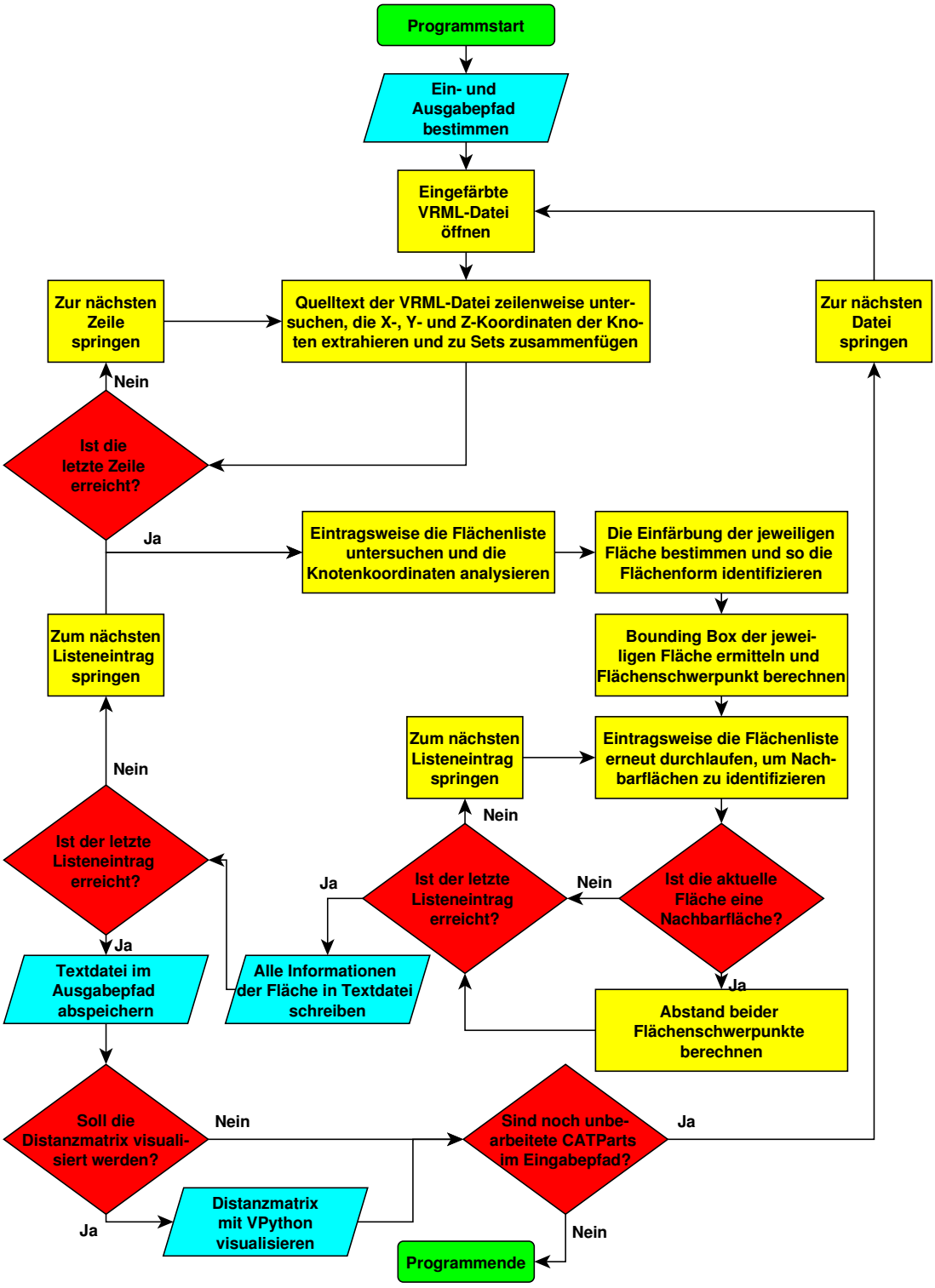

Abbildung B.11: vrml_matrix.py

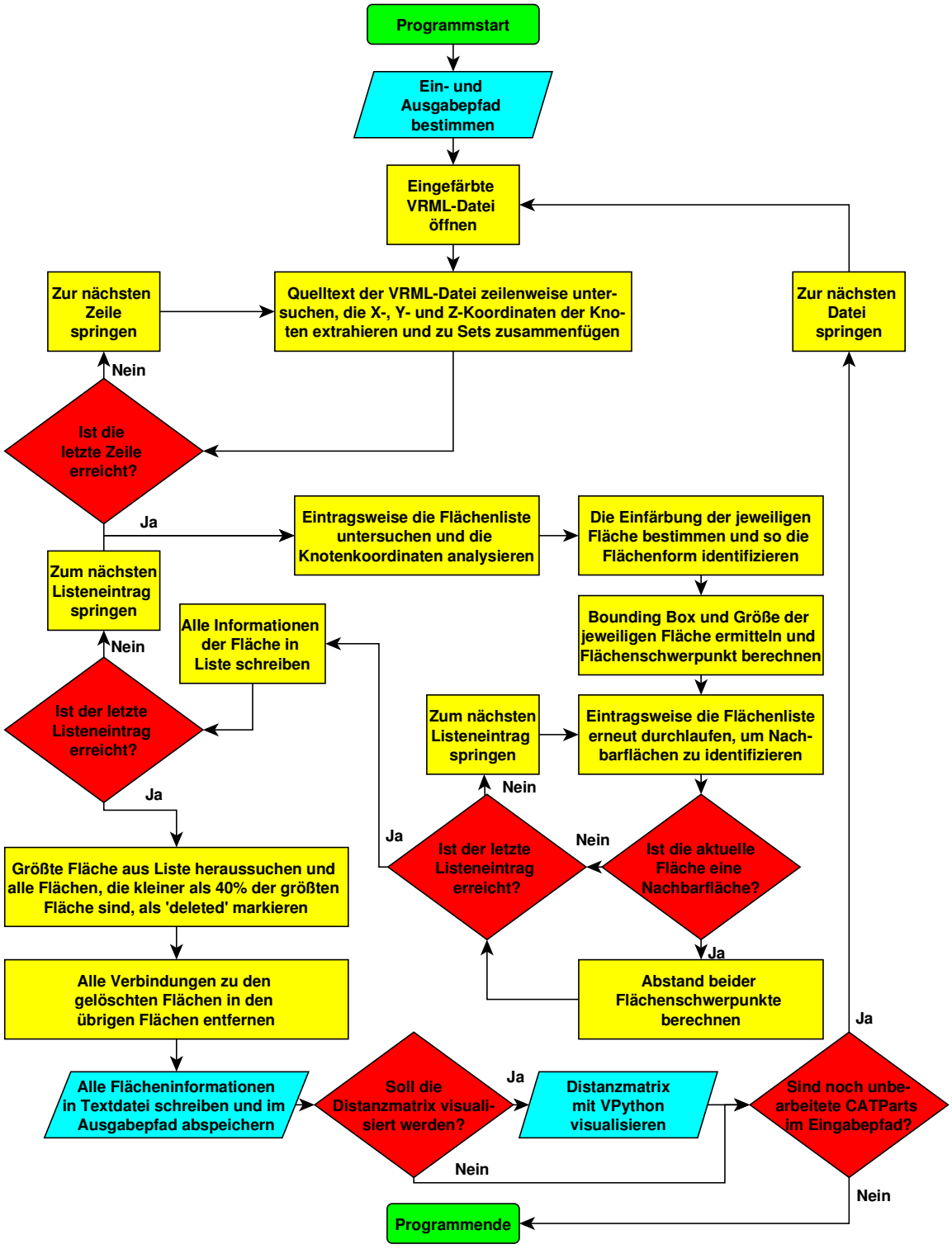

Abbildung B.12: simple_vrml_matrix.py

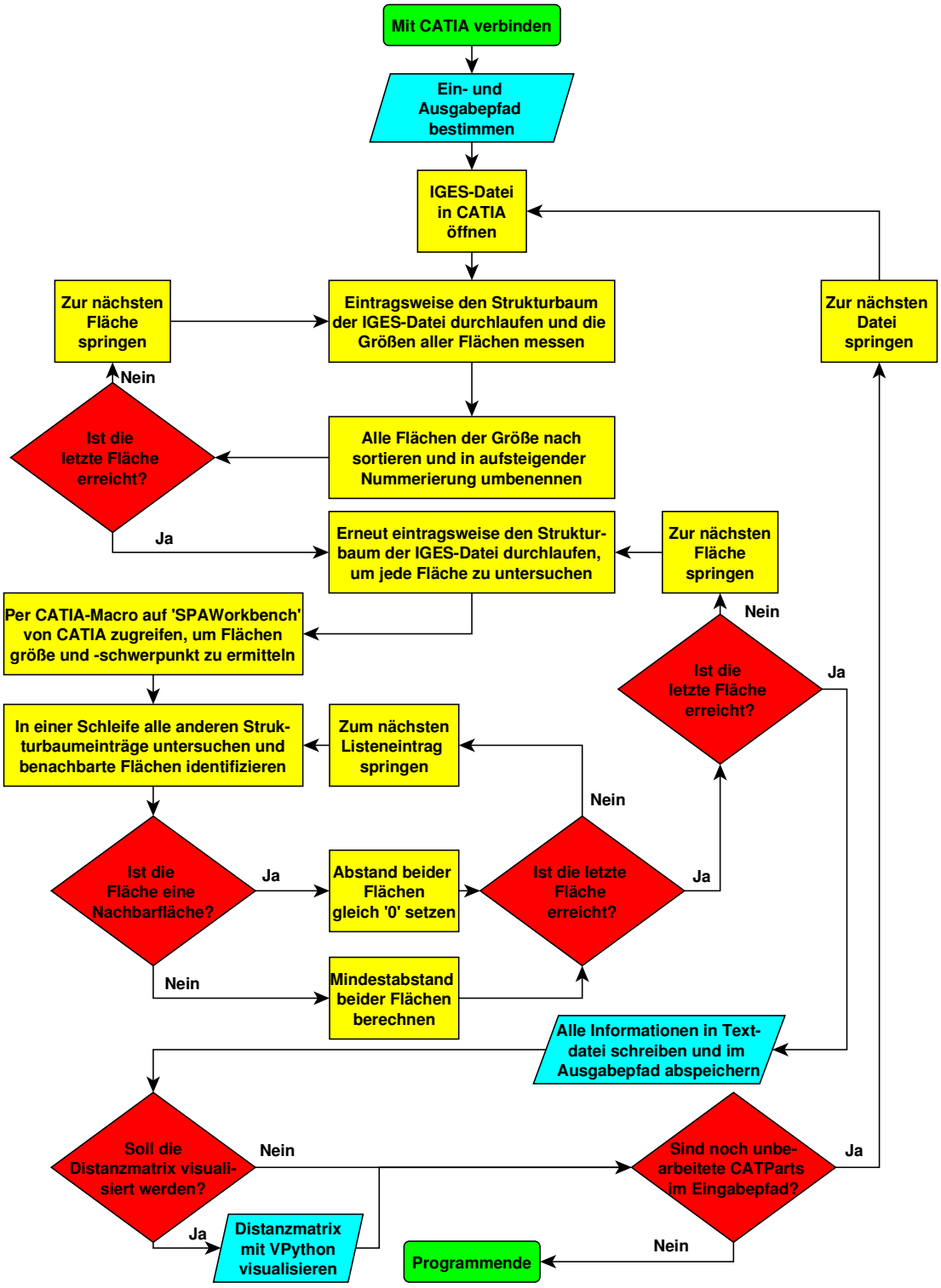

Abbildung B.13: iges_matrix.py

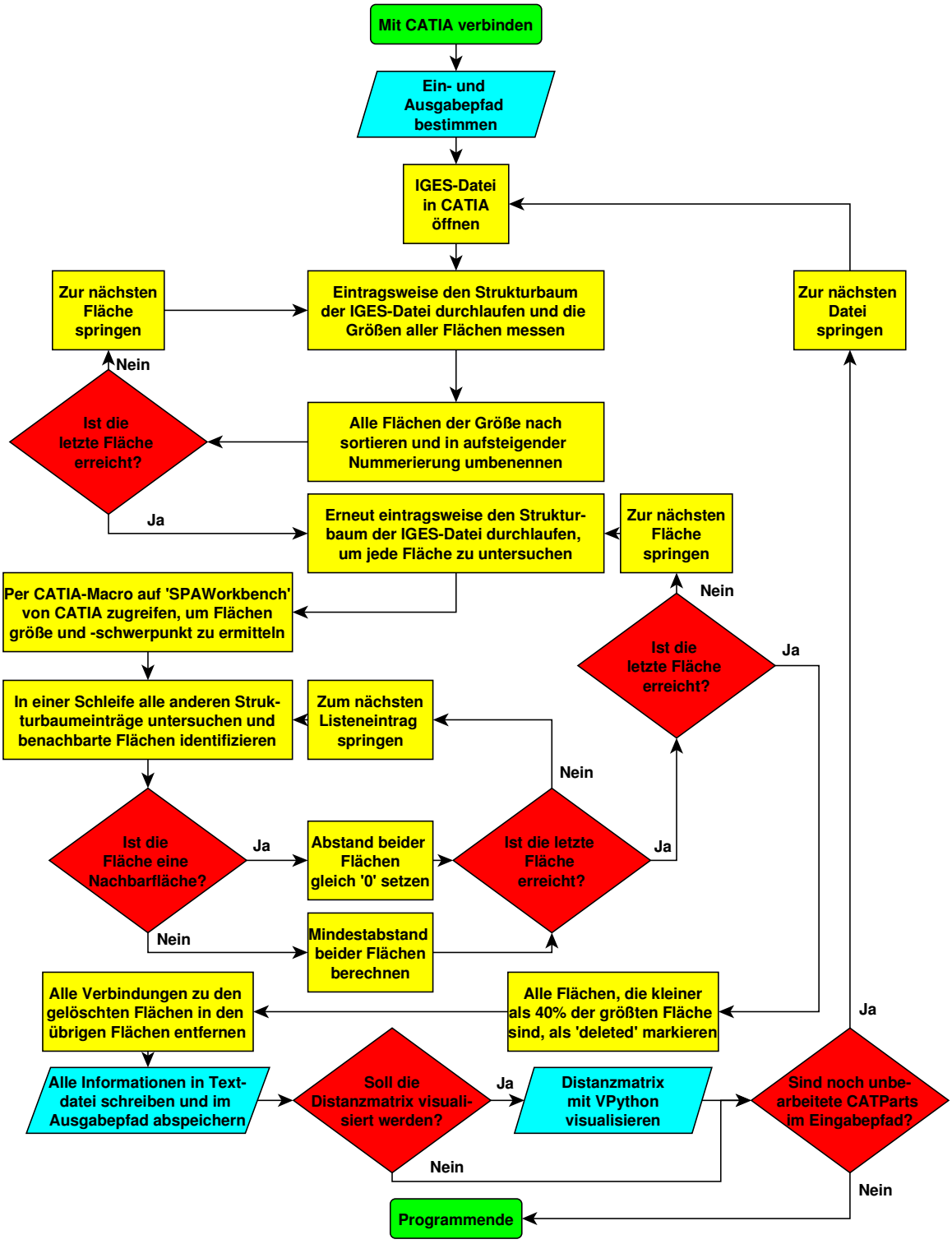

Abbildung B.14: simple_iges_matrix.py

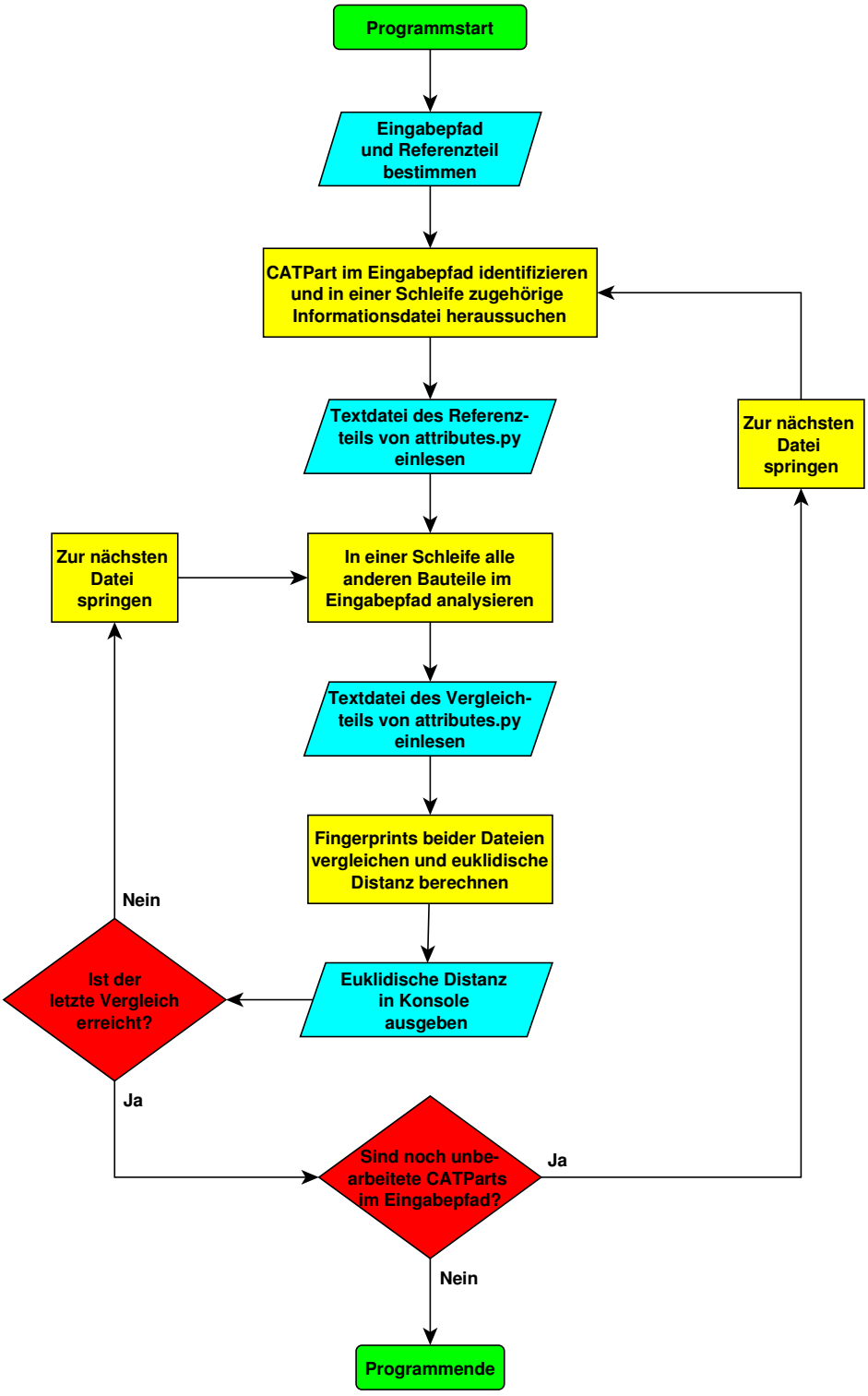

Abbildung B.15: attributes_euklid.py

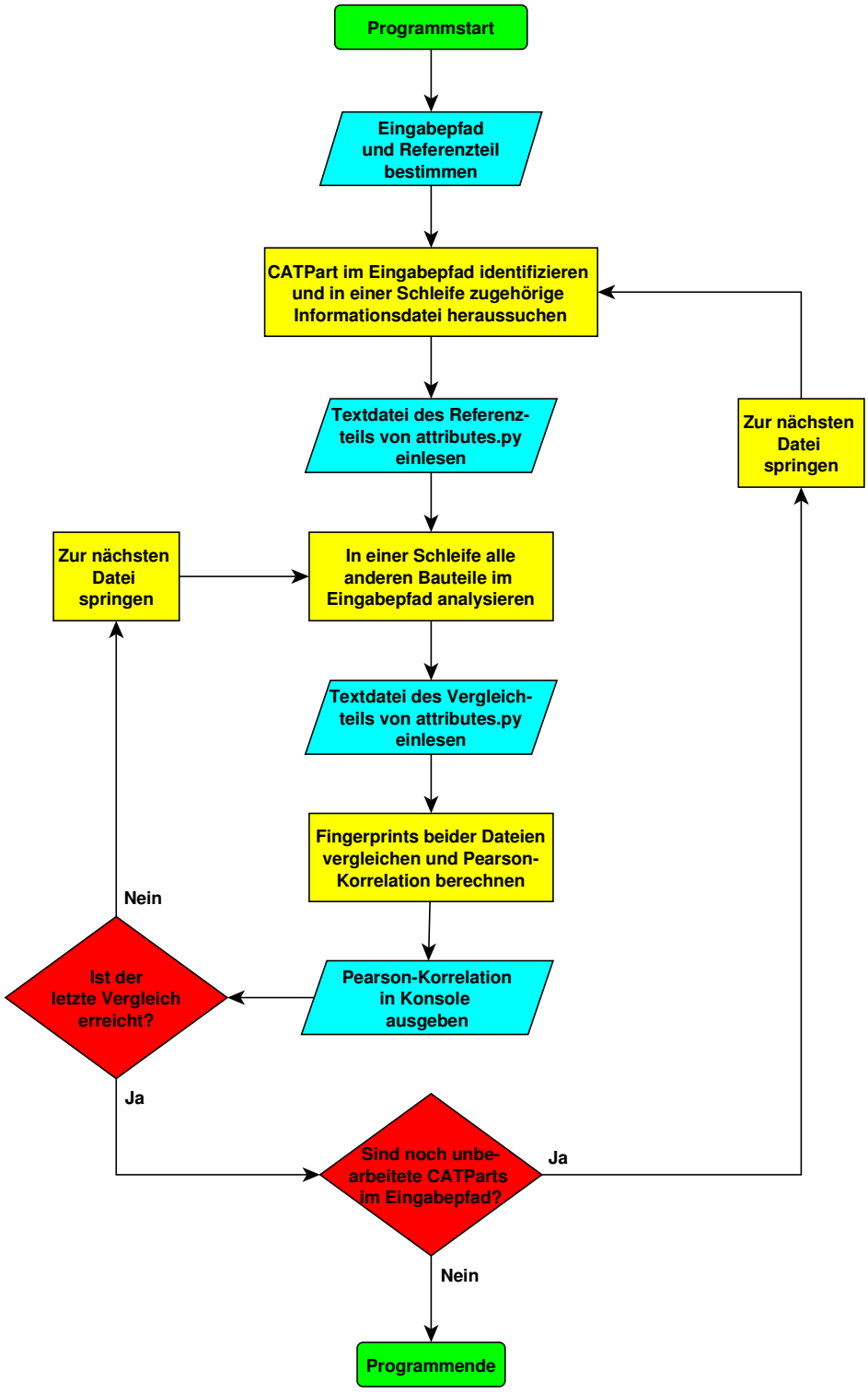

Abbildung B.16: attributes_pearson.py

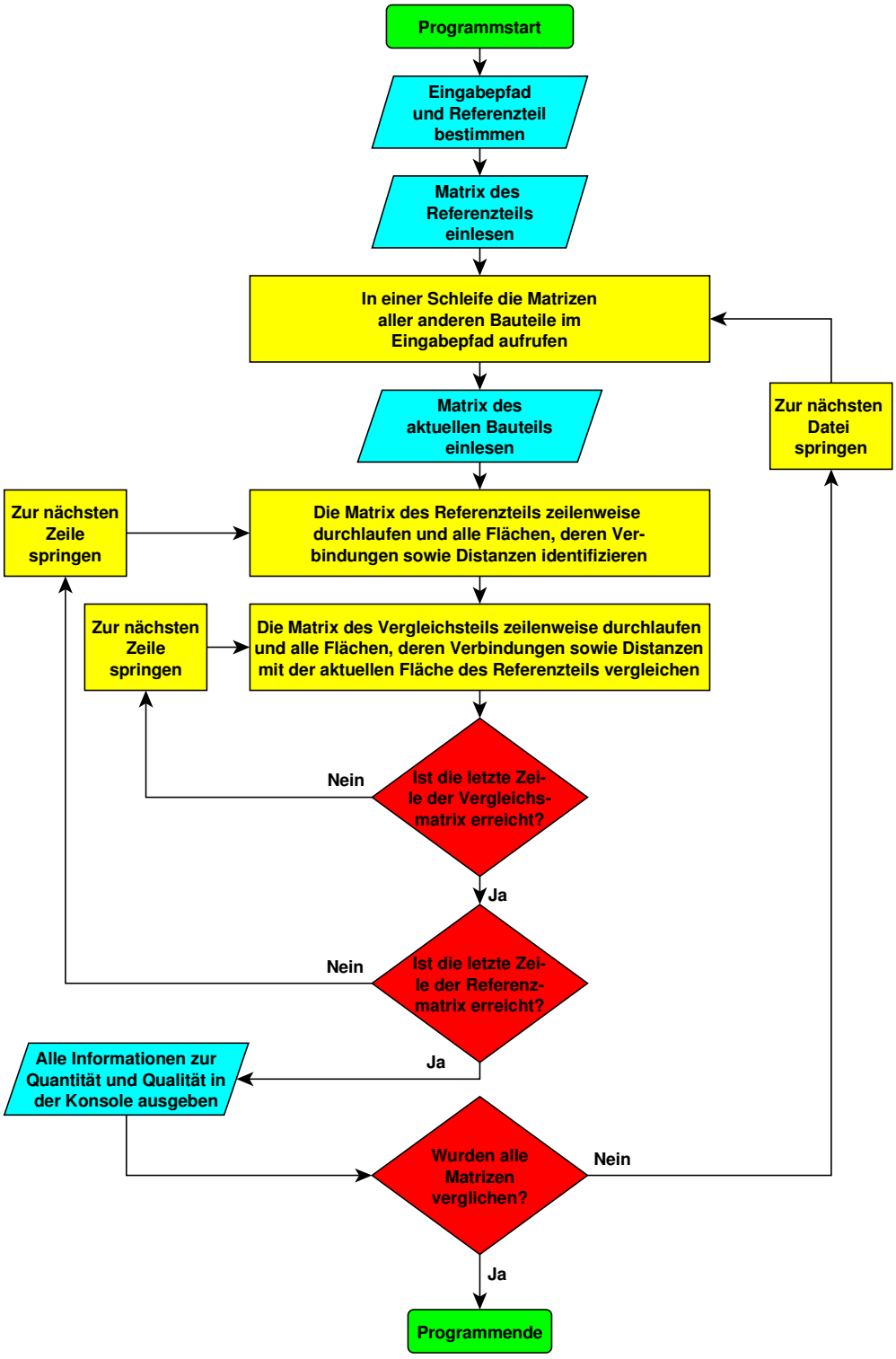

Abbildung B.17: vrml_matrix_compare.py

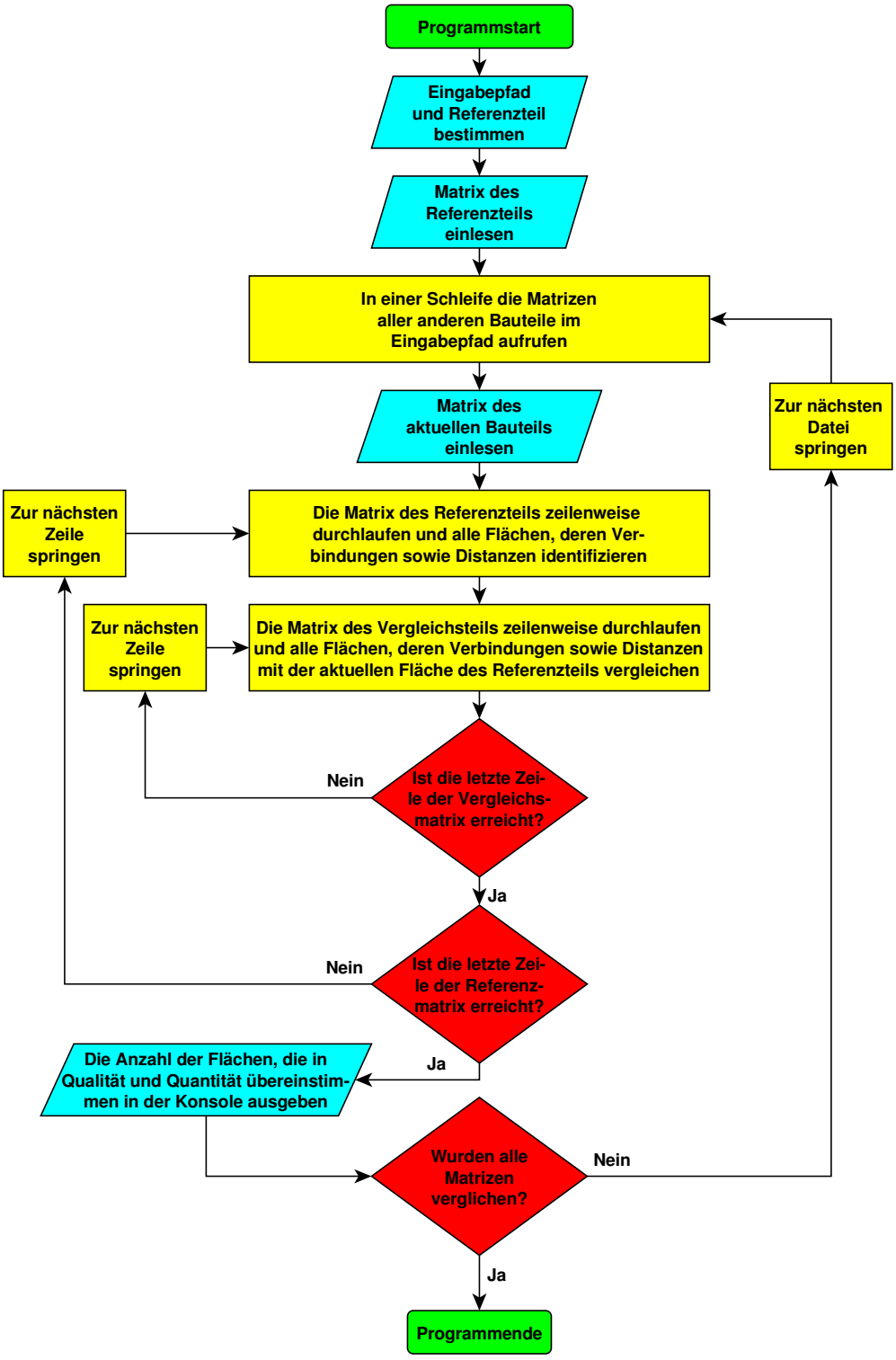

Abbildung B.18: vrml_matrix_statistic.py

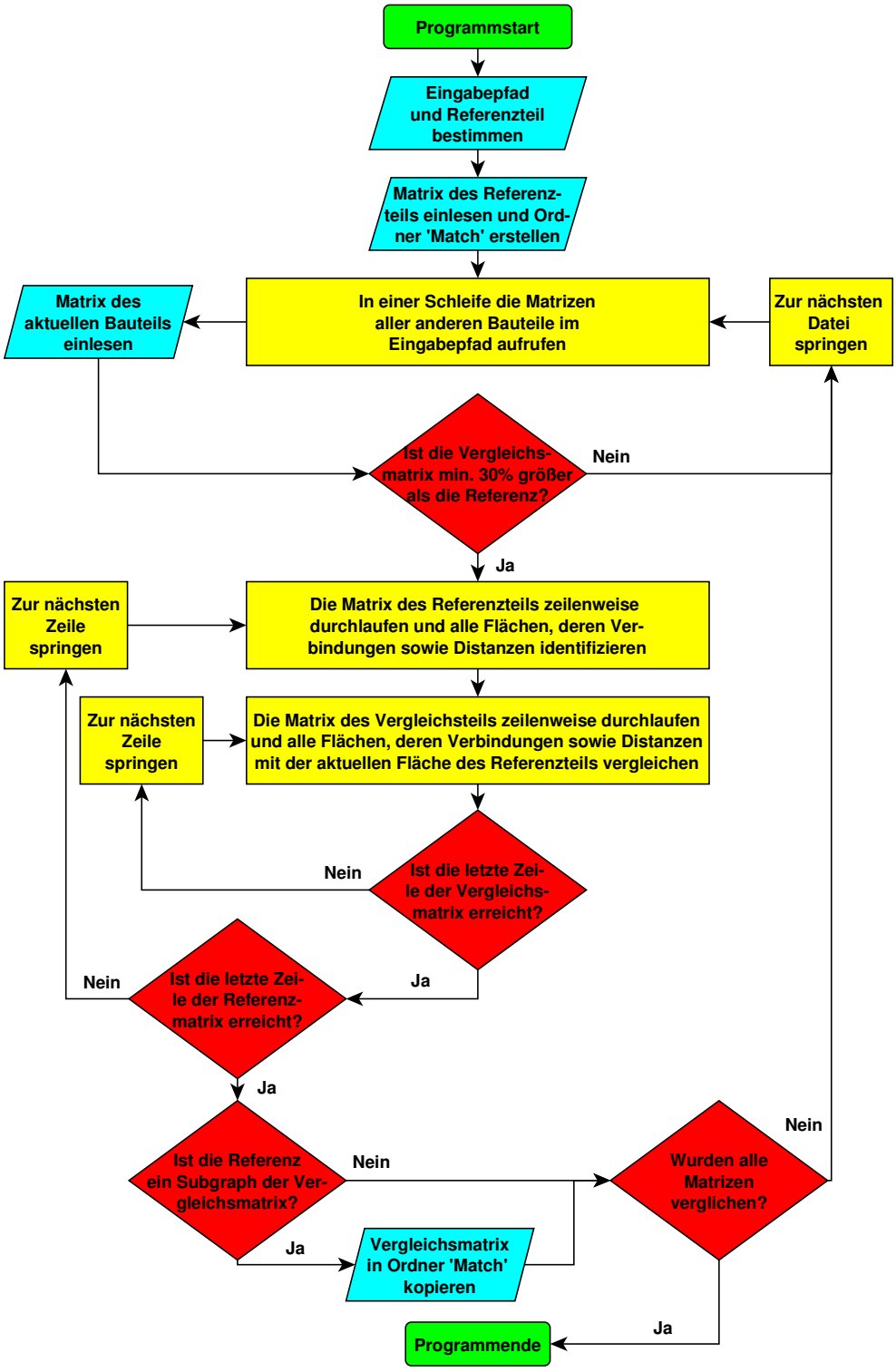

Abbildung B.19: vrml_matrix_subgraph.py

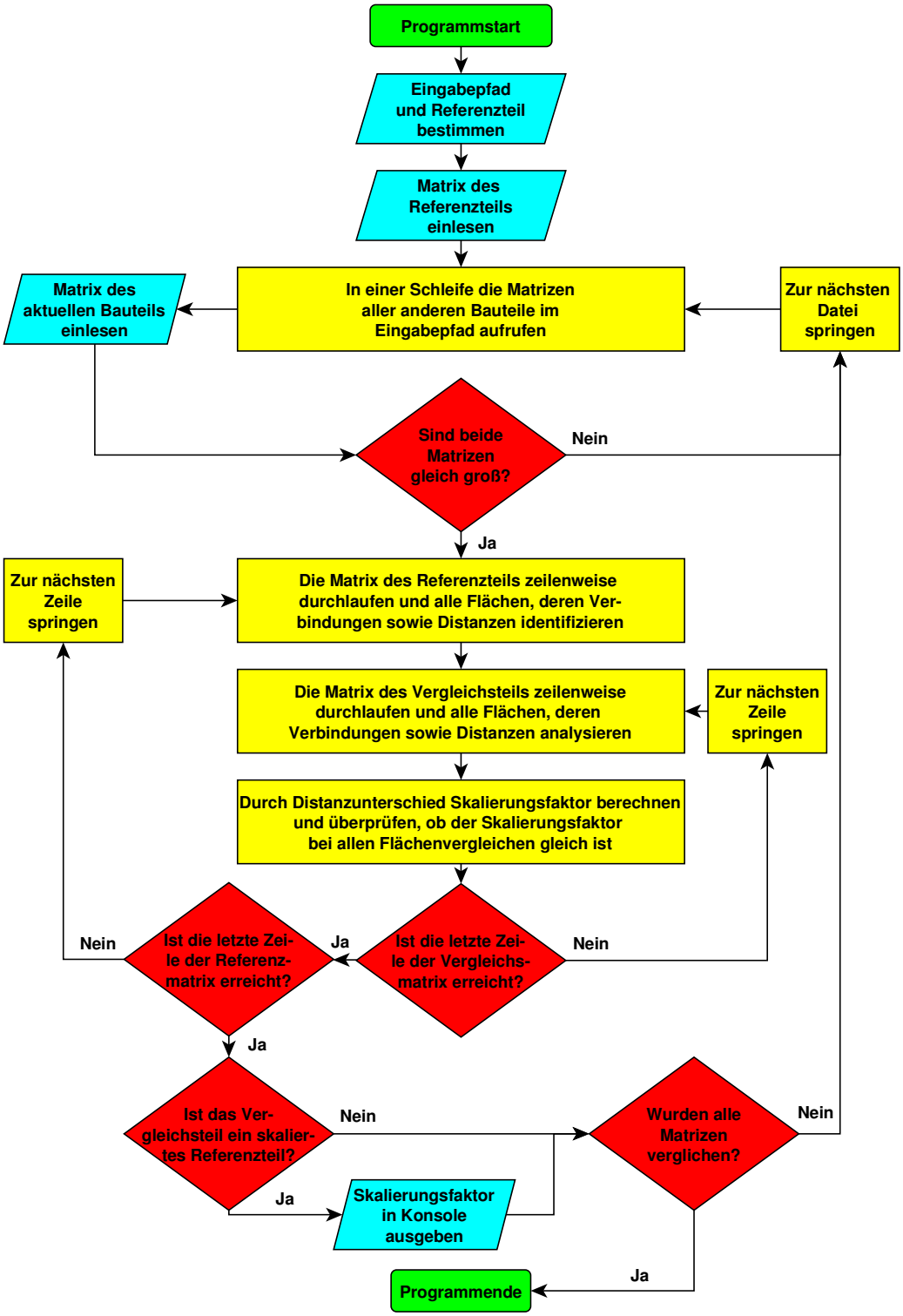

Abbildung B.20: vrml_matrix_scaled.py

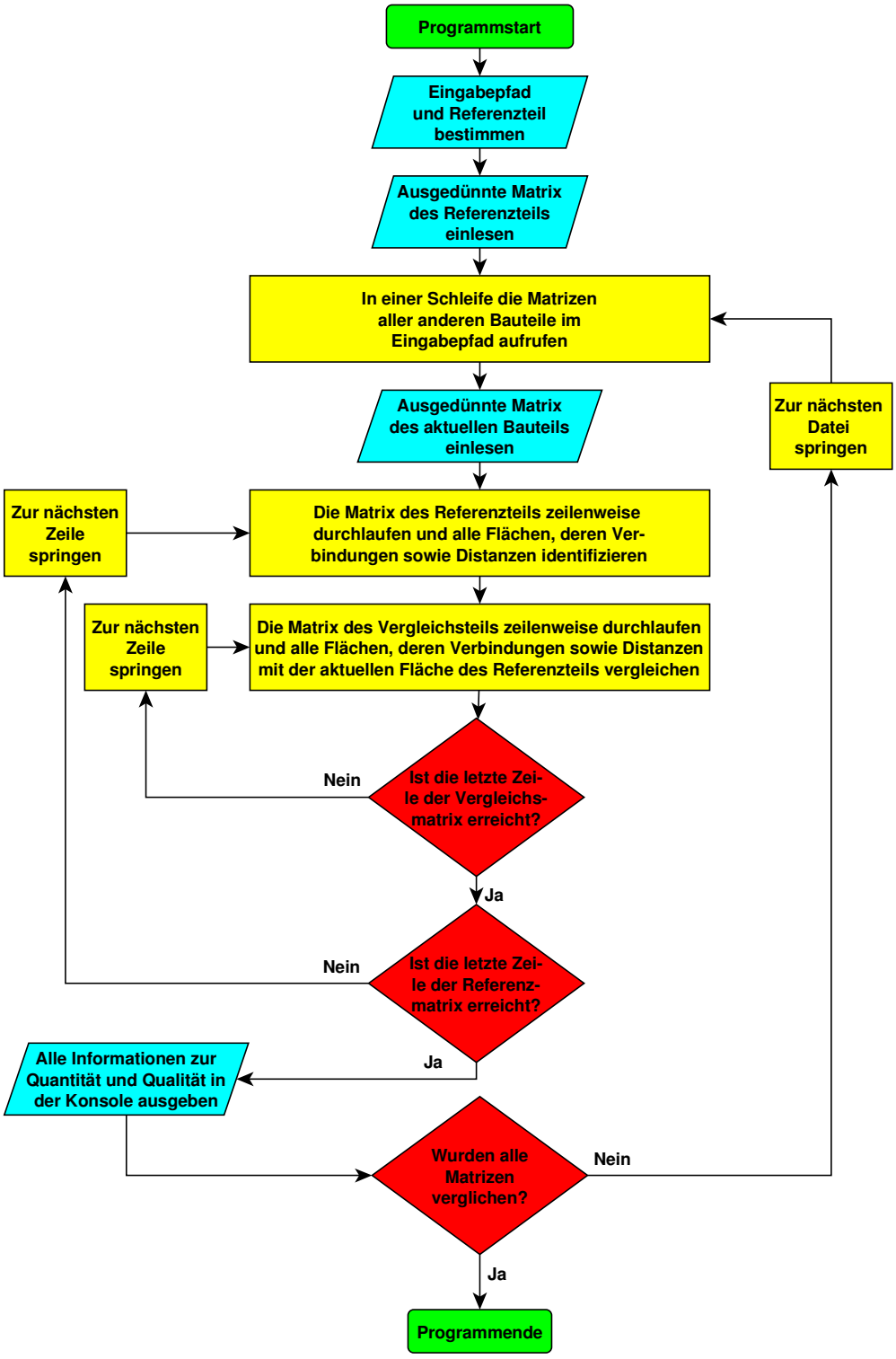

Abbildung B.21: simple_vrml_matrix_compare.py

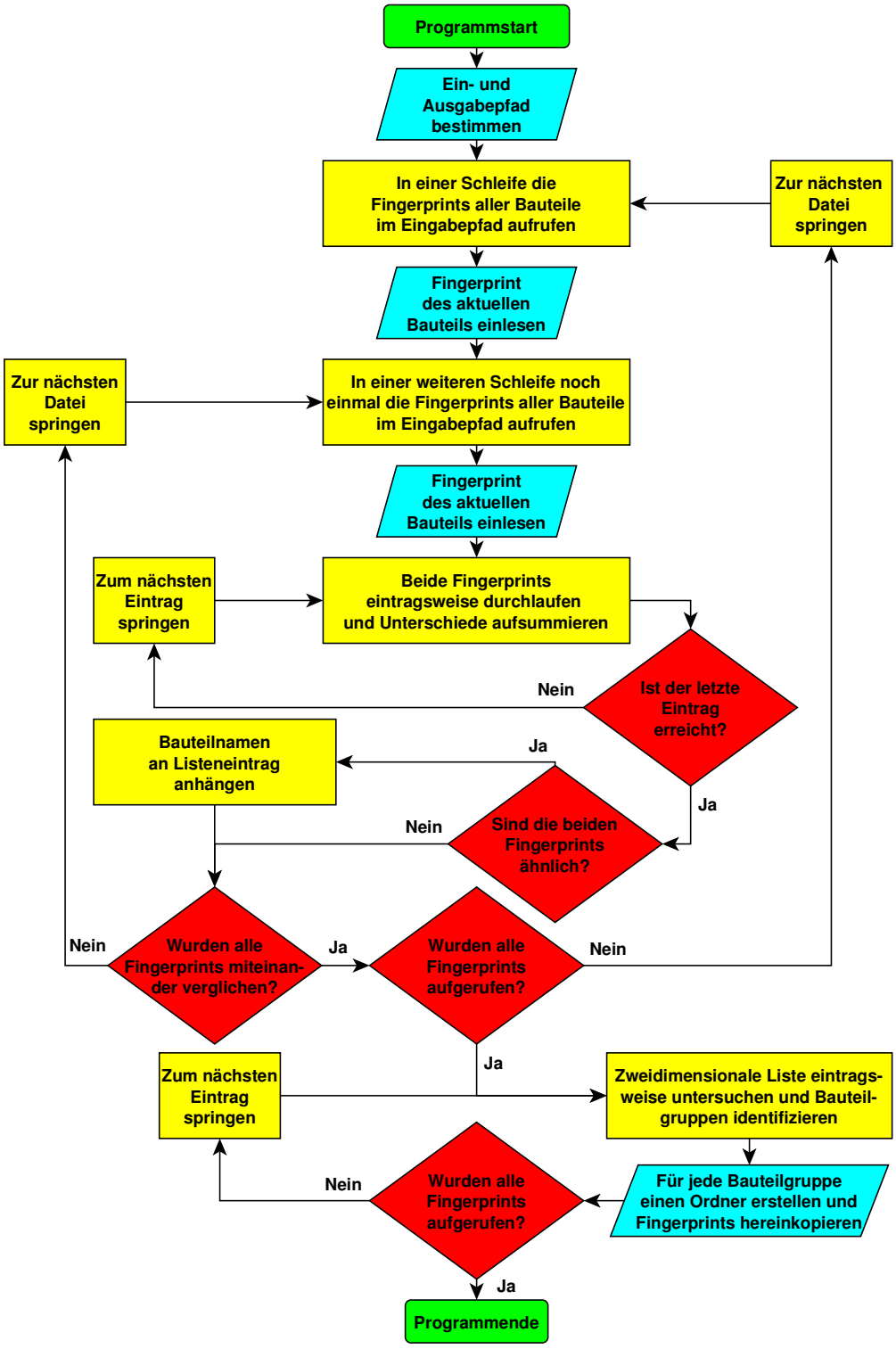

Abbildung B.22: attributes_cluster.py

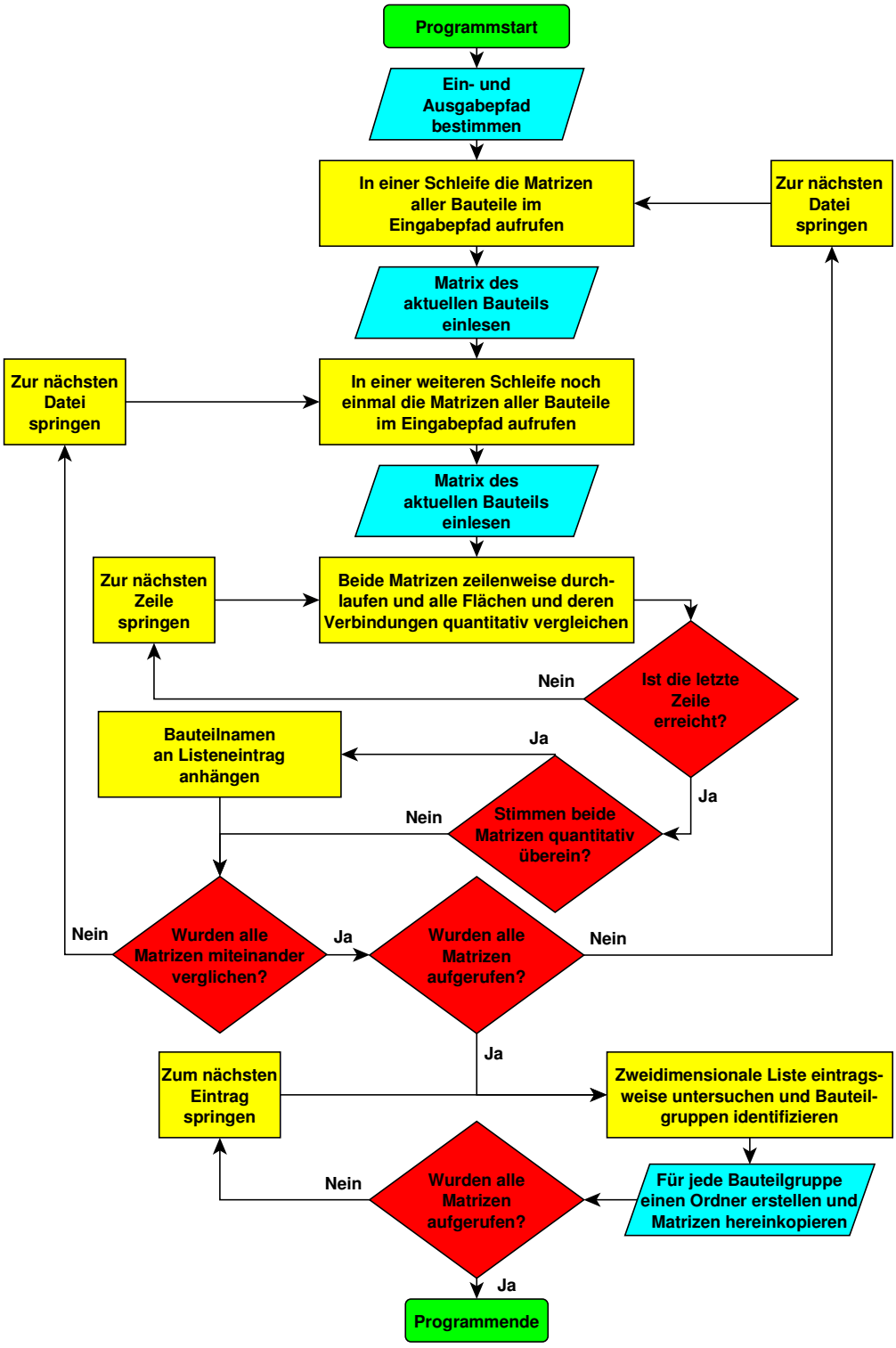

Abbildung B.23: vrml_cluster_quan.py

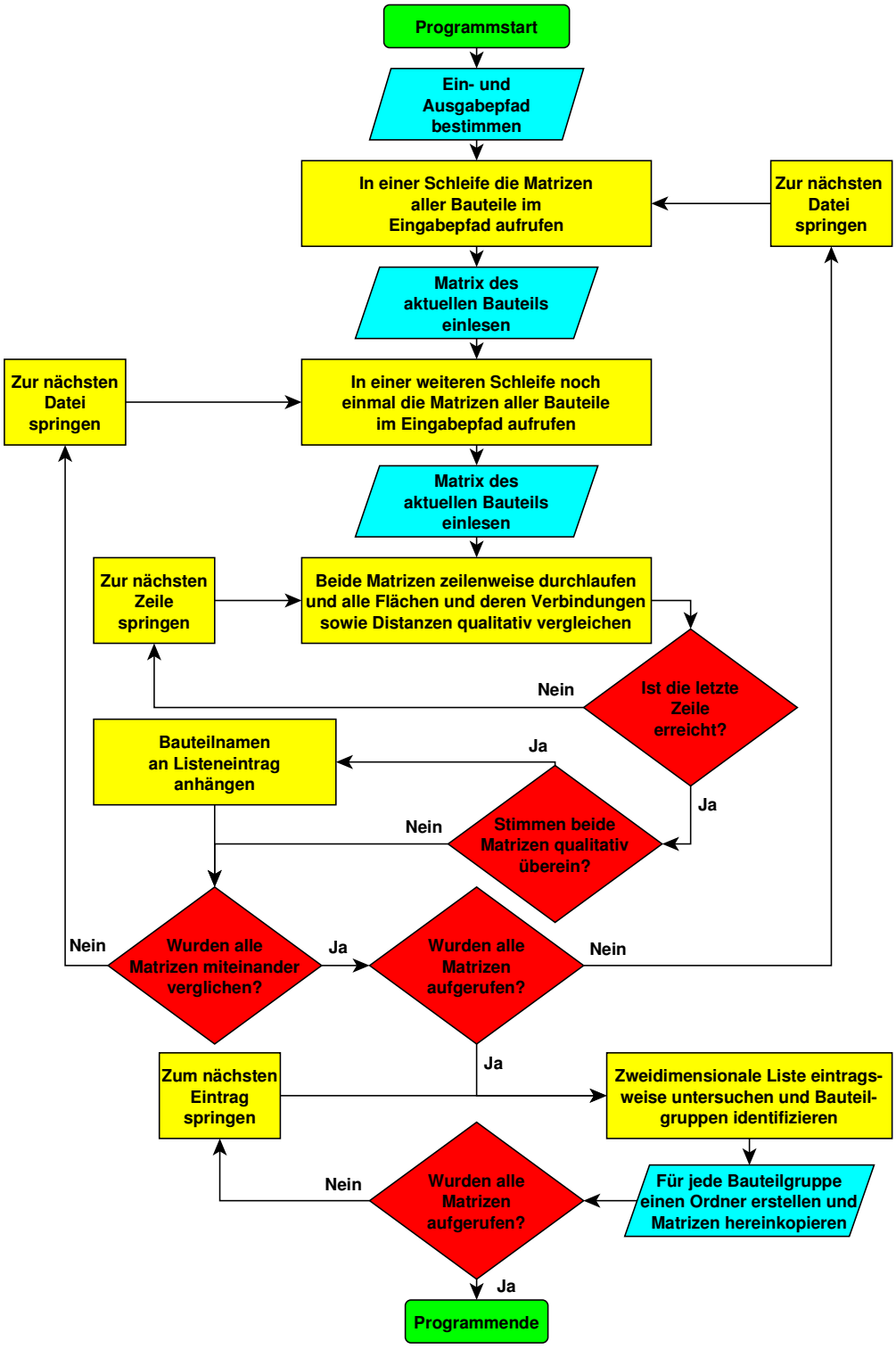

Abbildung B.24: vrml_cluster_qual.py

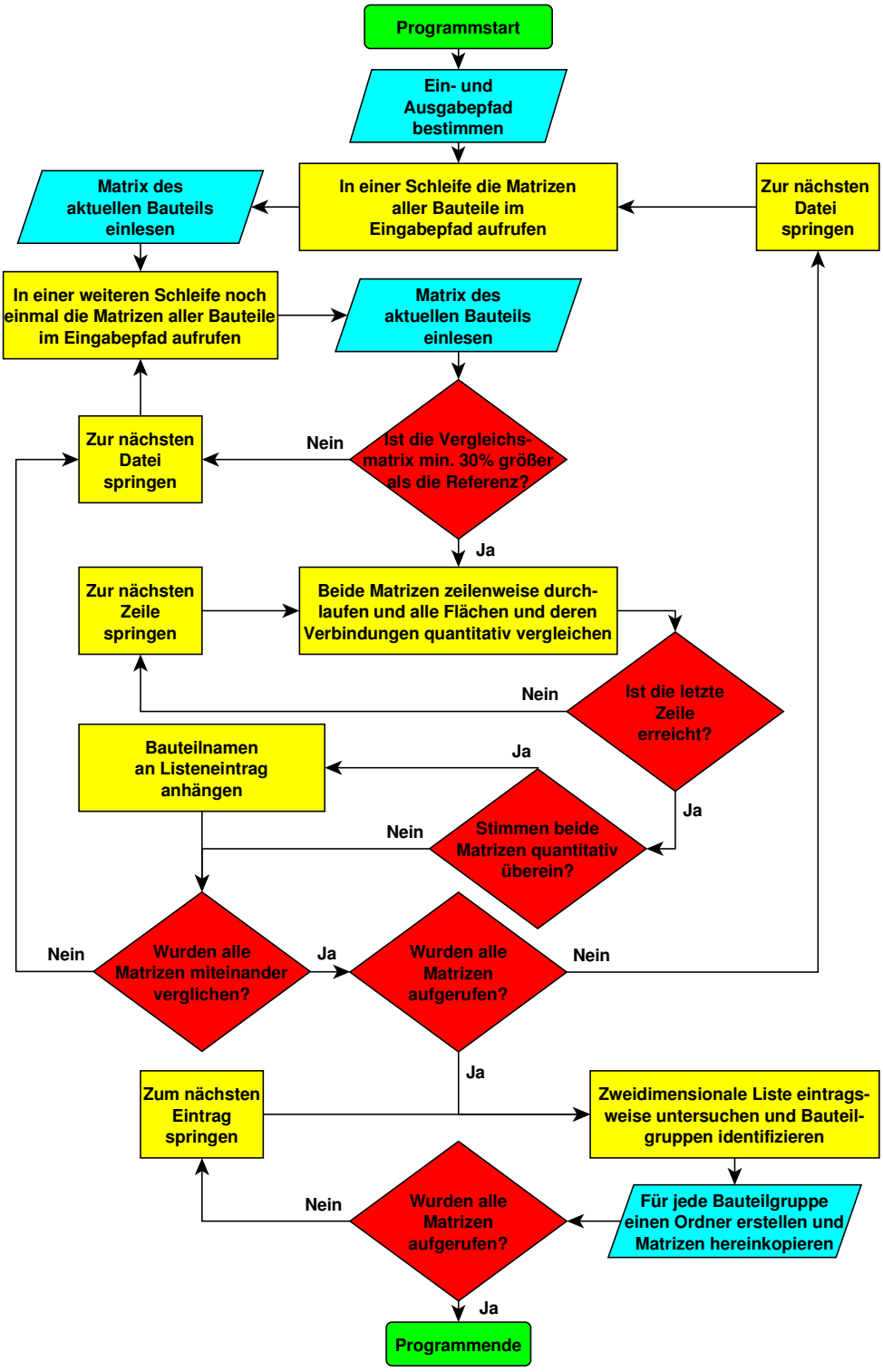

Abbildung B.25: subgraph_vrml_cluster_quan.py

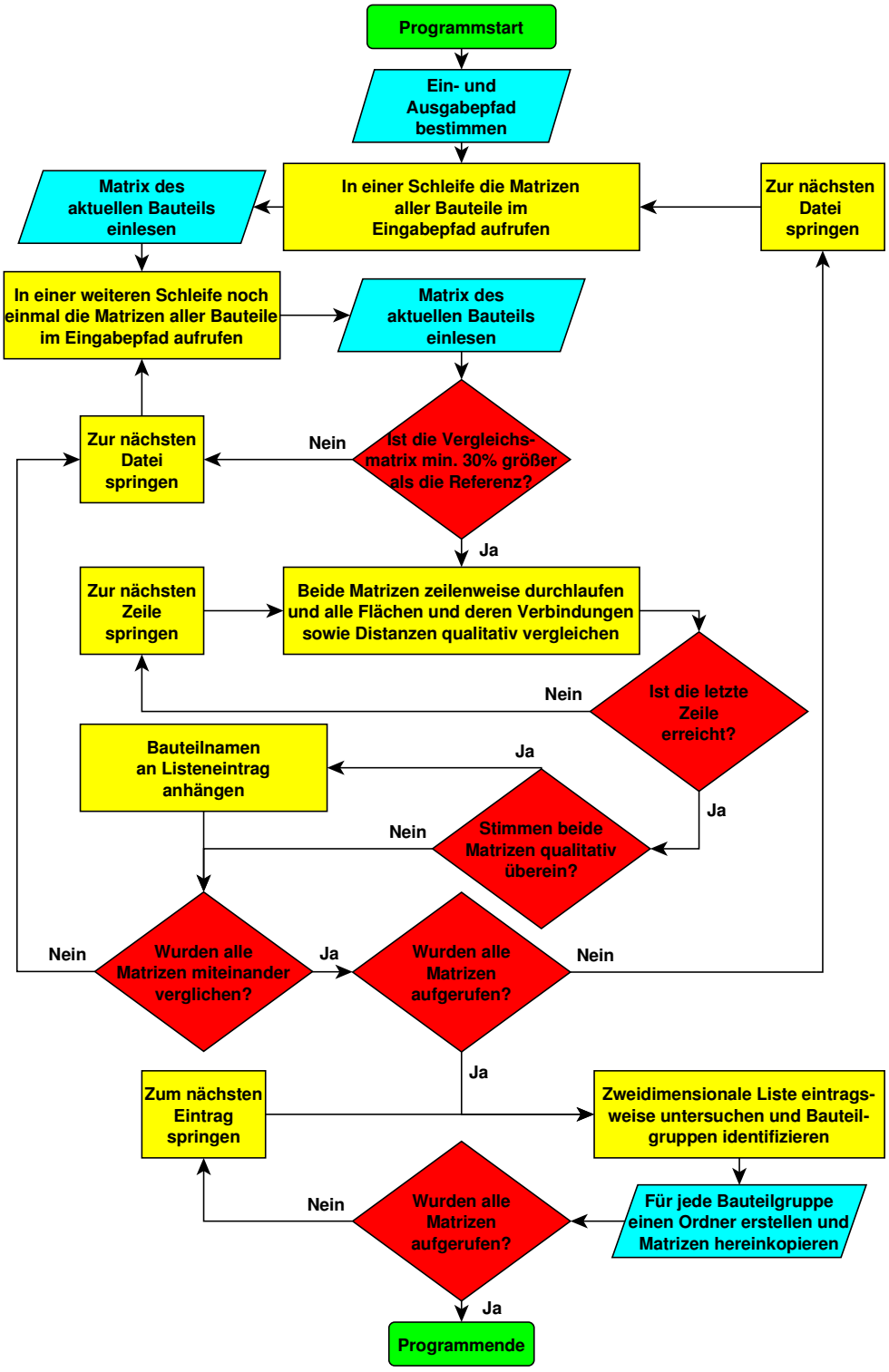

Abbildung B.26: subgraph_vrml_cluster_qual.py

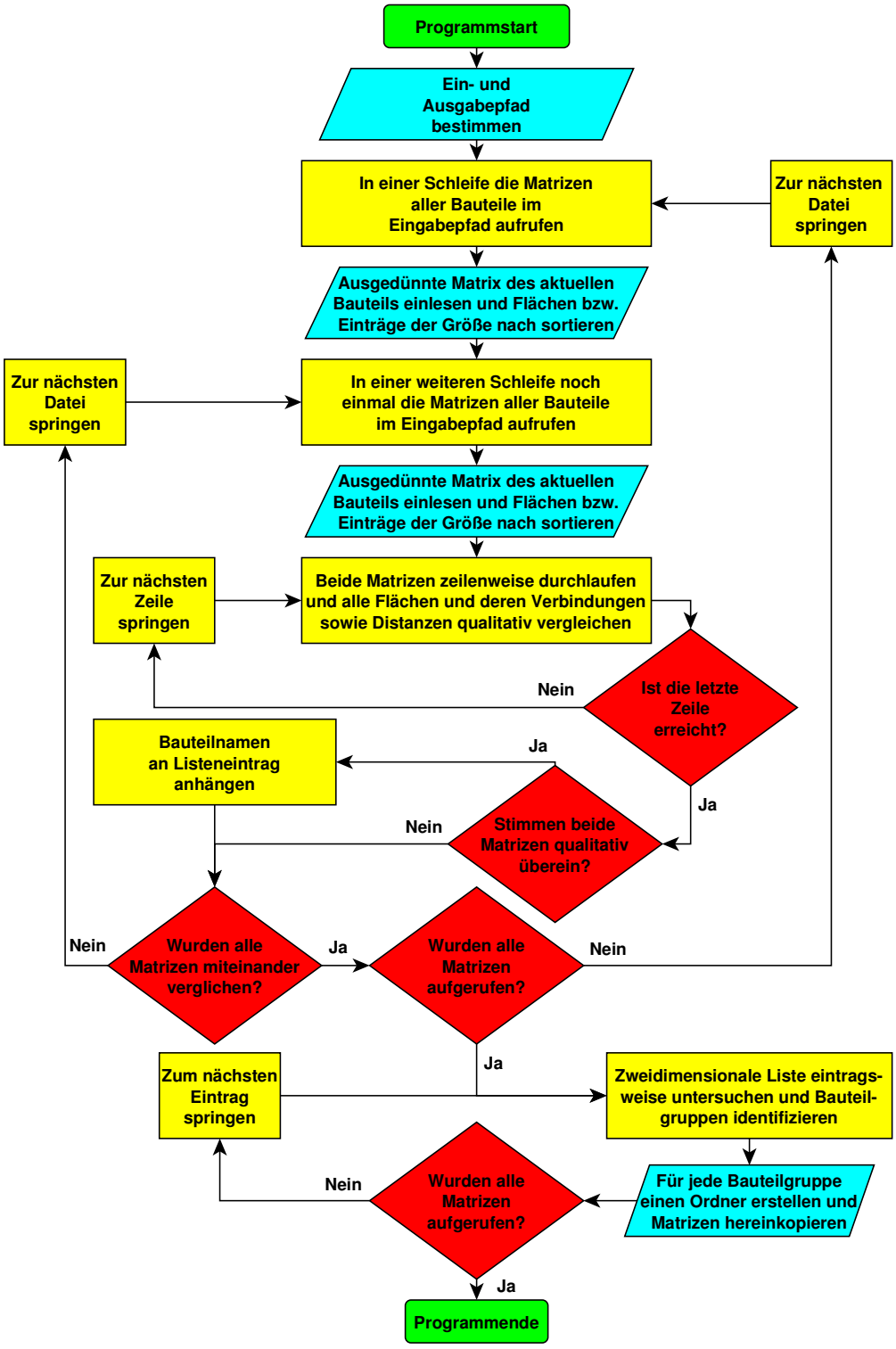

Abbildung B.27: simple_vrml_cluster.py

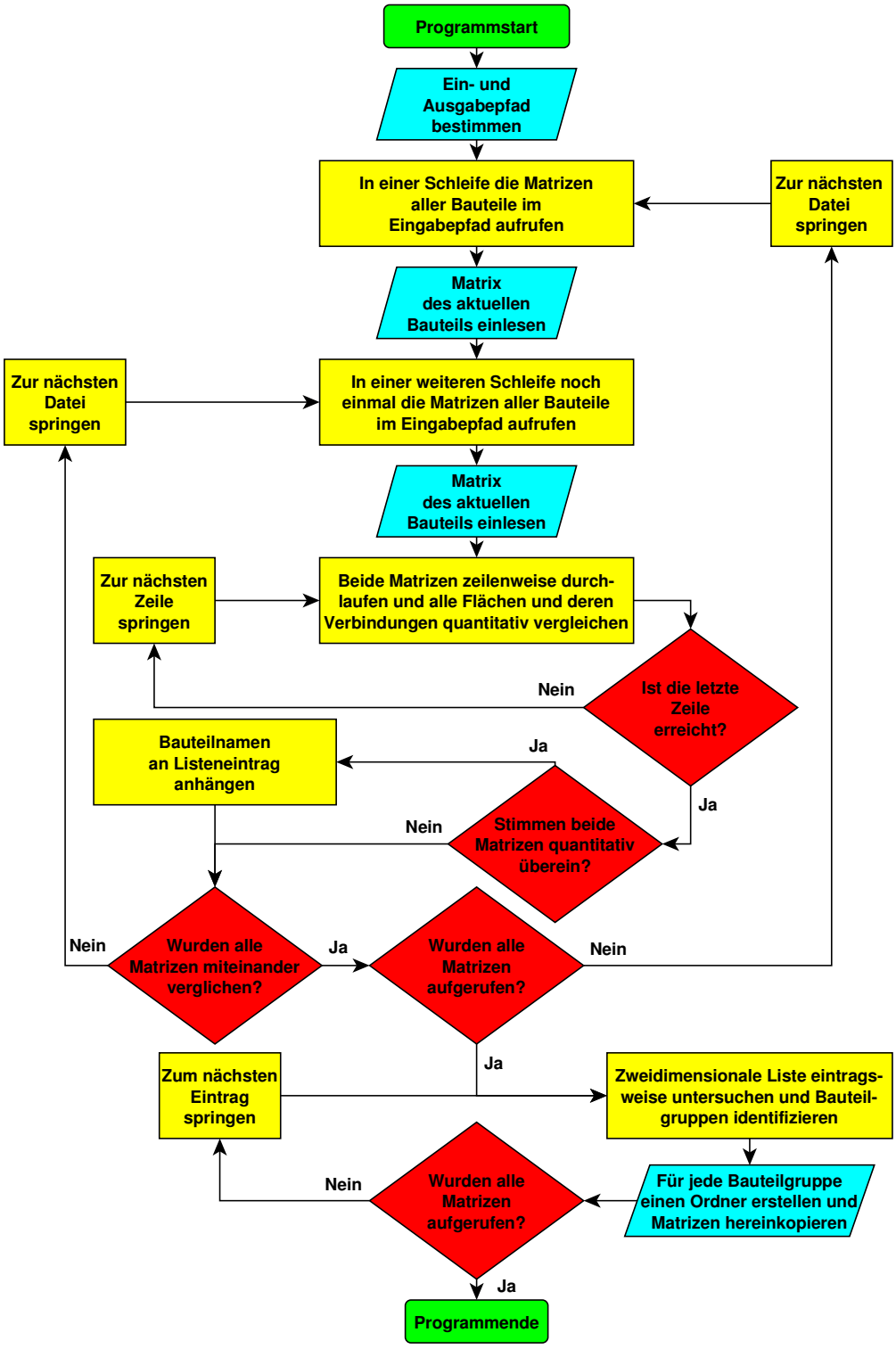

Abbildung B.28: iges_cluster.py

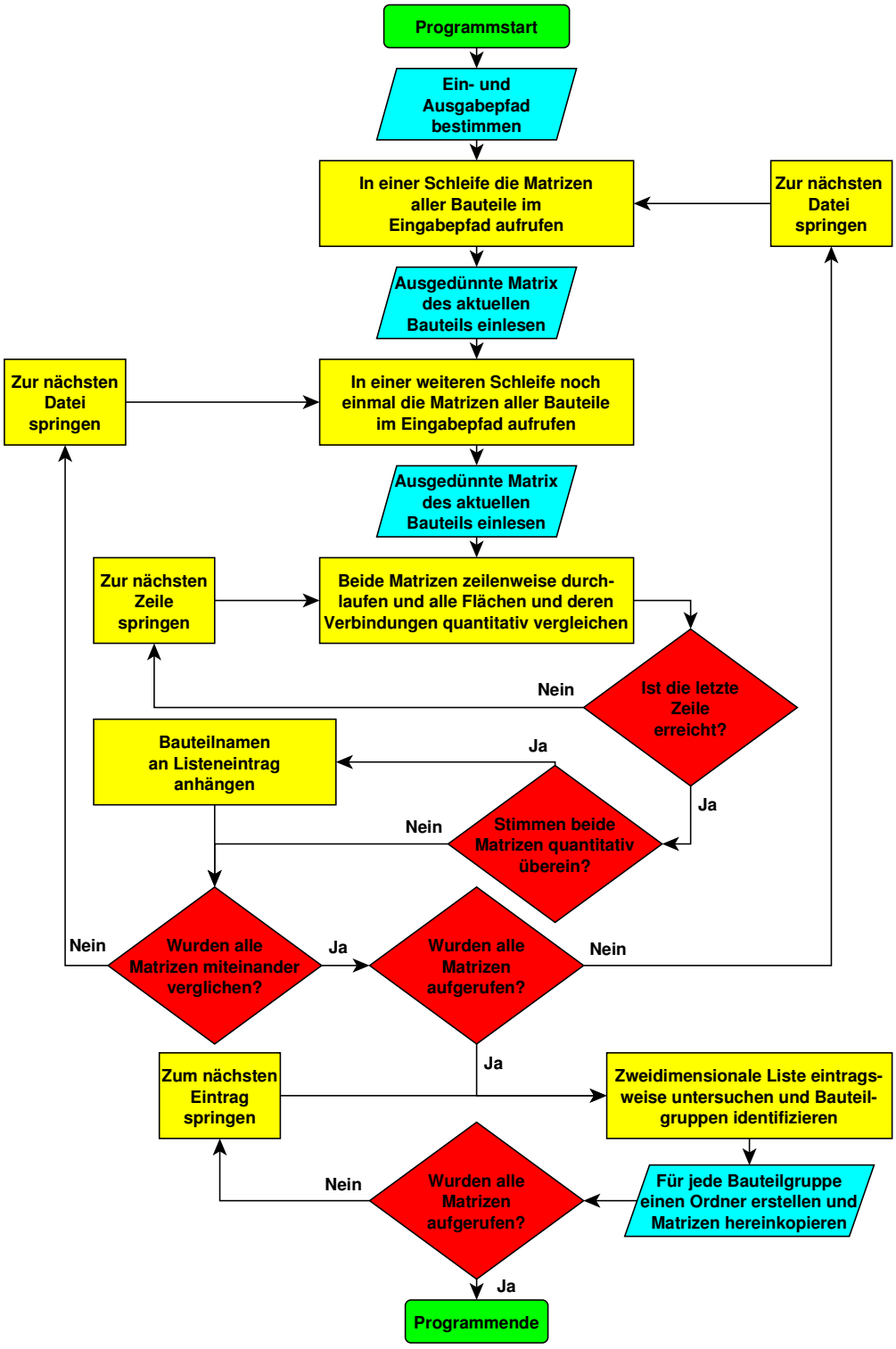

Abbildung B.29: simple_iges_cluster.py

Literaturverzeichnis

[*Dha-14]:
Preeti Dhandam
Visualisation of Voxel files aided by binvox in CATIA
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2015

[*Don-14]:
Hui Dong
Property Extraction of Voxelized CAD-models
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2014

[*Mia-14]:
Yu Miao
Generate 3D CATIA Models by 2D Photos
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2014

[*Özm-15]:
Engin Özmen
Softwaremethoden zur Suche und zur Klassifizierung von Bauteilen
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2015

[*Pen-13]:
Hu Peng
Eine Untersuchung der gegebenen Rekonstruktionsmethode für CAD-Modelle anhand von makroprogrammierten Bauteilvarianten
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2013

[*Roj-14]:
Robin Roj
A comparison of three design tree based search algorithms for the detection of engineering parts constructed with CATIA V5 in large databases
Journal of Computational Design and Engineering (2014)1(3):161-172
Elsevier

[*RoW-13]:
Robin Roj, Hans-Bernhard Woyand
An Automated Method for the Reconstruction of Engineering Parts with Python and CATIA V5 Based on Two-Dimensional Input
5th IEEE International Symposium on Logistics and Industrial Informatics, Wildau (2013):61-66
IEEE

[*RoW-14]:
Robin Roj, Hans-Bernhard Woyand
Three approaches for the detection of cad-part attributes as a preparation for an automatic classification
18th International Conference on Intelligent Engineering Systems, Tihany (2014):49-54
IEEE

[*RoW-15]:
Robin Roj, Hans-Bernhard Woyand
An Examination of Engineering Parts in Large CAD-Databases in Order to Create Adjacency Matrices and Build Clusters
19th International Conference on Intelligent Engineering Systems, Bratislava (2015):97-102
IEEE

[*Sch-13]:
Rudolf Scheider
Automatisierte Klassifizierung von CAD-Daten auf der Basis des STL-Datenformats
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2013

[*She-14]:
Parthkumar Sheth
Identifying Attributes of STL-Parts for Classification
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2014

[*Wag-15]:
Melanie Maryam Wagner
Softwaremethoden zur Suche und zur Klassifizierung von Bauteilen
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2015

[*Vad-14]:
Shashank Vadlamani
A performance validation and improvement of three different search engines for CAD models using Python and CATIA V5
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2014

[*Zhu-13]:
Lina Zhu
STL-Data Analysis for CAD-Feature Extraction
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2013

[*Zhu-14]:
Lina Zhu
Technical comparison of reconstructed engineering parts with their original CAT-Parts
Bergische Universität Wuppertal - Fachgebiet für Maschinenbauinformatik
Wuppertal 2014

[ADEK-05]:
Volker Arnold, Hendrick Dettmering, Torsten Engel, Andreas Karcher
Product Lifecycle Management beherrschen
Ein Anwenderhandbuch für den Mittelstand
Springer-Verlag
Berlin Heidelberg 2005, 2011
Seite 123

[AlDV-10]:
M. Alemanni, F. Destefanis, E. Vezzetti
Model-based definition design in the product lifecycle management scenario
The International Journal of Advanced Manufacturing Technology (2011)52:1-14
Springer

[AmNM-97]:
Andrea L. Ames, David R. Nadeau, John L. Moreland
VRML 2.0
Sourcebook
John Wiley & Sons
New York 1997

[AnDV-07]:
Tarik Filali Ansary, Mohamed Daoudi, Jean-Philippe Vandeborre
A Bayesian 3-D Search Engine Using Adaptive Views Clustering
IEEE Transactions on Multimedia (2007)9(1):78-88
IEEE

[AnTr-00]:
Reiner Anderl
STEP
Standard for the Exchange of Product Model Data
Eine Einführung in die Entwicklung, Implementierung, und industrielle Nutzung der Normenreihe ISO 10303 (STEP)
B. G. Teubner
Stuttgart, Leipzig 2000

[ATRB-95]:
A.P.Ashbrook, N.A.Thacker, P.I.Rockett and C.I.Brown
Robust Recognition of Scaled Shapes Using Pairwise Geometric Histograms.
6th British Conference on Machine Vision, Surrey (1995):503-512
BMVA Press

[BaNM-07]:
Bojan Babic, Nenad Nesic, Zoran Miljkovic
A review of automated feature recognition with rule-based pattern recognition
Computers in Industry (2008)59:321-337
Elsevier

[Beau-15]:
Beautiful Soup [Internet]
New York: Beautiful Soup: We called him Tortoise because he taught us.; c2015
www.crummy.com/software/BeautifulSoup/
Stand: 12.08.2015

[BeBe-85]:
Rolf Bernhardt, Werner Bernhardt
Nummerungssysteme
Grundbegriffe und Einführung, Systemvergleiche, praktische Anwendung, speziell bei EDV-Systemen
expert
Ehningen bei Böblingen 1985

[Bech-09]:
Margit Becher
XML
DTD, XML-Schema, XPath, XQuery, XSLT, XSL-FO, SAX, DOM
W3L
Herdecke, Witten 2009

[BGCH-07]:
David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, Shilpa Dani
An engineering design knowledge reuse methodology using process modelling
Research in Engineering Design (2007)18:37-48
Springer

[Binv-15]:
Binvox [Internet]
Princeton: binvox 3D mesh voxelizer, keywords: voxelization, voxelisation, 3D model - Mozi; c2015
www.cs.princeton.edu/~min/binvox/
Stand: 10.08.2015

[BKSS-05]:
Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, Dejan V. Vranić
Feature-Based Similarity Search in 3D Object Databases
ACM Computer Surveys (2005)37(4):345-387
ACM

[Boic-03]:
Mihai Boicu
Modeling and Learning with Incomplete Knowledge
ProQuest Information and Learning Company
Ann Arbor 2003

[Burc-01]:
Carsten Burchardt
Ein erweitertes Konzept für die integrierte Produktentwicklung
Otto-Guericke-Universität Magdeburg - Lehrstuhl für Maschinenbauinformatik
Magdeburg 2001

[Cade-15]:
CADENAS [Internet]
Augsburg: Maßgeschneiderte Softwarelösungen für Hersteller von Komponenten & Industrie - CADENAS (de); c2015
www.cadenas.de/produkte
Stand: 31.07.2015

[Carl-10]:
Carl Zeiss AG Solution Competence Center PLM [Internet]
Jena: Vorgaben für die Erstellung von Pro/ENGINEER-Modellen und -Zeichnungen; c2010
www.zeiss.de/C125679B0029303C/EmbedTitelIntern/pro_engineer_arbeitsrichtlinie/$File/Zeiss-ProENGINEER-Vorgaben.pdf
Stand: 19.07.2013

[ChGa-96]:
Chi-Cheng Peter Chu, Rajit Gadh
Feature-based approach for set-up minimization of process design from product design
Computer-Aided Design (1996)26(5):321-332
Elsevier

[ChLL-10]:
Chee Kai Chua, Kah Fai Leong, Chu Sing Lim
Rapid Prototyping
Principles and Applications
World Scientific Publishing
Singapur 2010

[Clar-05]:
Constanze Clarke
Automotive Production Systems and Standardisation
From Ford to the Case of Mercedes-Benz
Physica
Heidelberg 2005

[CoLa-07]:
Dave Cooper, Gianfranco LaRocca
Knowledge-Based Techniques for Developing Engineering Applications in the 21st Century
7th Integration and Operations Conference, Belfast (2007):1-22
American Institute of Aeronautics and Astronautics

[CuGo-12]:
Erik Cuevas, Mauricio González
Multi-circle detection on images inspired by collective animal behavior
Applied Intelligence (2012)39(1):101-120
Springer

[CYCH-08]:
Sean Cochrane, Robert Young, Keith Case, Jennifer Harding, James Gao, Shilpa Dani, David Baxter
Knowledge reuse in manufacturability analysis
Robotics and Computer-Integrated Manufacturing (2008)24:508-513
Elsevier

[Dass-15]:
Dassault Systèmes [Internet]
Vélizy-Villacoublay: EXALEAD OnePart - Dassault Systèmes; c2015
www.exalead-onepart.com
Stand: 31.07.2015

[Deng-07]:
Qianwang Deng
A Contribution to the Integration of Knowledge Management into Product Development
Otto-Guericke-Universität Magdeburg - Fakultät für Maschinenbau
Magdeburg 2007

[DiXK-11]:
Tobias Dipper, Xun Xu, Peter Klemm
Defining, recognizing and representing feature interactions in a feature-based data model
Robotics and Computer-Integrated Manufacturing (2011)27:101-114
Elsevier

[DiYL-14]:
Bo Ding, Xiao-yang Yu, Li Liu
3D CAD Model Representation and Retrieval Based on Hierarchical Graph
Journal of Software (2014)9(10):2499-2506
Academy Publisher

[Ecke-09]:
Claudia Eckert
IT-Sicherheit
Konzepte - Verfahren - Protokolle
Oldenbourg Wissenschaftsverlag
München 2009
Seite 32

[Ecla-15]:
eCl@ss [Internet]
Köln: eCl@ss Home; c2015
www.eclass.de
Stand: 31.07.2015

[ElTo-08]:
S.W.G. van der Elst, M.J.L. van Tooren
Application of a Knowledge Engineering Process to Support Engineering Design Application Development
Collaborative Product and Service Life Cycle Management for a Sustainable World
Advanced Concurrent Engineering (2008):417-431
Springer

[Erlh-11]:
Sebastian Erlhofer
Suchmaschinen-Optimierung
Das umfassende Handbuch
Galileo Press
Bonn 2011

[Eski-05]:
Mustafa Taner Eskil
Distributed Routine Design over the Internet with Cooperating MDM Agents
ProQuest Information and Learning Company
Ann Arbor 2005

[EsSa-00]:
Martin Ester, Jörg Sander
Knowledge Discovery in Databases
Techniken und Anwendungen
Springer
Berlin, Heidelberg 2000
Seite 1

[FaPS-96]:
Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth
From Datamining to Knowledge Discovery in Databases
Artifical Intelligence Magazine (1996)17(3):37-54
Association for the Advancement of Artificial Intelligence

[Fibi-02]:
Iris Fibinger
SVG - Scalable Vector Graphics
Praxiswegweiser und Referenz für den neuen Vektorgrafikstandard
Markt+Technik
München 2002

[FrPM-92]:
William J. Frawley, Gregory Piatestsky-Shapiro, Christopher J. Matheus
Knowledge Discovery in Databases: An Overview
Artificial Intelligence Magazine (1992)13(3):57-70
Association for the Advancement of Artificial Intelligence

[Gaag-10]:
Andreas Gaag
Entwicklung einer Ontologie zur funktionsorientierten Lösungssuche in der Produktentwicklung
Technische Universität München - Fakultät für Maschinenwesen der
München 2010

[GaRö-94]:
Roland Gabriel, Heinz-Peter Röhrs
Datenbanksysteme
Konzeptionelle Datenmodellierung und Datenbankarchitekturen
Springer
Berlin, Heidelberg 1994

[GeRe-15]:
Christophe Geuzaine, Jean-François Remacle
Gmsh Reference Manual
The documentation for Gmsh 2.10
A finite element mesh generator with built-in pre- and post-processing facilities
GNU General Public License
Liège 2015

[Goek-06]:
Matthias Goeken
Entwicklung von Data-Warehouse-Systemen
Anforderungsmanagement, Modellierung, Implementierung
Deutscher Universitäts-Verlag
Wiesbaden 2006
Seite 188

[Grei-06]:
Horst Greifeneder
Erfolgreiches Suchmaschinen-Marketing
Wie Sie bei Google, Yahoo, MSN & Co. ganz nach oben kommen
Gabler
Wiesbaden 2010

[GrLW-02]:
Hans Grabowski, Ralf Lossak, Jörg Weißkopf
Datenmanagement in der Produktentwicklung - Automatische Klassifikation von Produktdaten aus 3D-CAD-Systemen, PDM- und ERP-Systemen, XML- und Office-Dokumenten,...
Carl Hanser
München, Wien 2002
Seite 15

[Grub-93]:
Thomas R. Gruber
A translation approach to portable ontology specifications
Knowledge Acquisition (1993)5:199-220
Academic Press

[HaWB-00]:
D. Alfred Hancq, Andrew J. Walters, Jack L. Beuth
Development of an Object-Oriented Fatigue Tool
Engineering with Computers (2000)16:131-144
Springer

[HCTT-10]:
Y. Y. Hsu, W. C. Chen, P. H. Tai, Y. T. Tsai
A Knowledge-Based Engineering System for Assembly Sequence Planning
36th International MATADOR Conference, Manchester (2010):123-126
Springer

[HiHa-99]:
Robert J. Hilderman, Howard J. Hamilton
Knowledge Discovery and Interestingness Measures: A Survey
University of Regina - Department of Computer Science
Regina 1999

[Hill-06]:
Brad Hill
Google Suche für Dummies
Wiley-VCH
Weinheim 2006

[HKSV-01]:
Martin Heczko, Daniel Keim, Dietmar Saupe, Dejan V. Vranić
Verfahren zur Ähnlichkeitssuche auf 3D-Objekten
Datenbanksysteme in Büro, Technik und Wissenschaft (2001)9:384-401
Springer

[HoLR-05]:
Suyu Hou, Kuiyang Lou, Karthik Ramani
SVM-based Semantic Clustering and Retrieval of a 3D Model Database
Computer-Aided Design and Applications (2005)2:155-164
Taylor & Francis

[Hren-09]:
Gorazd Hren
Web-based environment for mechanism simulation integrated with CAD system
Engineering with Computers (2010)26:137-148
Springer

[IJLK-04]:
Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyaranaman, Karthik Ramani
Three-dimensional shape searching: state-of-the-art review and future trends
Computer-Aided Design (2005)37:509-530
Elsevier

[JaHo-10]:
Woo-Seok Jang, Yo-Sung Ho
3-D Object Reconstruction from Multiple 2-D Images
3D Research (2010)1(2):1-5
Springer

[JaKR-09]:
Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Karthik Ramani
Shape-based clustering for 3D CAD objects: A comparative study of effectiveness
Computer-Aided Design (2009)41:999-1007
Elsevier

[Jian-05]:
Li Jiang
A Framework for the Requirements Engineering Process Development
University of Calgary - Department of Electrical and Computer Engineering
Calgary 2005
Seite 40

[KeEi-06]:
Alfons Kemper, André Eickler
Datenbanksysteme
Eine Einführung
Oldenbourg Wissenschaftsverlag
München 2006
Seite 496

[KiKi-10]:
Yun Seon Kim, Kyoung-Yun Kim
DCR-based causal design knowledge evaluation method and system for future CAD applications
Computer-Aided Design (2012)44:947-960
Elsevier

[KiKi-11]:
Kyoung-Yun Kim, Yun Seon Kim
Causal design knowledge: Alternative representation method for product development knowledge management
Computer-Aided Design (2011)43:1137-1153
Elsevier

[KöBT-11]:
Gülser Köksal, Inci Batmaz, Murat Caner Testik
A review of data mining applications for quality improvement in manufacturing industry
Expert Systems with Applications (2011)38:13448-13467
Elsevier

[Korn-07]:
Patrick Kornprobst
CATIA V5 - Volumenmodellierung Grundlagen und Methodik in über 100 Konstruktionsbeispielen
Carl Hanser
München 2007
Seite 26

[KoWe-10]:
Sabine Roth-Koch, Engelbert Westkaemper
The implementation of a sketch-based virtual product development
Production Engineering Research and Development (2010)4(2-3):175-183
Springer

[KPNK-03]:
Marcel Körtgen, Gil-Joo Park, Marcin Novotni, Reinhard Klein
3D Shape Matching with 3D Shape Contexts
7th Central European Seminar on Computer Graphics, Budmerice (2003)
Technische Universität Wien

[KuBM-06]:
J. Kulon, P. Broomhead, D. J. Mynors
Applying knowledge-based engineering to traditional manufacturing design
International Journal of Advanced Manufacturing Technology (2006)30:945-951
Springer

[Kurb-92]:
Karl Kurbel
Entwicklung und Einsatz von Expertensystemen
Eine Anwendungsorientierte Einführung in wissensbasierte Systeme
Springer
Berlin, Heidelberg 1992
Seite 18

[LeFa-12]:
Yong Tsui Lee, Fen Fang
A new hybrid method for 3D object recovery from 2D drawings and its validation against the cubic corner method and the optimisation-based method
Computer-Aided Design (2012)44:1090-1102
Elsevier

[LeKi-96]:
Jae Yeol Lee, Kwangsoo Kim
Geometric reasoning for knowledge-based parametric design using graph representation
Computer-Aided Design (1996)28(10):831-841
Elsevier

[LiDH-11]:
Zou Linghao, Guo Dongming, Gao Hang
A method to analyze the difference of 3-D CAD model files based on feature extraction
Journal of Mechanical Science and Technology (2011)25(4):971-976
KSME, Springer

[Lino-15]:
Lino [Internet]
Mainz: simus classmate - Klassifikation, Struktuierung [sic!], Kalkulation von Daten - Lino GmbH; c2015
www.lino.de/simus-classmate.html
Stand: 31.07.2015

[Lupa-09]:
Norman Lupa
Einsatz wissensbasierter Features für die automatische Konfiguration von Produktkomponenten
Cuvillier
Göttingen 2009

[MaHW-09]:
Lujie Ma, Zhengdong Huang, Qingsong Wu
Extracting common design patterns from a set of solid models
Computer-Aided Design (2009)41:952-970
Elsevier

[Meie-04]:
Andreas Meier
Relationale Datenbanken
Leitfaden für die Praxis
Springer
Berlin, Heidelberg 2004

[Meyw-07]:
Martin Meywerk
CAE-Methoden in der Fahrzeugtechnik
Springer
Berlin, Heidelberg 2007
Seite 8

[Milt-08]:
Nick .R. Milton
Knowledge Technologies
Polimetrica
Mailand 2008
Seite 13

[Minp-04]:
Patrick Min
A 3D Model Search Engine
Princeton University - Department of Computer Science
Princeton 2004

[MiŠZ-13]:
Peter Michalik, Ján Štofa, Iveta Zolotová
Testing the Properties of K-means Algorithm for Data Mining Applications
5th IEEE International Symposium on Logistics and Industrial Informatics, Wildau (2013):99-102
IEEE

[MoKS-06]:
Seung Ki Moon, Soundar R.T. Kumara, Timothy W. Simpson
Data Mining and Fuzzy Clustering to Support Product Family Design
International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Philadelphia (2006):317-325
ASME

[MoSK-10]:
Seung Ki Moon, Timothy W. Simpson, Soundar R. T. Kumara
A methodology for knowledge discovery to support product family design
Annals of Operations Research (2010)174:201-218
Springer, LLC

[NaLa-91]:
Vijay Nagasamy, Noshir A. Lagrana
Reconstruction of Three-Dimensional Obejcts Using a Knowledge-Based Environment
Engineering with Computers (1991)7:23-35
Springer

[OFCD-01]:
Robert Osada, Thomas Funkhouser, Bernard Chazelle, David Dobkin
Matching 3D Models with Shape Distributions
International Conference on Shape Modeling & Applications, Genua (2001):154-167
IEEE

[Onst-14]:
Scott Onstott
Autodesk AutoCAD 2015 und AutoCAD LT 2015
Das offizielle Trainingsbuch
Wiley-VCH
Weinheim 2014

[Orlo-06]:
Michael A. Orloff
Grundlagen der klassischen TRIZ
Ein praktisches Lehrbuch des erfinderischen Denkens für Ingenieure
Springer
Berlin, Heidelberg 2006

[Park-03]:
Sang C. Park
Knowledge capturing methodology in process-planning
Computer-Aided Design (2003)35:1109-1117
Elsevier

[PBFG-05]:
Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen, Karl-H. Grote
Konstruktionslehre - Grundlagen erfolgreicher Produktentwicklung
Methoden und Anwendung
Springer
Berlin, Heidelberg 2005
Seite 541

[Pepe-06]:
Werner Pepels
Produktmanagement
Produktinnovation Markenpolitik Programmplanung Prozessorganisation
Oldenbourg Wissenschaftsverlag
München 2006
Seite 16

[PeRe-01]:
Mitchell Peabody, William C. Regli
Clustering Techniques for Databases of CAD Models
Drexel University
Philadelphia 2001

[Pete-05]:
Helge Petersohn
Data Mining - Verfahren, Prozesse, Anwendungsarchitektur
Oldenbourg Wissenschaftsverlag
München 2005
Seite 4

[PeWe-99]:
Fu Peifu, Jing Weifeng
An Intelligent CAD System for Cold-forging Parts in Automobile Industry
The IEEE International Vehicle Electronics Conference, Changchun (1999):410-413
IEEE

[Pocs-00]:
Zsolt Pocsai
Ontologiebasiertes Wissensmanagement für die Produktentwicklung
Shaker
Aachen 2000

[Poko-06]:
Jerzy Pokojski
Knowledge Based Engineering and Intelligent Personal Assistant Context in Distributed Design
Springer
Berlin, Heidelberg 2006

[Prin-15]:
Princeton Shape Retrieval and Analysis Group [Internet]
Princeton: Princeton Shape Benchmark; c2003
shape.cs.princeton.edu/benchmark/
Stand: 05.08.2015

[PuCS-07]:
Dante Pugliese, Giorgio Colombo, Maurizio Saturno Spurio
About the integration between KBE and PLM
Advances in Lifecycle Engineering for Sustainable Manufacturing Business (2007):131-136
Springer

[QuOw-12]:
William Roshan Quadros, Steven J. Owen
Defeaturing CAD Models Using a Geometry-Based Size Field and Facet-Based Reduction Operators
Engineering with Computers (2012)28(3):211-224
Springer

[RaHB-07]:
Svetan M. Ratchev, Hitendra Hirani, Maurice Bonney
Knowledge based formation of re-configurable assembly cells
Intelligent Manufacturing (2007)18:401-409
Springer

[RoVe-06]:
C. Romero, S. Ventura
Educational data mining: A survey from 1995 to 2005
Expert Systems with Applications (2007)33:135-146
Elsevier

[Rude-98]:
Stefan Rude
Wissensbasiertes Konstruieren
Shaker
Aachen 1998

[SAGS-13]:
A. Shtof, A. Agathos, Y. Gingold, A. Shamir, D. Cohen-Or
Geosemantic Snapping for Sketch-Based Modeling
Computer Graphics Forum (2013)32(2):245-253
Blackwell Publishing

[SAKR-05]:
Bernd Schäppi, Mogens M. Andreasen, Manfred Kirchgeorg, Franz-Josef Radermacher
Handbuch Produktentwicklung
Carl Hanser
München, Wien 2005
Seite 615

[Sand-03]:
Marcus Sandberg
Knowledge Based Engineering - In Product Development
Technical Report
Luleå University of Technology - Department of Applied Physics and Mechanical Engineering, Division of Computer Aided Design
Luleå 2003

[Sard-06]:
MD Baniamin Sarder
The Development of a Design Ontology for Products and Processes
ProQuest Information and Learning
Ann Arbor 2006

[ScGe-96]:
Michael Schumann, Detlef Gerst [Internet]
Hamburg: Innovative Arbeitspolitik - Ein Fallbeispiel
Gruppenarbeit in der Mercedes-Benz AG
SOFI-Mitteilungen Nr.24/1996; c1996
www.econbiz.de/archiv1/2008/17259_innovative_arbeitspolitik.pdf
Stand: 31.07.2015

[Sche-15]:
Günter Scheuermann
Inventor 2016
Grundlagen und Methodik in zahlreichen Konstruktionsbeispielen
Carl Hanser
München 2015

[Schn-16]:
Peter Schnauffer
CATIA Handbuch
Konstruieren mit CATIA V5 & V6
Springer
Berlin, Heidelberg 2016

[Schu-01]:
Dirk Schulze
Grundlagen der wissensbasierten Konstruktion von Modellen betrieblicher Systeme
Berichte aus der Wirtschaftsinformatik
Shaker
Aachen 2001
Seite 64-85

[ScRH-01]:
Reinhard Schütte, Thomas Rotthowe, Roland Holten
Data Warehouse Managementhandbuch
Konzepte, Software, Erfahrungen
Springer
Berlin, Heidelberg 2001
Seite 29

[Sega-07]:
Toby Segaran
Programming Collective Engineering
O'Reilly
Sebastopol 2007

[ShMä-95]:
Jami J. Shah, Martti Mäntylä
Parametrics and Feature-Based CAD/CAM
Concepts, Techniques, Applications
John Wiley & Sons
New York 1995

[ShSi-12]:
Shelza, Balwinder Singh
A Novel Approach for Comparison of Clustering Algorithms on CAD Images
International Journal of Computer Science and Information Technology & Security (2012)2(4):846-851
IRACST

[Siem-15]:
Siemens PLM [Internet]
Köln: Geolus Search: Siemens PLM Software - Germany; c2015
www.plm.automation.siemens.com/de_de/products/open/geolus/index.shtml
Stand: 31.07.2015

[Siem-16]:
Siemens JT Open [Internet]
Plano: JT Open: Siemens PLM Software; c2016
http://www.plm.automation.siemens.com/en_us/products/open/jtopen/
Stand: 04.04.2016

[Simp-15]:
Simple CV [Internet]
Ann Arbor: SimpleCV; c2015
simplecv.org/
Stand: 12.08.2015

[Simu-14]:
Simuform [Internet]
Dortmund: Geometric Search Software | Simuform Similia; c2014
www.simuform.com/geometric-search-software.html
Stand: 31.07.2015

[Stro-06]:
Oliver Strohmeier
Integration von Wissensmodulen in den virtuellen Produktentwicklungsprozess
Shaker
Aachen 2006

[Stuc-11]:
Heiner Stuckenschmidt
Informatik im Fokus
Ontologien
Konzepte, Technologien und Anwendungen
Springer
Berlin, Heidelberg 2011
Seite 5

[SuES-06]:
York Sure, Marc Ehrig, Rudi Studer
Automatische Wissensintegration mit Ontologien
Universität Karlsruhe - Institut für angewandte Informatik und formale Beschreibungsverfahren
Karlsruhe 2006

[Summ-04]:
Joshua D. Summers
Development of a Domain and Solver Independent Method for Mechanical Engineering Embodiment Design
ProQuest Information and Learning
Ann Arbor 2004

[TaWY-11]:
Ryota Takeuchi, Taichi Watanabe, Soji Yamakawa
Sketch-based Solid Prototype Modeling System with Dual Data Structure of Point-set Surfaces and Voxels
International Journal of CAD/CAM (2011)11(1):18-26
Hokkaido University

[TayG-03]:
Francis E. H. Tay, J. Gu
A methodology for evolutionary product design
Engineering with Computers (2003)19:160-173
Springer

[TiMü-03]:
Thorsten Tietjen, Dieter H. Müller
FMEA-Praxis
Das Komplettpaket für Training und Anwendung
Carl Hanser
München, Wien 2003

[Ulic-05]:
Eberhard Ulich
Arbeitspsychologie
vdf Hochschulverlag, Schäffer-Poeschel
Stuttgart 2005

[USPD-96]:
United States Product Data Association
IGES
Formerly ANS US PRO/IPO-100-1996
Initial Graphics Exchange Specification
IGES 5.3
IGES/PDES Organization
Charleston 1996

[VBDC-11]:
Wim J.C. Verhagen, Pablo Bermell-Garcia, Reinier E.C. von Dijk, Richard Curran
A critical review of Knowledge-Based Engineering: An identification of research challenges
Advanced Engineering Informatics (2012)26:5-15
Elsevier

[VeRa-10]:
Arvind Kumar Verma, Sunil Rajotia
A review of machining feature recognition methodologies
International Journal of Computer Integrated Manufacturing (2010)28(4):353-368
Taylor & Francis

[ViKN-12]:
S. Vinodh, R. P. Kumar and N. Nachiappan
Disassembly modeling, planning, and leveling for a cam-operated rotary switch assembly: a case study
The International Journal of Advanced Manufacturing Technology (2012)62:789-800
Springer

[Voss-08]:
Gottfried Vossen
Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme
Oldenbourg Wissenschaftsverlag
München 2008
Seite 45

[VWBZ-09]:
Sandor Vajna, Christian Weber, Helmut Bley, Klaus Zeman
CAx für Ingenieure - Eine praxisbezogene Einführung
Springer
Berlin, Heidelberg 2009
Seite 499

[WaHT-07]:
Jiale Wang, Yuanjun He, Haishan Tian
Voxel-based shape analysis and search of mechanical CAD-models
Forschung im Ingenieurwesen (2007)71:189-195
Springer

[Weis-08]:
David E. Weisberg [Internet]
Englewood: The Engineering Design Revolution - The People, Companies and Computer Systems That Changed Forever the Practice of Engineering; c2008
www.cadhistory.net
Stand: 21.07.2015

[WeYu-08]:
Liu Wei, He Yuanjun
Representation and retrieval of 3D CAD models in parts library
The International Journal of Advanced Manufacturing Technology (2008)36:950-958
Springer

[WHDK-15]:
Michael Wiegand, Maik Hanel, Julia Deubner, Uwe Krieg
Konstruieren mit NX 10
Volumenkörper, Baugruppen und Zeichnungen
Carl Hanser
München 2015

[WhSO-05]:
David R. White, Sunil Saigal, Steven J. Owen
Meshing complexity: predicting meshing difficulty for single part CAD models
Engineering with Computers (2005)21:76-90
Springer

[Wink-10]:
Paul Winkelmann
A theoretical framework for an intelligent design catalogue
Engineering with Computers (2011)27:183-192
Springer

[Wuns-05]:
Matthias Wunsch
Wissensbasierte Konstruktion kundenindividueller Produkte am Beispiel von Schuhen
Shaker
Aachen 2005

[YanW-04]:
Yan Wang
Constraint-Enabled Design Information Representation for Mechanical Products over the Internet
ProQuest Information and Learning
Ann Arbor 2004

[Zhao-02]:
Li Zhao
A Work Structure Based Approach to Collaborative Engineering Design
ProQuest Information and Learning
Ann Arbor 2002

[ZöWL-99]:
Martina Zölch, Wolfgang G. Weber, Loni Leder
Praxis und Gestaltung kooperativer Arbeit
vdf Hochschulverlag
Zürich 1999

[ZSFH-10]:
Shusheng Zhang, Yunfei Shi, Haitao Fan, Rui Huang, Julu Cao
Serial 3D model reconstruction for machining evolution of rotational parts by merging semantic and graphic process planning information
Computer Aided Design (2010)42:781-794
Elsevier

Lebenslauf

Zur Person

Name:	Robin Roj
Email:	r.roj@uni-wuppertal.de
Geboren am:	18.01.1987
Geburtsort:	Wuppertal

Schulische Ausbildung

Grundschule:	1993-1997 Grundschule Meyerstraße
Gymnasium:	1997-2006 Carl Duisberg Gymnasium
Universität:	Wintersemester 2006/07 - Sommersemester 2009 Bachelorstudium Maschinenbau - Bergische Universität Wuppertal
	Wintersemester 2009/10 - Sommersemester 2011 Masterstudium Computational Mechanical Engineering - Bergische Universität Wuppertal
	seit dem Wintersemester 2011/12 Promotion im Fachbereich Maschinenbauinformatik - Bergische Universität Wuppertal