



FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Entwicklung eines Ontologie-basierten Ansatzes für die graphische Darstellung von 3D-Bauteileigenschaften

BACHELORARBEIT

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)
im Studiengang Informatik

FRIEDRICH-SCHILLER-UNIVERSITÄT JENA
Fakultät für Mathematik und Informatik

eingereicht von Florian LUDEWIG
geb. am 29.04.2001 in Schleiz

Universitäre Betreuer: Prof. Dr. Birgitta KÖNIG-RIES, Jan M. KEIL
Betriebliche Betreuer: Diana PETERS, Tobias KÖHLER

Jena, 22. September 2022

Kurzfassung

In der modernen Fertigungstechnik werden digitale Modelle als Repräsentation des herzustellenden Bauteils benutzt. Neben der Automatisierung von Fertigungsprozessen können diese Modelle genutzt werden, um herstellungsrelevante Eigenschaften zu extrahieren. Das Deutsche Zentrum für Luft- und Raumfahrt e. V. hat eine Software entwickelt, welche solche Bauteileigenschaften detektiert und in Form einer Ontologie speichert.

In dieser Arbeit soll eine Anwendung zur grafischen Visualisierung der extrahierten Bauteileigenschaften entwickelt werden. Dafür wird die generierte Ontologie zusammen mit dem 3D-Modell des Bauteils auf einer Serveranwendung analysiert und verarbeitet. Die generierten Daten werden anschließend an eine Benutzeroberfläche gesendet, um sie dort in einem 3D-Viewer darzustellen. Nutzer sind dann in der Lage, Modelle mit ihren Bauteileigenschaften grafisch zu analysieren.

Für die Evaluation wird die Vollständigkeit von vorher definierten Anforderungen geprüft und die Benutzbarkeit untersucht. Die Ergebnisse dieser Evaluation sind größtenteils positiv und lassen auf eine benutzerfreundliche Anwendung schließen. Abschließend werden potenzielle Möglichkeiten der Verbesserung und Erweiterung aufgeführt.

Inhaltsverzeichnis

Abbildungsverzeichnis	4
Tabellenverzeichnis	6
Algorithmenverzeichnis	7
Abkürzungsverzeichnis	8
1 Einleitung	10
2 Stand der Wissenschaft	11
2.1 Technik	11
2.1.1 Produktherstellungsprozess	11
2.1.2 Das STEP-Format	12
2.1.3 Open CASCADE Technology	13
2.1.4 Ontologien	14
2.2 Forschung	16
2.2.1 Visualisierung von CAD-Modellen	16
2.2.2 Bauteileigenschaften	17
2.2.3 Extraktion von Bauteileigenschaften	19
2.2.4 Herstellbarkeit von Bauteilen	20
3 Ziel der Arbeit	22
4 Konzept	23
5 Implementierung	26
5.1 Architektur des Frontends	26
5.2 Styling der Webanwendung	29

5.3	Client-Server-Kommunikation	30
5.4	Architektur des Servers	31
5.5	Lesen der Eigenschaften	31
5.6	Tesselierung des Brep-Modells	33
5.7	Zuordnung von Eigenschaften und Meshes	34
5.8	Interaktiver 3D Viewer	36
5.9	Informationen aus der Ontologie	38
5.10	Deployment der Anwendung	39
6	Evaluation	41
6.1	Methodik	41
6.2	Ergebnisse	42
7	Fazit	48
	Literaturverzeichnis	50

Abbildungsverzeichnis

2.1	Beispielhafte Hierarchie einer STEP-Datei, in Anlehnung an einem Flansch-Bauteil	12
2.2	Beispiel einer Ontologie	14
2.3	Beispiel von Instanzen einer Ontologie, welche aus den Klassen und Relationen von Abbildung 2.2 hervorgehen	15
2.4	Grafische Repräsentation der Relationen von Basic Features [26] . .	18
2.5	Grafische Repräsentation der Relationen von Manufacturing Features [25]	19
2.6	Struktur der Ontologie von Koehler et al. [26]	20
2.7	Grafische Repräsentation der Eigenschaften von Manufacturing Restrictions [25]	21
4.1	Flowchart des umgesetzten Konzeptes	24
5.1	Verkürzte Hierarchie der React-Komponenten von der Webanwendung dieser Arbeit	27
5.2	Aufgliederung der Benutzeroberfläche in ihre einzelnen React-Komponenten	28
5.3	Property Path vom Features:ManufacturingFeature bis zur ManOnSTEP:AdvancedFace	32
5.4	Property Path von der ManOnSTEP:AdvancedFace bis zur ManOnSTEP:EdgeCurve	32
5.5	Beispiel der Tessellierung eines CAD-Bauteils	33
5.6	Schema der unverarbeiteten Daten wie sie nach Abschnitt 5.5 und 5.6 vorliegen	35
5.7	Schema der zugeordneten Daten, wie sie an die Webanwendung gesendet werden	35

5.8	Initiales Aussehen des 3D-Viewers nach dem Laden eines Beispiel-Modells	37
5.9	Aussehen des 3D-Viewers nach dem Klicken auf eine Kante, welche zwischen zwei Manufacturing Features liegt	38
6.1	Die in dieser Arbeit entwickelte Anwendung getestet mit einem Beispiel-Bauteil in drei verschiedenen Web-Browsern auf Linux . .	42
6.2	Performanz-Profil der Anwendung bei standardmäßiger Nutzung	44
6.3	Vergleich zwischen dem 3D Viewer von FreeCAD und dieser Arbeit anhand eines Beispiel-Modells	45
6.4	Beispiel-Bauteil mit einer Vielzahl von Manufacturing Features . .	46
6.5	Beispiel eines Manufacturing Features, welches komplett von anderen Flächen eingeschlossen ist	46
6.6	Zur Evaluation der Performanz genutzte Bauteile	47

Tabellenverzeichnis

5.1	Ausschnitt der CSS-Variablen von dem in dieser Arbeit genutzten Farbschemas	30
6.1	Ergebnisse der Performanz-Messungen von den Bauteilen aus Abbildung 6.6	47

Algorithmenverzeichnis

1	Rekursiver Algorithmus zur Ermittlung von Relationen einer Instanz in der Ontologie	39
---	---	----

Abkürzungsverzeichnis

ACAPP	Automatic Computer-Aided Process Planning
AP	Application Protocol
Brep	Boundary representation
CAD	Computer-Aided Design
CAPP	Computer-Aided Process Planning
CPU	Central Processing Unit
CSS	Cascading Style Sheets
GPU	Graphics Processing Unit
GLSL	OpenGL ES Shading Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Standards Organization
JSON	JavaScript Object Notation
PBR	Physically Based Rendering
PMI	Product Manufacturing Information
RDFS	Resource Description Framework Schema
SSR	Server Side Rendering
STEP	Standard for the Exchange of Product model data

OCCT	Open CASCADE Technology
OWL	Web Ontology Language
OpenGL	Open Graphics Library
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
VTK	Visualization Toolkit
WebGL	Web Graphics Library
XML	Extensible Markup Language
X3D	Extensible 3D graphics

1 Einleitung

Bei der Fertigung von Bauteilen ist es üblich, ein digitales Modell des herzustellenden Bauteils zu erstellen [53]. Das dabei entstandene 3D-Modell dient vor allem der Automatisierung von Fertigungsprozessen [38]. Aber das Modell kann beispielweise auch genutzt werden, um für den Fertigungsprozess relevante Eigenschaften zu extrahieren. Von einem 3D-Modell extrahierte Eigenschaften, die in einer Ontologie gespeichert sind, wie Löcher, Stufen, Konturen oder Rillen, sind nützlich, um den Herstellungsprozess des Bauteils zu optimieren [26]. In etwa um Kosten und Lieferzeiten zu reduzieren, beispielsweise durch Sparen von Material. Allerdings sind die Daten des 3D-Modells und deren Zusammenhänge mit der Ontologie für den Menschen nur schlecht lesbar.

Im Rahmen dieser Bachelorarbeit sollen fertigungsrelevante Eigenschaften möglichst übersichtlich in einer grafischen Oberfläche dargestellt werden. Die extrahierten Daten eines Bauteils liegen in einer Ontologie vor. Diese Wissensbasis wird zusammen mit der 3D-Repräsentation des Bauteils benutzt, um das Bauteil mit seinen Eigenschaften in einem Web-basierten 3D-Viewer zu präsentieren. Dadurch können die Eigenschaften vom Menschen einfacher nachvollzogen werden.

Dazu werden zunächst technische Grundlagen erklärt und ähnliche Arbeiten betrachtet. Daraufhin wird anhand von mehreren Anforderungen das Konzept der in dieser Arbeit entwickelten Anwendung erarbeitet und dessen Implementierungsdetails beschrieben. Schließlich wird die erstellte Anwendung auf ihre Qualität geprüft.

2 Stand der Wissenschaft

2.1 Technik

Dieses Kapitel gibt einen Überblick zu grundlegenden Konzepten und Technologien, welche in dieser Arbeit eine wichtige Rollen spielen.

2.1.1 Produktherstellungsprozess

Aufgrund der Globalisierung und der damit einhergehenden steigenden Konkurrenz müssen sich Unternehmen immer schneller adaptieren, um wettbewerbsfähig zu bleiben [4]. Dadurch wurde in den vergangenen vier Jahrzehnten der Herstellungsprozess von Produkten stark beeinflusst [53]. Die dabei vorangetriebene Digitalisierung der Produktplanung brachte Computer-Aided Design (CAD) und Computer-Aided Process Planning (CAPP) hervor. Dabei werden Design und Herstellung mithilfe von Computertechnologien verbunden [2]. Designer entwerfen digitale Bauteile (CAD), welche dann genutzt werden, um Herstellungsanweisungen zu erzeugen (CAPP). Schlussendlich kann mithilfe dieser Anweisungen, durch bestimmte Fertigungsmethoden, ein physisches Objekt hergestellt werden [55]. Die Etablierung von CAD und CAPP resultierte in einer Verbesserung der Effizienz und Flexibilität von Herstellungsprozessen [38]. Der Grund dafür ist die sukzessive Automatisierung dieser Verfahren. Diese automatisierten Prozesse werden auch als Automatic Computer-Aided Process Planning (ACAPP) bezeichnet [51]. Allerdings gehen damit auch Herausforderungen wie die Erstellung von geometrischen Modellen, die Integration von Design und Produktionsplanung oder die Standardisierung von Austauschprotokollen einher [52, 38, 54].

2.1.2 Das STEP-Format

Zur Vereinheitlichung von CAD-Austauschprotokollen wurde 1994 der ISO 10303 Standard veröffentlicht, welcher das STEP-Format (Standard for the Exchange of Product model data) definiert [24]. Es beschreibt die genaue Struktur von technischen Produktdaten und verwendet die Informationsmodellierungssprache EXPRESS [21], wodurch STEP erweiterbar für spezielle Anwendungsfälle ist [29]. So entstanden mit der Zeit auch mehrere Anwendungsprotokolle (Application Protocols (APs)). Beispielsweise gibt es das AP242, welches Rahmenbedingungen für modellbasiertes 3D-Engineering setzt. Durch die internationale Standardisierung und die Erweiterbarkeit ermöglicht STEP den reibungslosen Austausch von Bauteilen und kommt deshalb auch häufig in der Industrie zum Einsatz [1, 47]. Deswegen wird STEP von einer Vielzahl unterschiedlicher CAD-Modellierungssoftware unterstützt [27].

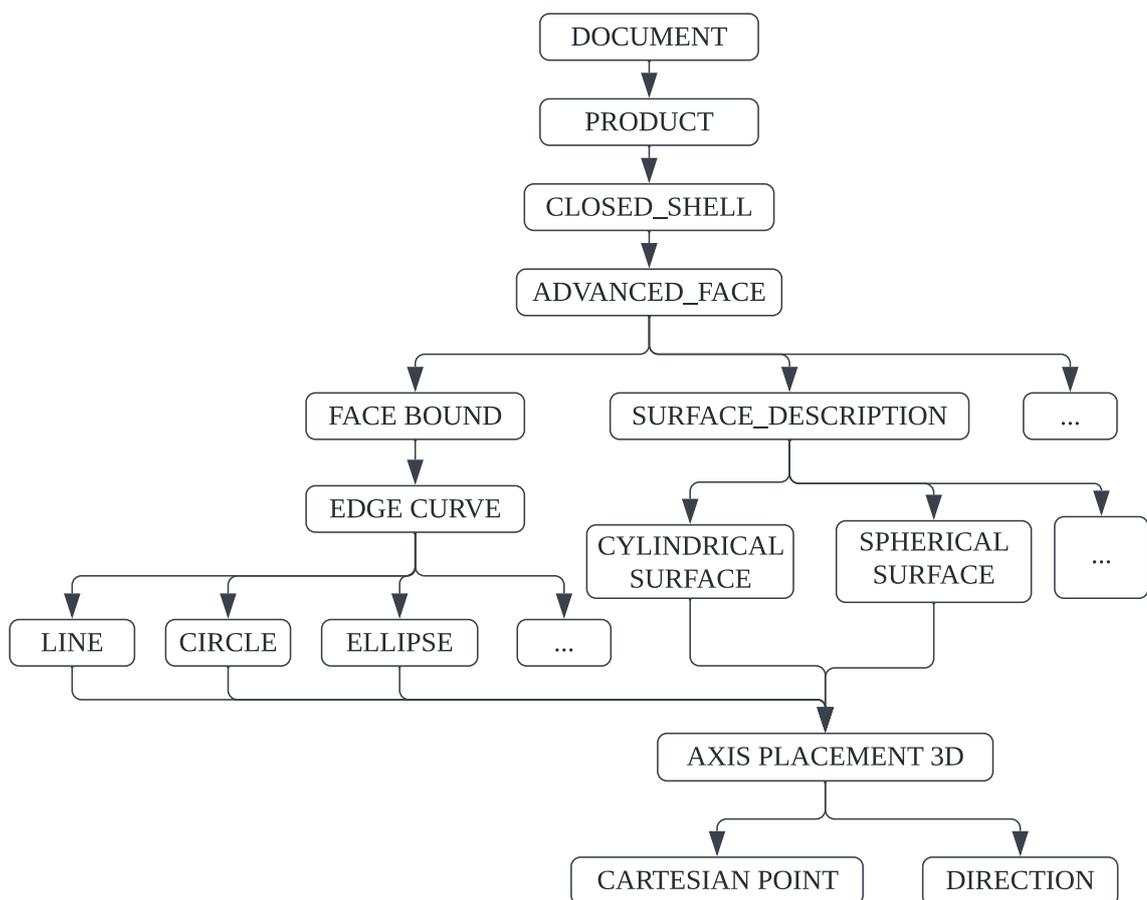


Abbildung 2.1: Beispielhafte Hierarchie einer STEP-Datei, in Anlehnung an einem Flansch-Bauteil

Modelle werden in einer hierarchischen Struktur von Entitäten dargestellt. Der Aufbau einer beispielhaften STEP-Datei ist in der Abbildung 2.1 zu se-

hen. Die Grafik wurde manuell erstellt und orientiert sich an die Referenzierungen von den Entitäten einer STEP-Datei. In der Wurzel der Hierarchie befindet sich ein Dokument DOCUMENT, welches ein Objekt PRODUCT enthält. Die Geometrie dieses Objektes ist eine geschlossene Hülle (CLOSED_SHELL), welche aus Flächen (ADVANCED_FACE) besteht. Die Flächen sind wiederum aus den Kanten LINE, CIRCLE und ELLIPSE aufgebaut. Schlussendlich wird die Platzierung der vorher genannten Entitäten im 3D-Raum durch Vektoren wie CARTESIAN_POINT und DIRECTION beschrieben. Jede dieser Entitäten hat dabei einen eindeutigen Identifikator. Zudem können mehrere Argumente an die Entitäten übergeben werden, welche entweder andere Entitäten referenzieren oder einfache Datentypen wie Zahlen oder Zeichenketten sind [44]. Somit ist es möglich komplexe Strukturen darzustellen, welche als Ganzes ein 3D-Modell beschreiben.

2.1.3 Open CASCADE Technology

Um mit dem komplexem STEP-Format zu arbeiten, werden spezialisierte Werkzeuge benötigt. Eines dieser Werkzeuge ist Open CASCADE Technology (OCCT)¹. OCCT ist ein open-source CAD-Kernel zur geometrischen Modellierung. Damit ist es möglich, 3D-Geometrien mithilfe der Boundary representation (Brep) zu analysieren und modifizieren. Brep bedeutet, dass die Geometrie als mathematische Beschreibung von Formen betrachtet wird, und nicht etwa als diskrete Menge von Eckpunkten, Kanten und Flächen, wie es bei Meshes der Fall ist [36]. OCCT unterstützt auch nativ das STEP-Format und das AP242.

Die C++ Bibliothek besteht aus sechs Modulen. Das erste und grundlegende Modul enthält fundamentale Klassen mit Datenstrukturen und Datentypen für die darüberliegenden Module. Ein weiteres Modul beschäftigt sich mit der Brep-Modellierung, welche die Geometrie als mathematische Beschreibung von Formen betrachtet [36]. Dazu werden Klassen für Punkte, Kanten und Flächen bereitgestellt, um damit geschlossene Oberflächen und zusammengesetzte Strukturen darzustellen. Im dritten Modul befinden sich einige Algorithmen zum Bearbeiten von Geometrien, wie zum Beispiel die Berechnung von Schnittpunkten und Schnittkanten. Zudem gibt es ein Modul für den Datenaustausch. Es implementiert Methoden zum Lesen und Schreiben von CAD-Bauteilen in verschiedenen Formaten. Unter anderem gehören dazu STEP, IGES, glTF und OBJ. [6]

Neben der C++ Bibliothek von OCCT gibt es auch PythonOCC². Es bietet vollen Zugriff auf alle Klassen und Methoden der C++ Bibliothek mittels Python.

¹<https://dev.opencascade.org>

²<https://github.com/tpaviot/pythonocc-core>

2.1.4 Ontologien

Eine Möglichkeit zur Modellierung komplexer Strukturen ist der Einsatz von Ontologien. In der Informatik ist unter dem Begriff "Ontologie" eine formale Darstellung von Wissen zu verstehen. Ontologien bestehen aus drei zentralen Elementen: Klassen, Relationen und Instanzen. Klassen sind Konzepte und werden durch Relationen miteinander verbunden. Klassen und Relation stellen somit die Terminologie für die Ontologie dar. Instanzen hingegen repräsentieren auftretende Terminologien und modellieren Assoziationen untereinander. [18].

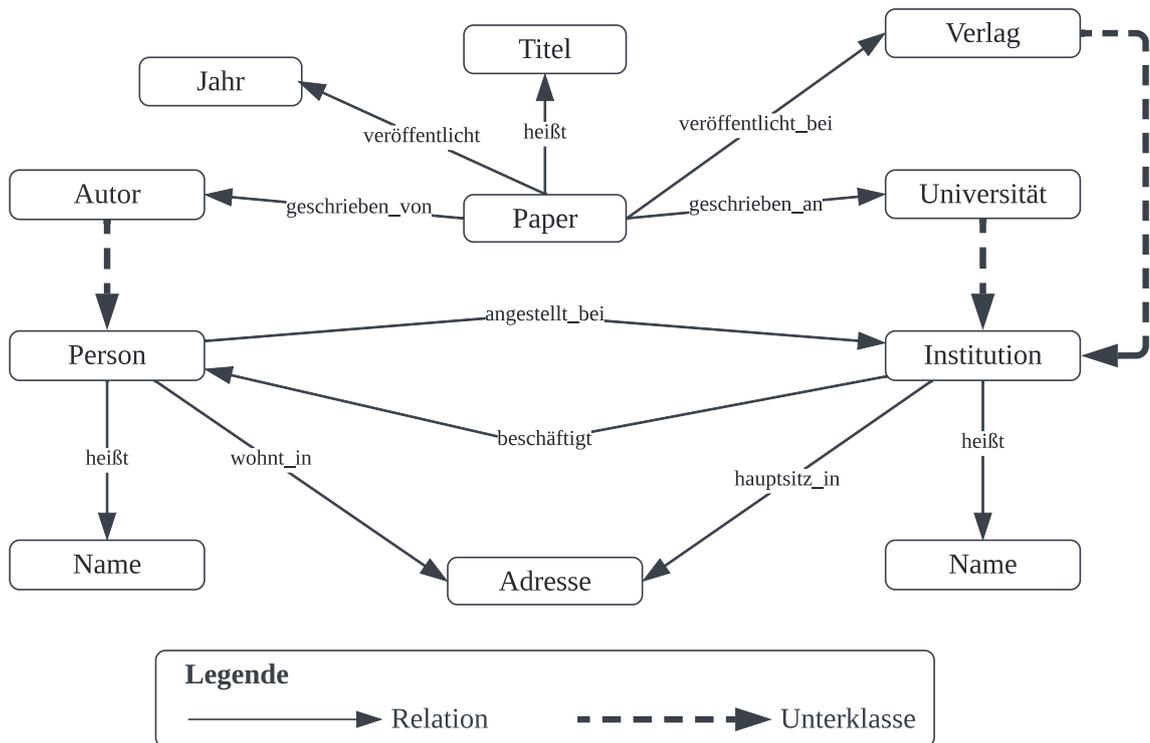


Abbildung 2.2: Beispiel einer Ontologie

In Abbildung 2.2 ist eine beispielhafte Ontologie zu sehen. Die Instanzen der Klasse Paper können einen Titel, einen Autor, ein Jahr der Veröffentlichung und weitere Relationen haben. Dabei ist Autor eine Unterklasse von Person. Personen haben wiederum einen Name, welcher durch eine Zeichenkette repräsentiert wird, genauso wie der Titel des Papers. Gleichermäßen können Personen an einer Institution beschäftigt sein, welche beispielweise die Universität sein kann, an der das Paper geschrieben wurde. Dabei stellt Abbildung 2.2 lediglich ein generalisiertes Datenmodell dar.

Ontologien können zudem mithilfe von Instanzen Daten speichern. Dazu werden Daten durch Instanzen einzelner Klassen repräsentiert. So können beliebig viele Instanzen der Klassen erzeugt und untereinander in Relation ge-

2.2 Forschung

In diesem Abschnitt wird zunächst der aktuelle Forschungsstand beleuchtet. Daraufhin wird die von dieser Arbeit zu schließende Lücke in der Forschung aufgezeigt.

2.2.1 Visualisierung von CAD-Modellen

3D-Modelle sind oft nur dann nützlich, wenn sie visualisiert werden können. Je nachdem, welches 3D-Format verwendet wird, gibt es verschiedene Tools zur Visualisierung.

Cignoni et al. veröffentlichten 2008 ein open-source Programm zur Verarbeitung von Meshes namens Meshlab [7]. Es beinhaltet sowohl einen 3D-Viewer basierend auf der Open Graphics Library (OpenGL) als auch zahlreiche Funktionalitäten zum Bearbeiten von Meshes. Darunter zählen beispielsweise die Konvertierung zwischen verschiedenen Formaten, Mesh-Reinigungsfilter oder Werkzeuge zum Messen. In diesem Zusammenhang ist unter dem Begriff einer Mesh, eine 3D-Geometrie zu verstehen, welche auf diskreten Flächen und Eckpunkten basiert [36]. Somit können Brep-Modelle, welche 3D-Geometrien als mathematische Formen betrachten, von Meshlab nicht importiert und angezeigt werden. Das bedeutet, dass das STEP-Format nicht mit MeshLab kompatibel ist.

Slyadnev et al. haben 2017 ein Framework zur Analyse von CAD-Modellen eingeführt, welches insbesondere für Brep-Modelle optimiert ist [43]. Das open-source Framework stellt viele Operationen zur Modellierung von CAD-Dateien bereit. Zudem ist es möglich 3D-Geometrien visuell zu analysieren. Für die Visualisierung wird hierbei das Visualization Toolkit (VTK) von Schroeder et al. [40] genutzt. Ähnlich zu dem Framework von Slyadnev et al. ist FreeCAD³, ein Programm zur parametrischen Modellierung und unterstützt somit ebenfalls Brep-Modelle. Sowohl FreeCAD als auch das Framework von Slyadnev et al. nutzen das in Kapitel 2.1.3 vorgestellte OCCT zur Verarbeitung der CAD-Modelle [46, 43].

Die bisher vorgestellten Projekte konzentrieren sich alle auf die Entwicklung von Anwendungen für den Desktop. Um plattformunabhängige 3D-Software zu entwickeln, kann beispielsweise die Web Graphics Library (WebGL) genutzt werden. Diese Bibliothek wurde 2011 veröffentlicht [17] und ist eine abgewandelte Version von OpenGL für den Browser [3]. Seit 2017 unterstützen die meisten Browser auch WebGL 2.0 [16]. Alle Anfragen an die WebGL-Schnittstelle durch

³<https://www.freecadweb.org>

JavaScript werden an die lokale Hardware, in den meisten Fällen eine Graphics Processing Unit (GPU), weitergeleitet. Dafür ist eine Kombination aus JavaScript- und Shader-Code nötig. Der Shader-Code, welcher einzelne Bildpunkte berechnet, wird dabei in OpenGL ES Shading Language (GLSL) geschrieben [3].

Seit der Einführung von WebGL werden immer mehr Web3D Anwendungen erstellt. Darunter zählen zum Beispiel Anwendungen zur Visualisierung von Karten und Städten oder zur Simulation von Fabriken [23, 34, 39]. Auch Potenziani et al. nutzen WebGL um kulturelles Erbe in Form eines 3D-Viewers online zugänglich zu machen [37]. Im Zuge ihrer Arbeit haben sie eine modulare JavaScript-Bibliothek entwickelt, welche die Visualisierung hochauflösender 3D-Modelle ermöglicht.

Allerdings unterstützt die Bibliothek von Potenziani et al. keine Brep-Modelle. Um diese Problematik zu lösen, befassten sich Simões et al. mit der Entwicklung eines Web-basierten Systems zur Interaktion mit industriellen CAD-Modellen [42]. Mithilfe eines C++ Backends konvertieren sie STEP AP242 Dateien automatisch in das Extensible 3D graphics (X3D) Format [11], welches dann in die interaktive Webanwendung importiert werden kann. Dazu verwenden Simões et al. OCCT. Unter anderem wird dabei die Geometrie vereinfacht, von Artefakten befreit und mit Physically Based Rendering (PBR) Texturen ausgestattet. Die generierten Daten können anschließend in einem 3D-Viewer angezeigt und mithilfe eines Editors animiert werden.

2.2.2 Bauteileigenschaften

Im Verlauf dieser Arbeit werden Eigenschaften von CAD-Bauteilen eine wichtige Rolle spielen, weswegen sich dieser Abschnitt damit genauer beschäftigt. Im Zusammenhang mit CAD-Modellen ist eine Eigenschaft eine Teilform oder Charakteristik, welche als Einheit betrachtet wird und Bedeutung für Herstellungsprozesse hat [50]. Diese Teilformen können, müssen aber nicht, geometrische Charakteristiken beschreiben. Nach Weber muss die Repräsentation und das Verhalten von Features zudem eindeutig formalisiert sein [48]. Oberflächenbestandteile mit ähnlichen Beschaffenheiten beschreiben dann die selbe Eigenschaft. Es gibt verschiedene Ansätze zur Erkennung und Klassifizierung solcher Bauteileigenschaften anhand eines CAD-Modells [41]. Nach Absprache mit dem Deutschen Zentrum für Luft- und Raumfahrt e. V. wird in dieser Arbeit die Klassifizierung von Koehler et al. genutzt. Dabei wird zwischen grundlegenden Eigenschaften ("Basic Features") und herstellungsspezifischen Eigenschaften ("Manufacturing Features") unterscheiden [26].

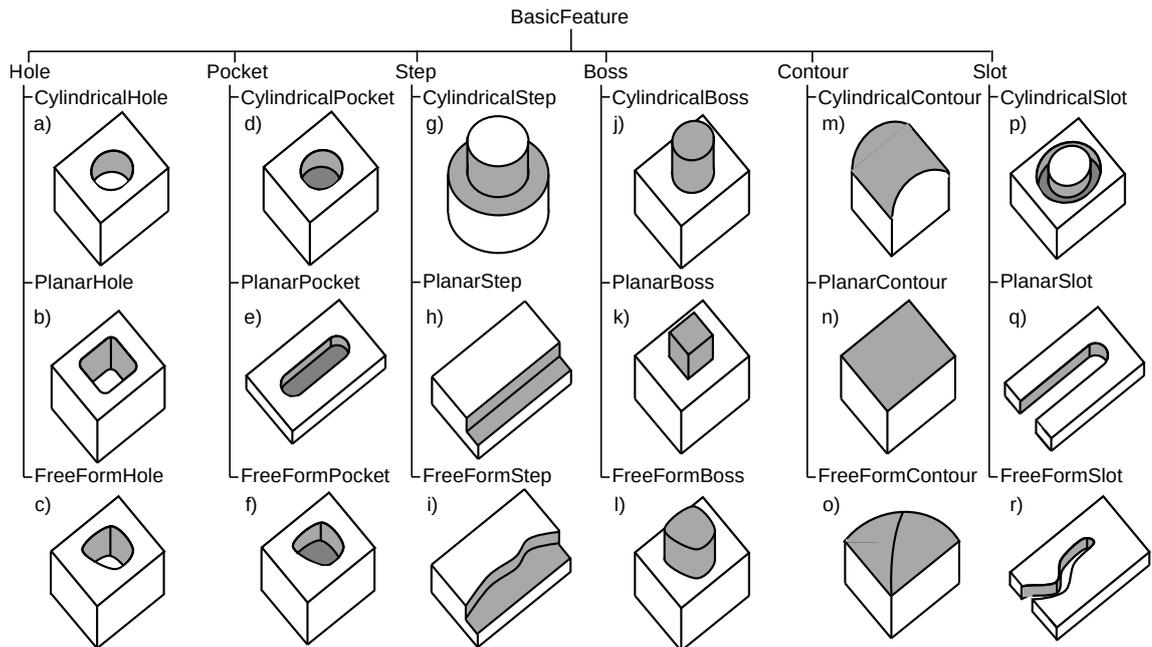


Abbildung 2.4: Grafische Repräsentation der Relationen von Basic Features [26]

Koehler et al. führen zudem sechs Typen von Basic Features ein: "Hole", "Pocket", "Step", "Boss", "Contour", und "Slot", welche selbst auch Variationen besitzen. Die verschiedenen Basic Features mit ihren Variationen werden beispielhaft in Abbildung 2.4 dargestellt. Dabei ist ein Hole eine Reihe von konkav verbundenen Flächen, die keine konkav verbundene Grundfläche besitzen. Ein Pocket dagegen, ist eine Reihe von konkav verbundenen Flächen, die eine konkav verbundene Grundfläche besitzen. Ein Step ist durch eine Seitenfläche mit einer konkav verbundenen Grundfläche ausgezeichnet. Ein Boss hat eine Menge von konkav verbundenen Seitenflächen, welche mit den Seiteninnenkanten einer Grundfläche verbunden sind. Eine Contour ist eine Fläche ohne konkav verbundene, äußere Flächen. Ein Slot besteht aus zwei oder mehr angrenzend verbundenen Steps. All diese Charakteristiken werden als Basic Feature bezeichnet, weil sie möglichst die gesamte Oberfläche des CAD-Modells abdecken sollten. Das heißt, fast alle Flächen des Modells sollen mindestens einem Basic Feature zugewiesen werden. [26]

Im Gegensatz zu Basic Features sollen Manufacturing Features für den Herstellungsprozess relevante Informationen bereitstellen. Manufacturing Features werden ebenfalls in verschiedene Untergruppen eingeteilt und werden durch ein oder mehrere Basic Features definiert. Wie in Abbildung 2.5 dargestellt, erfolgt deren Untergliederung grob in: "RotaryFeature", "PlanarFeature" und "FreeFormFeature". Diese Klassen haben wiederum alle weitere Unterklassen. Al-

le Manufacturing Features haben signifikante Relevanz für die Herstellungsanweisungen des Modells. So kann beispielsweise ein “InnerRotaryFeature” durch Bohren in der Herstellung umgesetzt werden. [26]

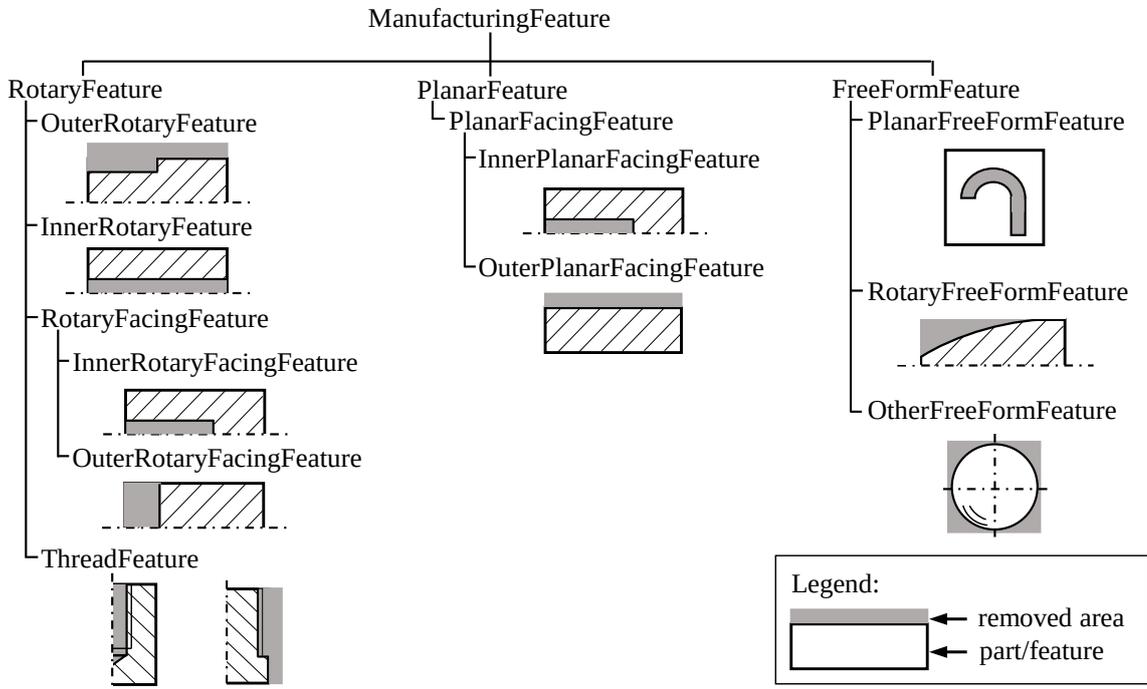


Abbildung 2.5: Grafische Repräsentation der Relationen von Manufacturing Features [25]

2.2.3 Extraktion von Bauteileigenschaften

Dieser Abschnitt soll dabei helfen zu verstehen, wie die im vorherigen Abschnitt beschriebenen Bauteileigenschaften von Koehler et al. detektiert werden. Als Eingabe dient eine STEP-Datei, welche dann in einem Zwei-Phasen-Prozess analysiert wird. Als erstes wird das STEP-Modell in eine Ontologie konvertiert, woraufhin diese Ontologie auf Eigenschaften durchsucht wird.

Die Konvertierung vom STEP-Format in eine Ontologie erfolgt mithilfe eines von Koehler et al. entwickelten Python-Skripts. Zunächst werden die Objekte des STEP-Schemas als Ontologie repräsentiert. Dazu werden Entitäten durch Klassen und Attribute durch Relationen modelliert. Danach können die Instanzen der STEP-Datei mithilfe von OWL in eine Ontologie eingepflegt werden. Nun liegen alle wichtigen Daten in Form einer Ontologie vor, welche als Eingabe für den nächsten Schritt der Erkennung der Eigenschaften dient. Hier werden Basic Features durch topologische Beschreibungen der Geometrie gefunden. Eine solche topologische Beschreibung könnte beispielsweise wie folgt heißen: “eine

Menge konkav verbundener Flächen, die zusammen eine geschlossene Kurve bilden“. Die Eigenschaften werden in eine hierarchische Struktur mit Unter- und Überklassen gegliedert. Diese Struktur wurde bereits in Abschnitt 2.2.2 beschrieben. [26].

Anschließend werden die gefunden Basic Features genutzt, um Manufacturing Features zu finden. Da Manufacturing Features aus einem oder mehreren Basic Features bestehen, werden Komposition, Form, Orientierung und Position von Basic Features zur Erkennung von Manufacturing Features verwendet. Die Gliederung in Basic und Manufacturing Features stellt dann neben der hierarchischen Struktur der Basic Features auch einen geschichteten Aufbau der Eigenschaften dar [26].

Die aus der Extraktion resultierende Ontologie hat die in Abbildung 2.6 dargestellte grundlegende Struktur. Darin ist zu erkennen, dass Basic Features die STEP-Entitäten beschreiben und Manufacturing Features das zugrunde liegende Bauteil. Zusätzlich werden die Manufacturing Features durch Basic Features definiert. Die Bedeutung der ManufacturingRestrictions wird in Abschnitt 2.2.4 genauer erklärt.

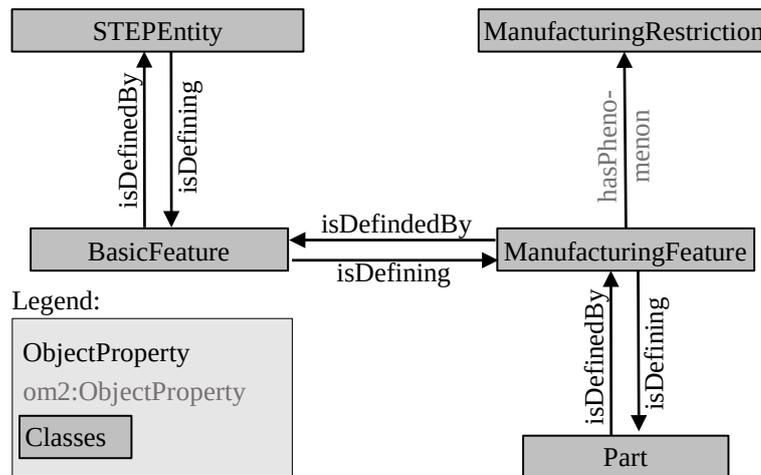


Abbildung 2.6: Struktur der Ontologie von Koehler et al. [26]

2.2.4 Herstellbarkeit von Bauteilen

Neben den Manufacturing Features sind noch weitere Informationen für die Herstellung wichtig. Je nach Herstellungsprozess gibt es unterschiedliche Fähigkeiten und Einschränkungen, welche von der genutzten Maschine abhängen. Um zu wissen, ob eine bestimmte Maschine ein bestimmtes Bauteil produzieren kann, müssen diese Einschränkungen modelliert werden. Koehler

et al. nutzen die in Abbildung 2.7 dargestellte Unterteilung in vier verschiedene "Manufacturing Restrictions": "Accuracy", "Resolution", "Manufacturability", "SurfaceStructure". Auch diese können noch in Untergruppen aufgeteilt werden. Die Accuracy ist dabei ein Maß für die Genauigkeit von geometrischen Elementen. Beispielsweise die Rundheit eines Zylinders oder die Ebenheit einer Fläche. Die Resolution beschreibt die Minimalgrößen, die von einem Prozess hergestellt werden können. Darunter zählen beispielsweise die minimale Wandstärke oder der minimale Lochdurchmesser. Die Manufacturability beschreibt Voraussetzungen, welche erfüllt sein müssen, um ein Bauteil mittels additiver Fertigung herzustellen. Beispielsweise können Überhangstrukturen oder Bohrungen mit zu kleinem Durchmesser nicht hergestellt werden. Schließlich gibt es noch die SurfaceStructure Einschränkungen, welche Limitierungen von geschichteten Fertigungsprozessen, wie dem 3D-Druck, beschreibt. Durch das schrittweise Auftragen von Schichten können abhängig von der Schichtenhöhe zum Beispiel stufenartige Artefakte bei schrägen Wänden entstehen. [25]

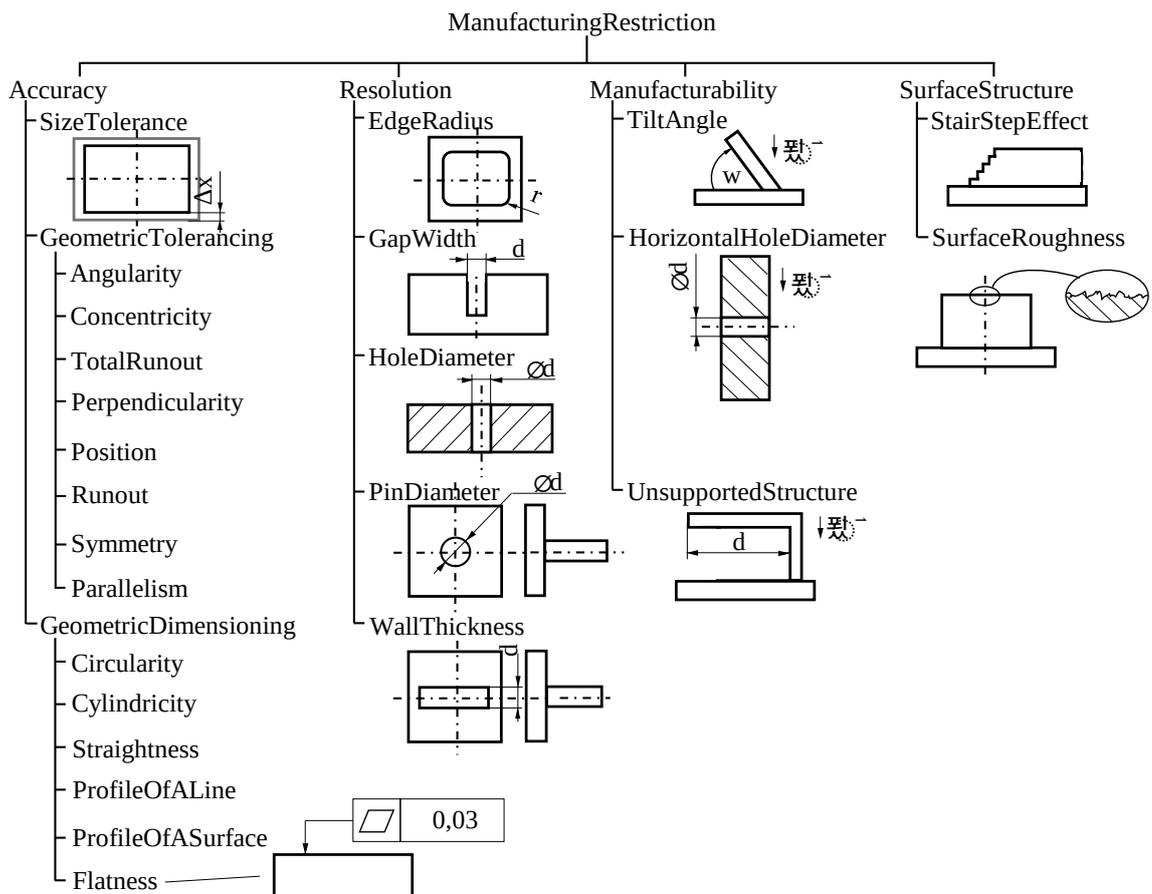


Abbildung 2.7: Grafische Repräsentation der Eigenschaften von Manufacturing Restrictions [25]

3 Ziel der Arbeit

Bauteileigenschaften wie sie in Abschnitt 2.2.2 beschrieben werden, haben einen direkten Einfluss auf den Herstellungsprozess von CAD-Bauteilen. Die extrahierten Bauteileigenschaften, welche in einer Ontologie vorliegen, können problemlos für automatisierte Prozesse genutzt werden. Allerdings sind die Beschreibungen der geometrischen Struktur des Modells im Zusammenhang mit der Ontologie nicht besonders nützlich für das Verständnis der beteiligten Personen. Deswegen ist es nötig die Bauteileigenschaften zusammen mit dem 3D-Modell grafisch zu visualisieren. Nur so können die detailreichen Daten effizient nachvollzogen werden. Die Kombination der Eigenschaften aus der Ontologie mit dem 3D-Modell ist somit der wissenschaftliche Ansatz dieser Arbeit.

In Abschnitt 2.2.1 wurden deswegen mehrere Arbeiten zur Visualisierung von CAD-Modellen gezeigt, welche sowohl als Desktop-Anwendung, also auch als Webanwendung bereitgestellt werden. Einige davon unterstützten sogar Brep-Modelle und damit auch das STEP-Format. Zudem wurde skizziert wie Koehler et al. Ontologien nutzen um Bauteileigenschaften zu extrahieren und zu modellieren (Abschnitt 2.2.2, 2.2.3 und 2.2.4). Allerdings gibt es zum Zeitpunkt des Schreibens keine Arbeiten die eine Methode vorstellen, welche plattformunabhängig ist, Brep-Modelle unterstützt und von Ontologien beschriebene Eigenschaften darstellen kann.

Somit stellt ein plattformunabhängig CAD-Viewer, welcher durch Ontologien repräsentierte Bauteileigenschaften visualisieren kann, eine Lücke in der Forschung dar, welche durch diese Arbeit geschlossen werden soll. Zur Umsetzung wurde eine Anwendung mit Server-Client-Architektur entwickelt. Die Benutzeroberfläche ist eine Webanwendung, welche das Bauteil mit seinen Eigenschaften in einem 3D-Viewer darstellt. Die dafür notwendigen Daten werden anhand der von den Nutzern bereitgestellten Eingabe-Dateien durch eine Serveranwendung generiert.

4 Konzept

Nachdem im vorigen Kapitel der wissenschaftliche Ansatz der Arbeit beschrieben wurde, sind in diesem Kapitel alle Anforderungen an die zu erstellende Anwendung aufgeführt. Diese Vorgaben wurden mit dem Deutschen Zentrum für Luft- und Raumfahrt e. V. (DLR) ausgearbeitet. Weiterhin wird das Konzept beschrieben mit welchem die Anforderungen umgesetzt wurden.

- A1** Die Anwendung soll plattformunabhängig sein.
- A2** Die Bedienung der Anwendung soll so einfach und intuitiv sein, dass Personen mit grundlegender Erfahrung in den Bereichen CAD und Ontologien, Bauteile grafisch analysieren können.
- A3** Es muss eine externe Software eingebunden werden, welche die Bauteileigenschaften von einer STEP-Datei detektiert und in einer OWL-Datei speichert.
- A4** Die Endgeräte der Nutzer sollen möglichst stark von aufwendigen Berechnungen entlastet werden.
- A5** Das Bauteil soll grafisch und dreidimensional mit seinen einzelnen Flächen und Kanten angezeigt werden.
- A6** Nutzer sollen die Möglichkeit haben das Bauteil interaktiv von allen Seiten zu betrachten.
- A7** Eigenschaften sollen aufgelistet werden und durch Auswählen auf dem 3D-Modell hervorgehoben werden.
- A8** Eigenschaften sollen durch Klicken auf die Flächen und Kanten des Modells auswählbar sein.

- A9 Falls mehrere Eigenschaften zu einer Fläche oder Kante gehören sollen auch mehrere Eigenschaften hervorgehoben werden, wenn Nutzer auf diese Kante oder Fläche klicken.
- A10 Durch andere Flächen vollkommen eingeschlossene Bauteileigenschaften sollen erkennbar sein.
- A11 Details zu den Eigenschaften aus der Ontologie sollen angezeigt werden.
- A12 Die Anwendung soll auf aktuellen Betriebssystem, wie Windows, Linux und macOS gehostet werden können.

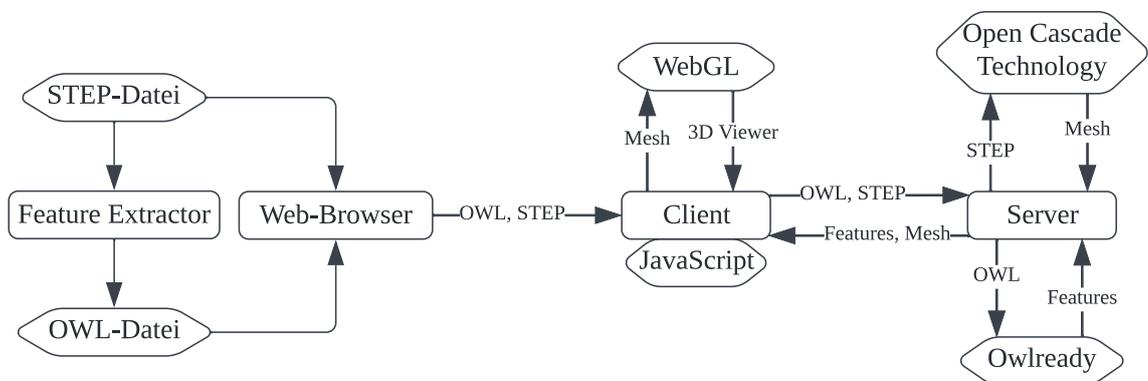


Abbildung 4.1: Flowchart des umgesetzten Konzeptes

Zur Realisierung der Anforderungen wurde eine Client-Server-Architektur gewählt, welche aus einer Anwendung für die Endnutzer und einer separaten Serveranwendung besteht. Die Architektur der Umsetzung ist in Abbildung 4.1 skizziert. Um die Plattformunabhängigkeit (A1) und die einfache Bedienung (A2) zu gewährleisten ist die Anwendung für die Endnutzer eine Webanwendung, welche in jedem modernen Browser ausführbar ist. Als Eingabe werden zwei Dateien erwartet, eine STEP- und eine OWL-Datei. Die OWL-Datei ist dabei das Resultat von dem in Abschnitt 2.2.3 beschriebenen Prozess (A3). Diese Dateien sendet die Anwendung nachfolgend an einen Server. Um das Endgerät zu entlasten werden dort die rechenaufwendigen Prozesse ausgeführt (A4). Dazu zählen das Auslesen von STEP-Entitäten, die Auflistung aller Eigenschaften aus der Ontologie, die Verknüpfung zwischen den STEP-Entitäten und den Eigenschaften, und die Konvertierung des Brep-Modells in eine diskrete Mesh. Nachdem der Server die generierten Daten zurück an die Webanwendung gesendet hat, wird das Modell mithilfe von WebGL in einem 3D-Viewer dargestellt (A5, A6). Mithilfe eines JavaScript-Frameworks wird der Zustand der Webanwendung verwaltet

(A7, A8, A9). Die Flächen des Modells können in einen transparenten Modus versetzt werden um Eigenschaften zu erkennen, welche vollkommen von anderen Flächen eingeschlossen sind (A10). Weiterhin sendet der Server auch Informationen über die Bauteileigenschaften aus der Ontologie mit an die Webanwendung, welche innerhalb des 3D-Viewers angezeigt werden (A11). Außerdem wird sowohl der Server als auch die Webanwendung in eine Umgebung gebündelt, welche in beliebigen Produktivsystemen ausgeführt werden kann (A12). Die genaue Implementierung der einzelnen Konzepte wird ausführlich in den folgenden Kapiteln beschrieben.

5 Implementierung

In diesem Kapitel werden die Details der Implementierung und Design-Entscheidungen des in Kapitel 4 skizzierten Konzeptes beschrieben.

5.1 Architektur des Frontends

Um den Endnutzern eine einfache Schnittstelle anzubieten, wurde für die Umsetzung dieser Arbeit eine Webanwendung gewählt. Dazu wird das Web Framework Next.js¹ verwendet. Web Frameworks erleichtern den Aufbau einer Webanwendung, da diese ein Grundgerüst bereitstellen. So müssen Funktionalitäten wie Komponenten- und Zustandsverwaltung nicht selbst implementiert werden. Next.js fügt der bekannten React.js² Bibliothek zusätzliche Funktionen hinzu. Zum Beispiel kann mithilfe von Next.js Server Side Rendering (SSR) verwendet werden. Bei herkömmlichen Webanwendungen gibt der Server eine HTML-Seite ohne sichtbaren Inhalt zurück, welche dann im Browser der Nutzer von JavaScript [22] befüllt wird. Mit SSR geschieht dies schon auf dem Server, sodass Nutzer bereits das befüllte HTML-Dokument erhalten. Dadurch entlastet SSR das genutzte Endgerät [20]. Next.js kann auch mit TypeScript [10] verwendet werden. TypeScript ist eine Erweiterung von JavaScript und erleichtert durch die Möglichkeit Datentypen zu spezifizieren die Arbeit in größeren Projekten. Zudem ist es mit Next.js problemlos möglich, Code-Bibliotheken von Dritten in das Projekt einzubinden, welche beim Kompilieren automatisch optimiert werden [49]. Das heißt, dass ungenutzter Code aus den finalen JavaScript-Paketen entfernt wird, wodurch sich die Größe der Webanwendung reduziert.

Next.js-Projekte setzen sich aus drei Bestandteilen zusammen: die Seiten der Anwendungen, die Komponenten und die Kontexte. Für jede Seite

¹<https://nextjs.org>

²<https://reactjs.org>

wird eine Komponente als Einstiegspunkt angelegt. Weiterhin stellen React-Komponenten [32] einen wichtigen Bestandteil der Anwendung dar. Komponenten sind HTML-Bausteine, welche beliebig oft wiederverwendet werden können. Um unnötigen Code zu vermeiden und die Übersichtlichkeit des Codes zu steigern, sollte die Anwendung in kleinere Komponenten gegliedert werden. Dabei hat jede Komponente einen Lebenszyklus, eine Render-Funktion und optionale Eingabe-Parameter. Von der Render-Funktion wird der anzuzeigende HTML-Code zurückgegeben. Der HTML-Code kann wiederum weitere Komponenten enthalten und sich dynamisch verändern. Somit hat jede React-Anwendung einen Komponenten-Baum, welcher die hierarchische Struktur der Anwendung widerspiegelt. In Abbildung 5.1 ist die Hierarchie der Komponenten dargestellt wie sie in der React-Anwendung dieser Arbeit wiederzufinden ist. Aus diesen Komponenten ergibt sich der in Abbildung 5.2 dargestellte Aufbau der Benutzeroberfläche.

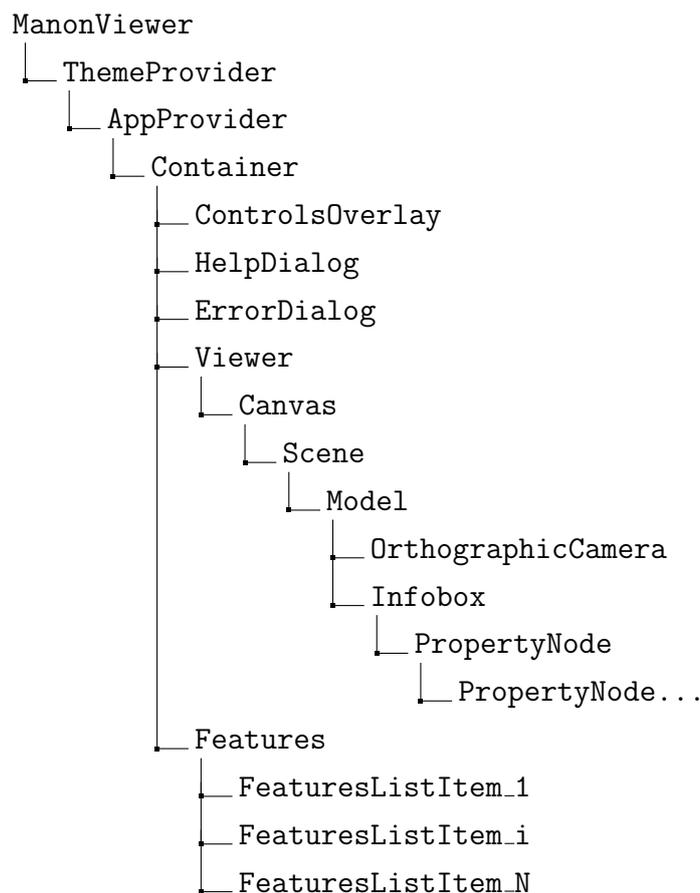


Abbildung 5.1: Verkürzte Hierarchie der React-Komponenten von der Webanwendung dieser Arbeit

Die dynamische Darstellung von Inhalten wird durch die Zustandsverwal-

tung von React möglich. Mithilfe von React Hooks [31] lässt sich der Zustand von Komponenten kontrollieren. Der Zustand wird dann beispielsweise durch Nutzerinteraktionen verändert. Nach jeder Änderung des Zustands wird die Komponente, und damit auch ihre Kind-Komponenten, neu gerendert. Dadurch wird der HTML-Code, und somit das Aussehen der Website modifiziert.

Der dritte wichtige Bestandteil der Anwendung sind React-Kontexte [33]. Sie erlauben es, Daten innerhalb der Komponenten-Hierarchie zu teilen. Insbesondere können alle Kind-Komponenten eines Kontextes auf dessen Daten zugreifen. Ein React-Kontext wird in dieser Arbeit genutzt, um den Zustand der gesamten Anwendung zu verwalten. Das hat den Vorteil, dass alle Komponenten direkt auf den gesamten Zustand zugreifen und diesen modifizieren können. Zudem wird dadurch das aufwendige Weiterreichen von Daten durch die gesamte Komponenten-Hierarchie mittels Eingabe-Parameter überflüssig.

Zusammenfassend basiert die Architektur des Frontends auf dem Next.js Framework, welches grundlegende Funktionalitäten bereitstellt. Die Webanwendung wird auf der Index-Seite angezeigt, welche aus hierarchisch angeordneten Komponenten besteht. Die Verwaltung des globalen Zustands der Anwendung wird mithilfe eines React-Kontextes realisiert.

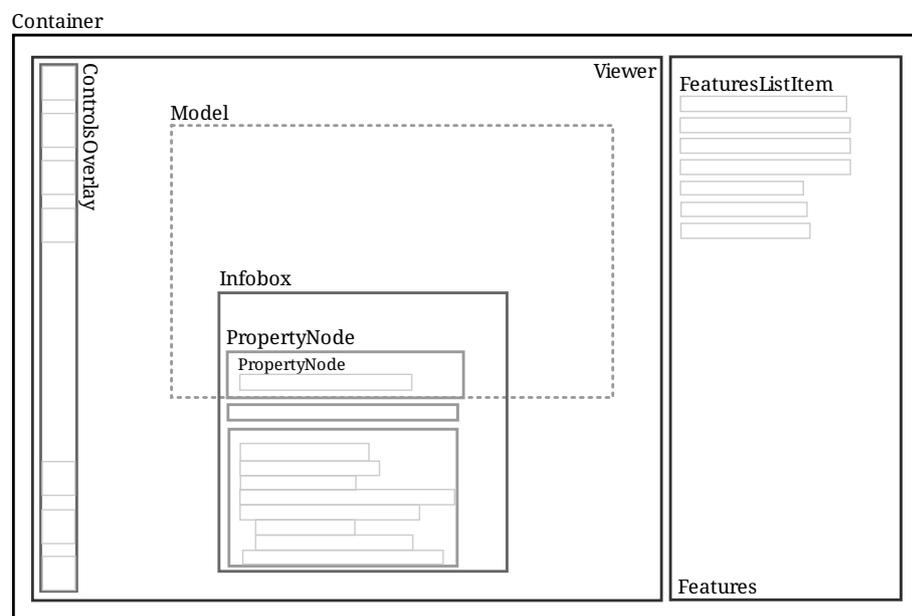


Abbildung 5.2: Aufgliederung der Benutzeroberfläche in ihre einzelnen React-Komponenten

5.2 Styling der Webanwendung

Im vorherigen Abschnitt wurde besprochen, wie Seiten, Komponenten und Kontexte die Grundbausteine der Webanwendung sind. Allerdings spielt neben der Funktionalität der Anwendung auch das Aussehen der grafischen Oberfläche, eine zentrale Rolle für die Benutzbarkeit der Anwendung. Denn nur eine benutzerfreundliche Schnittstelle ermöglicht das effiziente Arbeiten mit der Anwendung. Die optische Gestaltung von Websites wird mit Cascading Style Sheets (CSS)³ realisiert. CSS ist eine Sprache welche, es ermöglicht die Farben, die Schrift und das Layout von Websites zu modifizieren und für verschiedene Geräte anzupassen. Das funktioniert, indem HTML-Elementen gezielt bestimmte Styles zugewiesen werden.

Jedoch unterliegt das manuelle Schreiben von CSS-Dokumenten einigen Herausforderungen [14]. Dazu zählen beispielsweise die oftmals sehr komplexen Selektoren, um bestimmte HTML-Elemente zu selektieren oder das Überschreiben von bestehenden Styles. Es gibt verschiedene CSS Frameworks, die versuchen die verschiedenen Herausforderungen zu lösen. In dieser Arbeit wurde sich für Tailwind CSS⁴ entschieden. Tailwind CSS ist nach Bootstrap⁵ das meistgenutzte CSS Framework und hat die höchste Entwickler-Zufriedenheit [15]. Im Gegensatz zu anderen CSS Frameworks ist es mit Tailwind CSS nicht nötig CSS-Dokumente zu schreiben. Tailwind CSS bietet für alle CSS-Eigenschaften Klassen an, welche im HTML-Code genutzt werden können. Beispielsweise fügt die Klasse `bg-white` auf das zu stylende HTML-Element die CSS-Eigenschaft `background-color: #fff` hinzu.

Zusätzlich wird CSS in dieser Arbeit verwendet, um das Farbschema der Oberfläche zu verwalten. Nutzer haben die Möglichkeit, zwischen einem hellen und einem dunklen Modus zu wechseln. Abhängig vom ausgewählten Modus wird auf den `body` HTML-Element der Website entweder die Klasse `theme-light` oder `theme-dark` angewandt. Diese Klassen beinhalten eine Liste von CSS-Variablen welche die verschiedenen Farbwerte repräsentieren. Ein Ausschnitt dieser CSS-Variablen ist in Tabelle 5.1 zu erkennen. Mithilfe der Erweiterbarkeit von Tailwind CSS können diese Farben durch eigene Klassennamen repräsentiert werden [19]. Durch die in dieser Arbeit verwendete Konfiguration kann einem HTML-Text beispielsweise der Farbwert der `--color-foreground-500` CSS-Variable zugewiesen werden, indem die Klasse `text-500` verwendet wird. Im

³<https://www.w3.org/TR/css-2021>

⁴<https://tailwindcss.com>

⁵<https://getbootstrap.com>

hellen Modus nimmt der Text dann den Farbwert #32302F, und im dunklen Modus, den Farbwert #D6D6D6 an.

CSS-Variable	.theme-light	.theme-dark
--color-background-100	#FAFAFA	#1D1D1D
--color-background-500	#EDED	#2F2F2F
--color-background-900	#DEDEDE	#474747
--color-foreground-100	#969392	#828282
--color-foreground-500	#32302F	#D6D6D6
--color-foreground-900	#0D0D0D	#FFFFFF
--color-primary-100	#FCC5C5	#FCC5C5
--color-primary-500	#F74F4F	#F74F4F
--color-primary-900	#610505	#610505
--color-mesh-face	#D8D5D4	#828282
--color-mesh-face-highlight	#FFA0A0	#ED6161
--color-mesh-face-secondary	#61FFF4	#6EE0DF
--color-mesh-edge	#32302F	#D6D6D6
--color-mesh-edge-highlight	#D30808	#DF8888
--color-mesh-edge-secondary	#0D8BA1	#1F928C

Tabelle 5.1: Ausschnitt der CSS-Variablen von dem in dieser Arbeit genutzten Farbschemas

5.3 Client-Server-Kommunikation

Nachdem Nutzer die Webanwendung geöffnet haben, wird ihnen ein Eingabefeld für Dateien präsentiert. Dort können OWL- und STEP-Dateien mittels “Drag-and-Drop” ausgewählt werden. Andere Dateiformate werden nicht akzeptiert. Um die Dateien weiterzuverarbeiten, müssen sie über das Netzwerk mithilfe des Hypertext Transfer Protocol (HTTP) [5] an den Server gesendet werden. Die Dateien im Payload werden dabei in einem FormData [13] Objekt kodiert. Nachdem die Verarbeitung auf dem Server abgeschlossen ist, wird eine Antwort zurückgeschickt, welche als JavaScript Object Notation (JSON)⁶ kodiert ist. Welche Informationen in dem vom Server zurückgegebenen JSON-Dokument stehen wird im Abschnitt 5.7 beschrieben.

⁶<https://www.json.org>

5.4 Architektur des Servers

Der Server zur Verarbeitung der gesendeten Dateien wurde in dieser Arbeit mit Python⁷ und dem Flask⁸-Framework implementiert. Python in Verbindung mit Flask bietet den Vorteil, dass es sehr einfach ist, einen Web-Server zu erstellen. Dafür sind nur wenige Zeilen Code nötig. Ausschlaggebend für die Wahl von Python als Programmiersprache war auch, dass die Bibliotheken Owlready (siehe Abschnitt 2.1.4) und OCCT (siehe Abschnitt 2.1.3), welche für die Verarbeitung der STEP- und OWL-Dateien benötigt werden, für Python verfügbar sind.

Der Aufbau des Servers ist sehr einfach. Es gibt lediglich eine Schnittstelle, welche die in Abschnitt 5.3 beschriebene HTTP POST-Anfrage verarbeitet. Dabei werden die gesendeten Dateien temporär auf dem Server gespeichert und analysiert. Nach Fertigstellung wird entweder eine Fehlermeldung oder eine erfolgreiche Antwort mit JSON-Daten an die Webanwendung zurückgesendet. Wie genau die Analyse der Dateien aussieht, wird in den folgenden Abschnitten beschrieben.

5.5 Lesen der Eigenschaften

Wie in Abschnitt 2.2.3 beschrieben, werden die von Koehler et al. extrahierten Bauteileigenschaften in einer Ontologie gespeichert. Diese Ontologie wird in Form einer OWL-Datei von den Nutzern bereitgestellt und wie in Abschnitt 5.3 beschrieben, auf den Server hochgeladen. In der Ontologie sind sowohl die Bauteileigenschaften als auch die STEP-Datei repräsentiert. Die sogenannte MANON-Ontologie von Koehler et al. besteht aus mehreren Namespaces. Drei dieser Namespaces werden in diesem Abschnitt verwendet. MANON/ManOn enthält allgemeines Wissen über die Fertigungstechnik, MANON/ManOnSTEP modelliert das STEP-Schema und MANON/Features beinhaltet die verschiedenen Basic und Manufacturing Features.

In diesem Schritt sollen alle Manufacturing Features des Bauteils mit ihren zugehörigen Flächen- und Kanten-Identifikatoren aus der Ontologie gelesen werden. Die Identifikatoren beziehen sich dabei auf die jeweiligen STEP-Entitäten. Dazu werden mithilfe von Owlready zunächst alle in der Ontologie enthaltenen Instanzen der Klasse `Features:ManufacturingFeature` ermittelt. Anschließend müssen die Flächen und Kanten jeder gefundenen

⁷<https://www.python.org>

⁸<https://flask.palletsprojects.com>

Features:ManufacturingFeature-Instanz ermittelt werden. Die Flächen haben eine ManOn:isDefinedBy-Relation, welche die zugehörigen Basic Features referenziert. Diese Features:BasicFeature-Instanzen haben wiederum eine ManOnSTEP:hasMember-Relation, welche alle dazugehörigen STEP-Flächen referenziert. Der Property Path von den Manufacturing Features bis hin zur STEP-Fläche ist schematisch in Abbildung 5.3 dargestellt. Gleichermäßen ist es möglich die Kanten des Bauteils aus der OWL-Datei zu lesen. Dazu werden mithilfe des in Abbildung 5.4 dargestellten Property Paths für jede ManOnSTEP:AdvancedFace-Instanz alle zugehörigen ManOnSTEP:EdgeCurve-Instanzen ermittelt. Die ManOnSTEP:AdvancedFace- und ManOnSTEP:EdgeCurve-Klassen der Ontologie repräsentieren jeweils die ADVANCED_FACE- und EDGE_CURVE-Entität des STEP-Schemas.

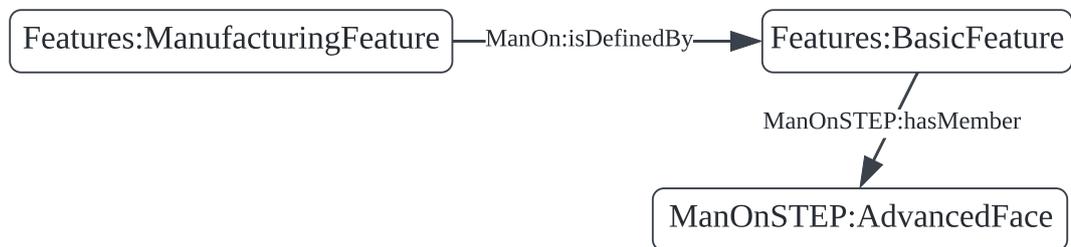


Abbildung 5.3: Property Path vom Features:ManufacturingFeature bis zur ManOnSTEP:AdvancedFace

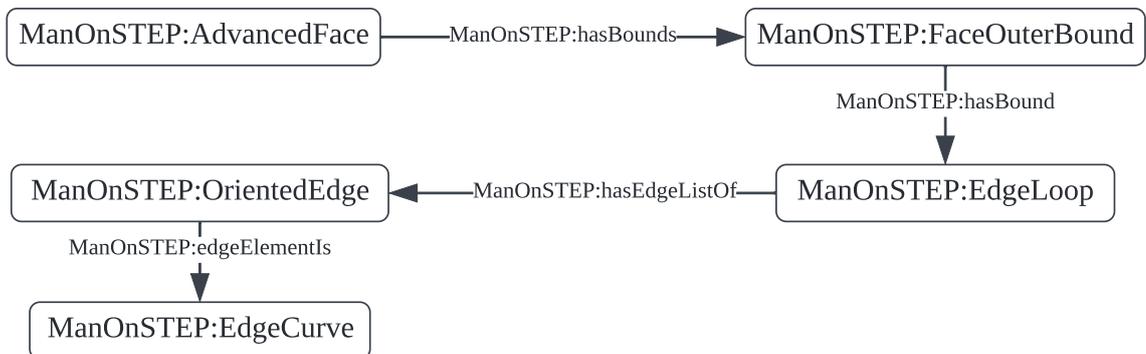


Abbildung 5.4: Property Path von der ManOnSTEP:AdvancedFace bis zur ManOnSTEP:EdgeCurve

Abschließend werden die Identifikatoren der STEP-Entitäten, welche in der URI der Instanz enthalten sind, gespeichert. Dazu werden die Identifikatoren der Flächen- und Kanten-Entitäten in zwei separate Listen geladen und mit den ursprünglichen Manufacturing Features assoziiert. Die so bestimmte Zugehörigkeit von Flächen und Kanten zu Manufacturing Features wird, wie in Abschnitt 5.7 beschrieben, zur Zuordnung von Features und Geometrien benötigt.

5.6 Tesselerung des Brep-Modells

Wie in Abschnitt 2.1.3 beschrieben, modellieren STEP-Dateien Brep-Modelle. Allerdings wird für die Darstellung von 3D-Grafiken eine Mesh mit diskreten Eckpunkten, Kanten und Flächen benötigt. Somit muss das von den Nutzern hochgeladene STEP-Modell in eine Mesh konvertiert werden. Dieser Vorgang wird auch als Tesselerung bezeichnet [30]. In Abbildung 5.5 ist die Eingabe und Ausgabe der Tesselerung an einem Beispiel dargestellt. Dabei wird die mathematische Beschreibung von geometrischen Formen in Dreiecke gegliedert. Je genauer die Mesh dem ursprünglichen Brep-Modell entsprechen soll, desto mehr Dreiecke werden benötigt.

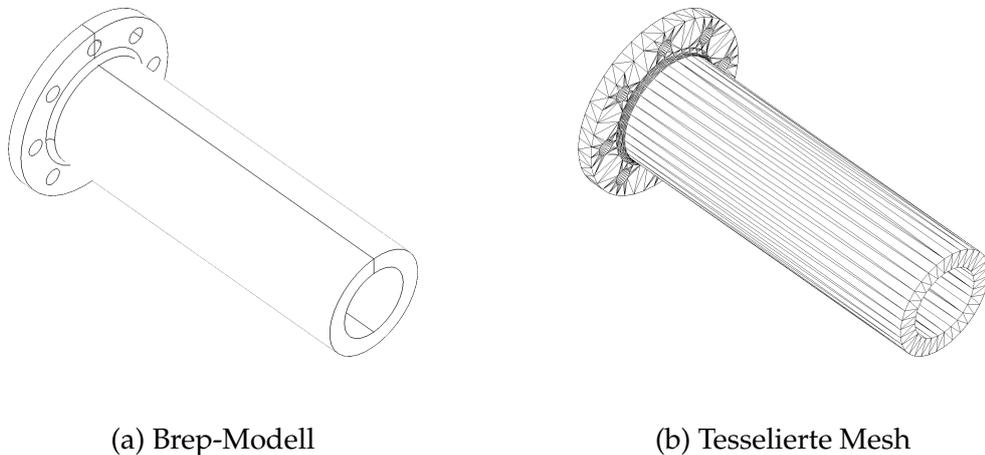


Abbildung 5.5: Beispiel der Tesselerung eines CAD-Bauteils

Zur Tesselerung wird in dieser Arbeit OCCT benutzt. Anfangs wurde die Tesselerung mithilfe von OpenCascade.js⁹ in der Webanwendung implementiert. Allerdings müssen Nutzer dann die gesamte OCCT-Bibliothek herunterladen, was für lange Ladezeiten sorgt. Gleichzeitig ist die Tesselerung ein rechenintensiver Prozess, welcher das Endgerät stark belasten würde. Außerdem ist es der Code besser wartbar, wenn sowohl das Lesen der Eigenschaften (Abschnitt 5.5), als auch die Konvertierung von Brep zu Mesh auf dem Server durchgeführt wird. Deswegen wurde sich für die Implementierung der Tesselerung auf dem Server mittels PythonOCC entschieden.

Üblicherweise wird bei der Tesselerung das gesamte Modell in eine Mesh konvertiert. Um später in der grafischen Darstellung die einzelnen Bauteileigenschaften eindeutig abzugrenzen, wäre es dann nötig, jedes Dreieck der Mesh zu

⁹<https://ocjs.org>

einer Fläche des STEP-Modells zuzuordnen. Für diese Arbeit ist es allerdings einfacher das Modell in mehrere kleinere Meshes zu gliedern. Dann wird jede Fläche des STEP-Modells eindeutig durch eine Mesh repräsentiert und kann später im 3D-Viewer eindeutig abgegrenzt werden. Analog wird mit den Kanten des Brep-Modells vorgegangen. Anstatt jedem tesselierten Kantensegment die Zugehörigkeit zu einer Kante des Brep-Modells zuzuordnen, wird jede Kante des Brep-Modells einzeln tesseliert.

Um eine diskrete Mesh, wie sie in Abbildung 5.5b zu sehen ist, zu modellieren, werden drei Listen benötigt:

Eckpunkte Die Koordinaten von den Eckpunkten der Dreiecke

Dreiecke Die Indizes der Koordinaten, welche zusammen ein Dreieck bilden

Normalen Die Normalenvektoren der Dreiecke

Diese Informationen können mithilfe von PythonOCC ermittelt werden, indem durch jede Fläche des Brep-Modells iteriert wird. Dabei werden die Flächen mit einer selbst gewählten Genauigkeit tesseliert und anschließend deren Eckpunkte, Dreiecke und Normalen ausgelesen. Auch der Identifikator jeder Flächen-Entität des STEP-Modells wird ermittelt. Zusätzlich werden die Kanten von jeder Fläche mithilfe der gleichen Methode tesseliert. Da Kanten lediglich Segmente aus Linien sind, ist es nur notwendig die Eckpunkte zu bestimmen, wozu die Konturen der Flächen nachgezogen werden. Dabei kann es passieren, dass Kanten von zwei aneinandergrenzenden Flächen doppelt auftreten. Um diese Dopplung zu vermeiden, werden bereits hinzugefügte Kanten anhand ihres Hashwertes wiedererkannt und ignoriert. Schlussendlich liefert das beschriebene Vorgehen für jede Fläche und Kante eine diskrete Darstellung, welche in Listen gespeichert vorliegt und für die Darstellung der 3D-Grafik verwendet werden kann.

5.7 Zuordnung von Eigenschaften und Meshes

Das in Abschnitt 5.5 beschriebene Verfahren ermittelt eine Liste von Manufacturing Features und die in Abschnitt 5.6 beschriebene Tessellierung generiert eine Liste von Flächen- und Kanten-Geometrien. In Abbildung 5.6 ist das Schema der vorliegenden Daten dargestellt. Es besteht eine Verknüpfung von den Manufacturing Features aus der Ontologie (`OntologyFeature`) zu den Flächen- und Kanten-Geometrien (`FaceGeometry` und `EdgeGeometry`). Da die Daten in drei verschiedenen Listen gespeichert sind, werden Indizes benötigt, um die Elemente

in konstanter Laufzeit zu referenzieren. Deswegen sollen die Verknüpfungen der Manufacturing Features auf den Index der Flächen- und Kanten-Geometrien zeigen. Für die Webanwendung ist es zudem erforderlich, die Verknüpfung auch in die umgekehrte Richtung zu kennen. Das heißt, für jede Flächen- und Kanten-Geometrie soll auch bekannt sein zu welchem Manufacturing Feature sie gehört. Daraus ergibt sich das erforderliche Schema, welches in Abbildung 5.7 zu sehen ist. Alle Verknüpfungen zeigen dabei auf den Index des jeweiligen Objektes, wodurch die Referenzierung in beide Richtungen in konstanter Laufzeit erfolgen kann.

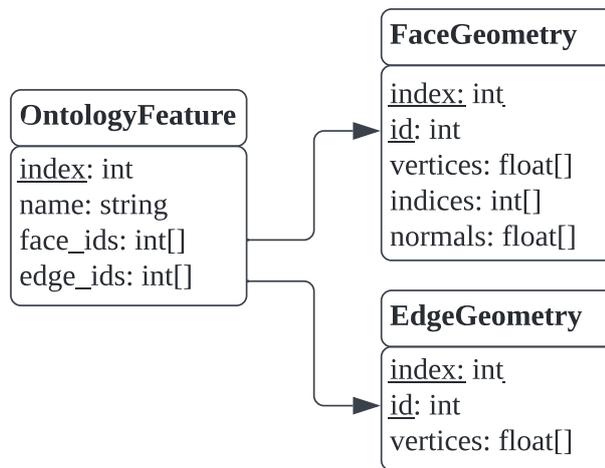


Abbildung 5.6: Schema der unverarbeiteten Daten wie sie nach Abschnitt 5.5 und 5.6 vorliegen

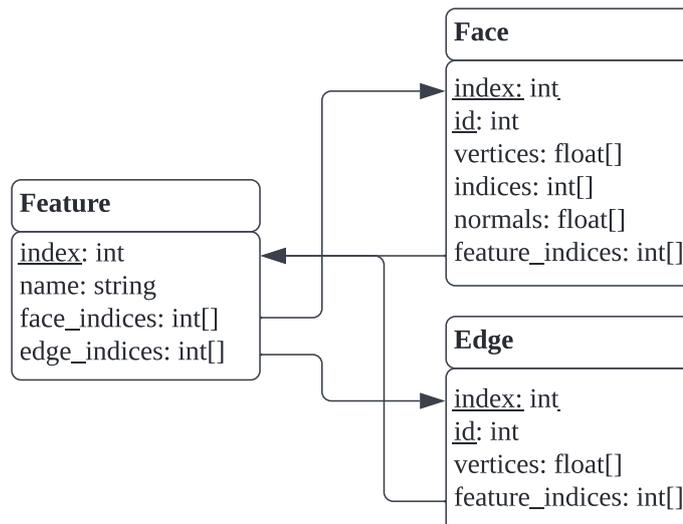


Abbildung 5.7: Schema der zugeordneten Daten, wie sie an die Webanwendung gesendet werden

Um die Daten wie in Abbildung 5.7 umzustrukturieren, sind drei Schritte notwendig. Zuerst wird durch die Liste der Flächen-Geometrien iteriert. Für jedes FaceGeometry-Objekt der Liste wird durch jedes Manufacturing Feature iteriert um nach `face_ids` zu suchen, welche der `id` des FaceGeometry-Objektes entsprechen. Die gefundenen Indizes werden in der `feature_indices`-Liste des Face-Objektes gespeichert. Im zweiten Schritt wird gleichermaßen mit den Kanten-Geometrien vorgegangen. Zuletzt wird durch die Manufacturing Features iteriert, um mittels der `face_ids` und `edge_ids` des `OntologyFeature`-Objektes passende Flächen und Kanten zu finden. Die Treffer werden dann in den `face_indices`- und `edge_indices`-Listen der Feature-Objekte gespeichert. Schlussendlich werden die drei Listen, wie sie in 5.7 zu sehen sind, als JSON serialisiert und auf Anfrage an die HTTP-Schnittstelle der Webanwendung gesendet.

5.8 Interaktiver 3D Viewer

Im Abschnitt 2.2.1 wurde bereits erwähnt, dass für die Entwicklung von 3D-Anwendung im Browser die WebGL-Bibliothek verwendet werden kann. Allerdings ist das Erstellen von 3D-Grafiken mithilfe von WebGL relativ aufwendig und kompliziert. Um diesen Prozess zu vereinfachen, kann eine JavaScript-Bibliothek namens `Three.js`¹⁰ verwendet werden. Diese Bibliothek bietet eine einfache Schnittstelle zu WebGL. So ist es möglich Objekte wie Geometrien, Linien und Kameras hinzuzufügen um komplexe Szenen zu erstellen [12]. Alle Objekte haben eine Position, Skalierung und Rotation im dreidimensionalen Raum. Das Erscheinungsbild von Objekten kann zusätzlich durch Materialien und Lichter verändert werden, indem `Three.js` Schatten- und Lichtinteraktionen simuliert. Darüber hinaus ist es möglich, diese Szenen mit ihren Eigenschaften zu animieren, wodurch auch physikalisch korrekte Bewegungen und Nutzerinteraktionen realisiert werden können. Durch die einfache Nutzung und hohe Flexibilität von `Three.js` ist es ohne WebGL Kenntnisse möglich, komplexe 3D-Grafiken im Browser zu erstellen. Deswegen wird `Three.js` in dieser Arbeit verwendet.

Nachdem die in Abbildung 5.7 dargestellten Daten als HTTP-Antwort vom Server an die Webanwendung zurück gesendet wurden, können die Flächen und Kanten in den 3D-Viewer geladen werden. Für die Flächen wird dazu eine `BufferGeometry` mit den Eckpunkten, Normalen und Dreiecken genutzt. Die `BufferGeometry`-Klasse ermöglicht die Modellierung von beliebigen Meshes. Die

¹⁰<https://threejs.org>

Kanten werden hingegen durch `LineSegments`-Objekte repräsentiert, welche nur die Eckpunkte als Eingabe benötigen. `LineSegments` modellieren Linien im dreidimensionalen Raum. Währenddessen wird den Flächen und Kanten mithilfe von Materialien ihr initiales Aussehen zugewiesen. Zudem wird die initiale Kamera so positioniert, dass das gesamte Objekt dargestellt wird. Für das in Abbildung 5.5a gezeigte Modell sieht der 3D-Viewer anfangs wie in Abbildung 5.8 dargestellt aus.

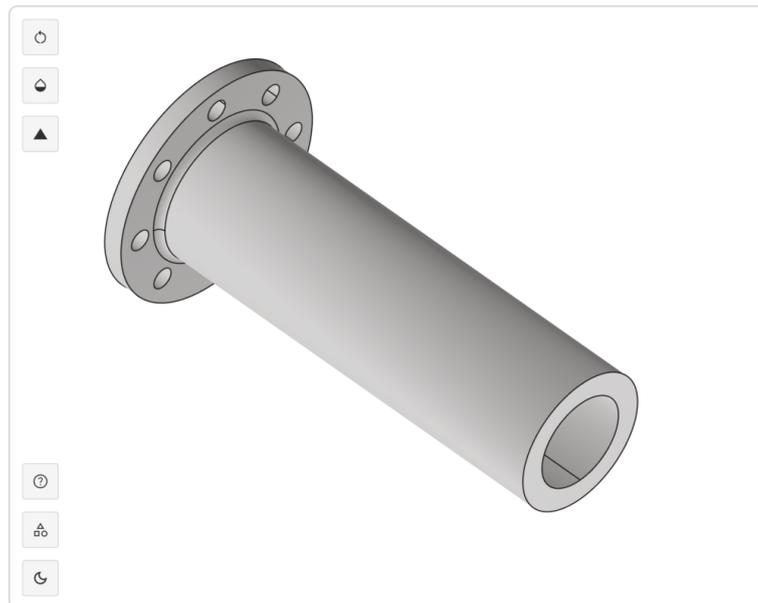


Abbildung 5.8: Initiales Aussehen des 3D-Viewers nach dem Laden eines Beispiel-Modells

In diesem Zustand können Nutzer Manufacturing Features durch Klicken auf Flächen oder Kanten auswählen. Ausgewählte Features werden wie in Abbildung 5.9 rot markiert. Zudem können Nutzer eine Liste aller Manufacturing Features anzeigen lassen. Von dort können die Features ebenso ausgewählt werden. In Abbildung 5.9 wurde auf eine Kante geklickt, welche sich zwischen zwei Manufacturing Features befindet. Dadurch werden alle Features markiert die zu dieser Kante gehören. Die rot markierten Flächen stellen das ausgewählte Feature dar und die blau markierten Flächen gehören zu anderen Features, welche auch die angeklickte Kante beinhalten. In der Mitte der Abbildung 5.9 ist zudem eine Infobox mit Daten aus der Ontologie zu sehen. Diese Infobox wird in Abschnitt 5.9 genauer beschrieben.

Weiterhin gibt es sechs verschiedene Buttons auf der linken Seite des 3D-Viewers, welche ebenso in Abbildung 5.9 zu sehen sind. Von oben nach unten, setzt der erste Button die Kamera in den ursprünglichen Zustand zurück. Der

zweite Button sorgt dafür, dass die Flächen des Modells leicht transparent werden. Dadurch ist es beispielsweise möglich, komplett eingeschlossene Manufacturing Features zu erkennen. Der dritte Button blendet die Kanten des Bauteils ein oder aus. Mithilfe des vierten Buttons kann ein Hilfe-Dialog angezeigt werden, welcher den Nutzern die Bedienung des 3D-Viewers erklärt. Der fünfte Button zeigt oder versteckt die Liste der Manufacturing Features auf der rechten Seite der Anwendung. Der sechste Button wechselt zwischen dem hellen und dem dunklen Modus der Oberfläche.

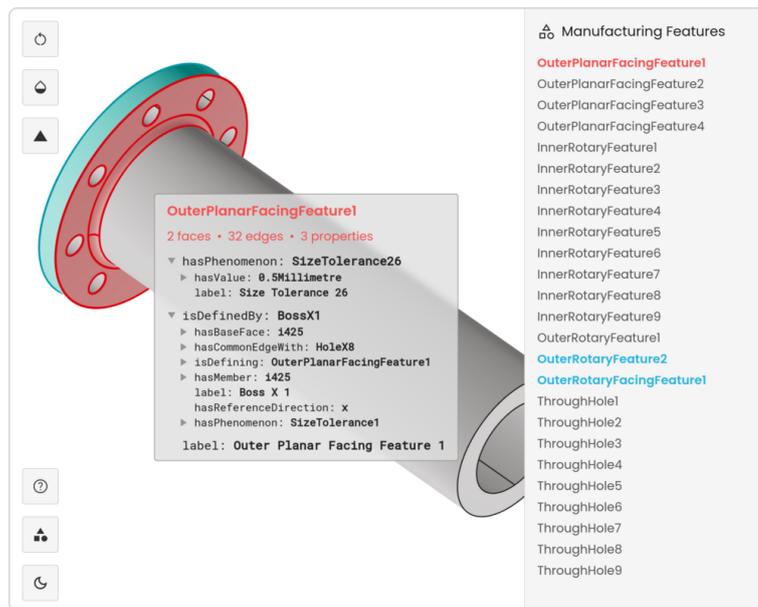


Abbildung 5.9: Aussehen des 3D-Viewers nach dem Klicken auf eine Kante, welche zwischen zwei Manufacturing Features liegt

5.9 Informationen aus der Ontologie

Neben dem Ermitteln der Manufacturing Features und der Tessellierung des STEP-Modells lädt der Python Server zusätzliche Informationen aus der hochgeladenen OWL-Ontologie. Für jede Instanz der Manufacturing Features werden dessen Relationen ermittelt und an die Webanwendung gesendet. Allerdings werden nicht nur die direkten Relationen geladen, sondern auch tiefere Verknüpfungen. Dazu wird der rekursive Algorithmus 1 für jede Instanz instance der Manufacturing Features aufgerufen.

Die Parameter `depth` und `max_depth` beinhalten die derzeitige und die maximale Tiefe des Property Paths. Deswegen wird die Prozedur `GET_PROPS` für jedes Feature mit einer `depth` von 0 ausgeführt. Die Rekursion wird abgebrochen, so-

bald die derzeitige Tiefe größer gleich der maximalen Tiefe ist. Die maximale Tiefe wurde in dieser Arbeit auf den Wert drei gesetzt. Falls die Abbruchbedingung nicht ausgeführt wird, werden die Relationen der Instanz iteriert. Für jede Relation prop wird dann deren Wert value geladen. Falls der Wert eine andere Instanz ist, wird der Algorithmus rekursiv mit dieser Instanz ausgeführt. Andernfalls wird lediglich der Wert gespeichert. Am Ende gibt die Prozedur die gefundenen Relationen mit ihren Werten und Kindern zurück.

Das dabei entstandene Python-Objekt wird mithilfe von Flask in ein JSON-Objekt konvertiert und an die Webanwendung gesendet. Die Webanwendung zeigt dann die Relationen mit ihren Werten wie in Abbildung 5.9 an. Die einzelnen Relationen mit ihren Kind-Instanzen sind dabei an den grauen Dreiecken aufklappbar und wurden mithilfe einer rekursiven React-Komponente umgesetzt.

Algorithmus 1 Rekursiver Algorithmus zur Ermittlung von Relationen einer Instanz in der Ontologie

```

procedure GET_PROPS(instance, depth, max_depth)
  if depth  $\geq$  max_depth then
    return []
  end if
  properties  $\leftarrow$  []
  for prop in instance.GET_PROPERTIES() do
    node  $\leftarrow$  { 'name': prop.name, 'value': Null, 'children': [] }
    for value in prop[instance] do
      if HASATTR(value, 'get_properties') then ▷ Hat Kinder
        node['value']  $\leftarrow$  value.name
        node['children']  $\leftarrow$  GET_PROPS(value, depth + 1, max_depth)
      else ▷ Blattknoten
        node['value']  $\leftarrow$  value
      end if
    end for
    properties.APPEND(node)
  end for
  return properties
end procedure

```

5.10 Deployment der Anwendung

Für die Ausführung der Anwendung sind verschiedene Abhängigkeiten notwendig. Um Versionskonflikte und Installationsprobleme auf unterschiedlichen Betriebssystemen zu vermeiden, wird die Anwendung in plattformun-

abhängige Umgebungen gebündelt. Das wird mithilfe von Docker¹¹ möglich. Mit Docker können Anwendungen einheitlich virtualisiert und somit auf allen weitverbreiteten Betriebssystemen ausgeführt werden. Die dabei entstandenen Docker Images können dann ohne Problem in einem Produktivsystem eingesetzt werden. Deswegen wird in dieser Arbeit sowohl die Webanwendung als auch der Python-Server mittels einer Docker-Umgebung ausgeführt. Das Docker Image der Webanwendung benutzt dabei das Verwaltungssystem NPM¹² um JavaScript-Bibliotheken mit spezifischen Versionen zu installieren. Bei der Serveranwendung wird dafür Conda¹³ verwendet.

¹¹<https://docker.com>

¹²<https://www.npmjs.com>

¹³<https://anaconda.org>

6 Evaluation

In diesem Kapitel wird die Anwendung, welche aus der Implementierung des vorherigen Kapitels hervorgeht, auf ihre Qualität geprüft. Dazu wird erst die Evaluationsmethodik vorgestellt und anschließend deren Ergebnisse präsentiert.

6.1 Methodik

Zur Evaluation der Anwendung wird die Vollständigkeit und die Benutzbarkeit der Anwendung untersucht. Die Vollständigkeit der Anwendung signalisiert, wie gut die Anforderungen erfüllt wurden und die Benutzbarkeit wie effizient mit der Anwendung gearbeitet werden kann. Deswegen wird zunächst validiert, inwiefern die in Kapitel 4 vorgestellten Anforderungen durch die entwickelte Anwendung erfüllt werden:

- A1** Zur Prüfung der Plattformunabhängigkeit wird die Webanwendung in drei verschiedenen Web-Browsern (Chromium¹, Mozilla Firefox² und Microsoft Edge³) und auf zwei Betriebssystem (Linux und Windows) getestet.
- A2** Die intuitive Bedienung der Anwendung wird argumentativ evaluiert.
- A3** Die Einbindung einer externen Software zur Detektion von Bauteileigenschaften wird argumentativ evaluiert.
- A4** Um zu überprüfen ob die Endgeräte möglichst stark entlastet werden, wird die Auslastung der CPU, GPU und des Arbeitsspeichers gemessen.

¹<https://www.chromium.org>

²<https://www.mozilla.org/firefox>

³<https://www.microsoft.com/edge>

A5, A6 Dass das Bauteil korrekt dreidimensional dargestellt und von allen Seiten betrachtet werden kann, wird überprüft indem der erstellte 3D-Viewer mit dem Viewer von FreeCAD verglichen wird.

A7, A8, A9 Um das Auswählen von Bauteileigenschaften sowohl im 3D-Viewer als auch von einer Liste zu überprüfen, wird ein Test-Modell herangezogen, welches alle Manufacturing Features beinhaltet. Dadurch kann getestet werden, ob jede Eigenschaft korrekt auswählbar ist.

A10 Die Erkennbarkeit von eingeschlossenen Eigenschaften wird anhand eines Beispiel-Modells getestet.

A11 Die Darstellung der Details zu den Eigenschaften aus der Ontologie wird argumentativ und anhand eines Test-Modells evaluiert.

A12 Das plattformunabhängige Hosting der Anwendung wird argumentativ evaluiert.

Neben den Anforderungen aus Kapitel 4 ist auch die Performanz der Anwendung relevant für die Benutzbarkeit. Deswegen wird anhand drei verschiedener Bauteile, welche alle unterschiedlich komplex sind, die Laufzeit der implementierten Methoden gemessen und verglichen. Gemessen wird sowohl die gesamte Wartezeit für den Nutzer und dessen Verteilung auf die verschiedenen Prozesse.

6.2 Ergebnisse

In diesem Abschnitt wird die Evaluationsmethodik aus Abschnitt 6.1 umgesetzt und dessen Ergebnisse präsentiert.



Abbildung 6.1: Die in dieser Arbeit entwickelte Anwendung getestet mit einem Beispiel-Bauteil in drei verschiedenen Web-Browsern auf Linux

Die Plattformunabhängigkeit (**A1**) wurde durch das Testen von drei modernen Web-Browsern bestätigt. Wie in Abbildung 6.1 zu sehen ist, funktioniert

die Webanwendung in jedem der drei getesteten Web-Browsern problemlos. Das gleiche Resultat konnte sowohl auf Linux als auch auf Windows festgestellt werden. Außerdem gibt es keine Unterschiede im Aussehen der Oberfläche.

Die intuitive Bedienung der Anwendung (A2) wird durch dessen einfache Benutzeroberfläche realisiert. Nutzer müssen lediglich zwei Dateien in die Anwendung laden und können anschließend das Bauteil mit seinen Eigenschaften durch Klicken auf das Modell analysieren. Außerdem wird die Bedienung kurz in einem Hilfe-Dialog erklärt.

Eine externe Software für die Extraktion der Bauteileigenschaften (A3) muss vorliegen, da Nutzer sonst keine Möglichkeit haben die Ontologie zu generieren. In dieser Arbeit wird dafür das in Abschnitt 2.2.3 beschriebene Verfahren vom Deutschen Zentrum für Luft- und Raumfahrt e. V. genutzt [26].

Um das Endgerät möglichst stark zu entlasten (A4) wurden die Tessellierung und das Auslesen der Ontologie auf die Serveranwendung verlagert. Außerdem wurde die Webanwendung, welche auf dem Endgerät ausgeführt wird, möglichst ressourcenschonend hinsichtlich CPU, GPU und Arbeitsspeicher entwickelt. Die Webanwendung verwaltet lediglich Daten, welche absolut notwendig sind und der 3D-Viewer wird nur neu gerendert, wenn eine Veränderung geschah. Um die Entlastung des Endgeräts zu verifizieren wurde ein Performanz-Profil mithilfe der Chrome DevTools⁴ generiert, welches in Abbildung 6.2 zu sehen ist. Das Profil wurde für ungefähr sechs Sekunden aufgezeichnet. In den ersten 1800 ms wurden die OWL- und STEP-Dateien für ein Beispiel-Modell via Drag-and-Drop hochgeladen und auf dem Server verarbeitet. Nachdem der Server das Resultat der Verarbeitung zurückgesendet hat, gab es bis 2400 ms keine Interaktion. Daraufhin wurde die Kamera des 3D-Viewers bis 3500 ms verschoben. Bei ca. 4200 ms wurde ein Manufacturing Feature in dem 3D-Viewer ausgewählt, woraufhin es keine Interaktionen mit der Anwendung gab. Die CPU-Auslastung ist ganz oben in Abbildung 6.2 in gelb zu sehen und ist zu fast allen Zeitpunkten sehr gering. Lediglich nach der Antwort des Servers, während das Modell in den 3D-Viewer geladen wird, ist eine starke Auslastung zu erkennen. Das blaue Flächendiagramm in Abbildung 6.2 zeigt die Größe des verwendeten Arbeitsspeichers. Anfangs werden 8.5 MB verwendet und nach dem Laden des Modells maximal 15.6 MB. Ganz unten ist in Abbildung 6.2 die Auslastung der GPU zu erkennen. Dabei fällt auf, dass die GPU hauptsächlich dann ausgelastet ist, wenn die Kamera des 3D-Viewer verändert wird. Somit stellt die Webanwendung während der Nutzung keine hohen Anforderungen an die Rechenleistung

⁴<https://developer.chrome.com/docs/devtools>

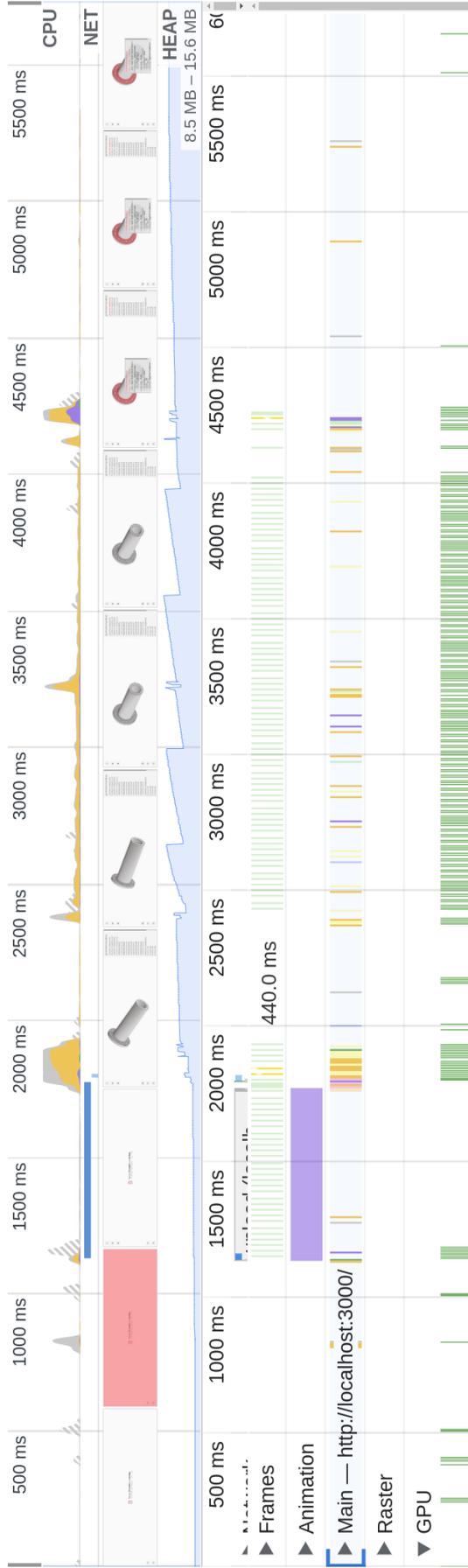


Abbildung 6.2: Performanz-Profil der Anwendung bei standardmäßiger Nutzung

des Endgeräts.

Die Qualität der dreidimensionalen Darstellung des Bauteils (A5, A6) wird mithilfe von Abbildung 6.3 evaluiert. Dazu wurde ein Beispiel-Bauteil sowohl in FreeCAD, als auch in die entwickelte Anwendung geladen. Beide Modelle sehen nahezu identisch aus. Lediglich die Platzierung von manchen Kanten weicht voneinander ab. So ist beispielsweise auf der Oberseite des Rohres in Abbildung 6.3a eine Kante zu erkennen, welche in Abbildung 6.3b nicht zu sehen ist. In diesem Fall befindet sich die Kante in Abbildung 6.3b auf der Unterseite des Rohres. Diese Abweichungen können mit der Implementierung der Tessellierung zusammenhängen. Zudem ist es in beiden Anwendungen möglich, die Kamera des 3D-Viewers beliebig zu transformieren.

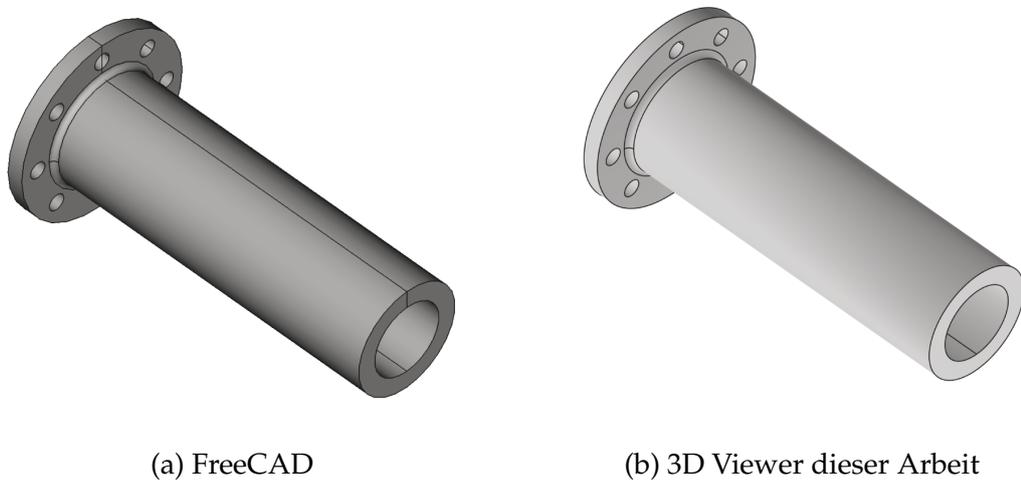


Abbildung 6.3: Vergleich zwischen dem 3D Viewer von FreeCAD und dieser Arbeit anhand eines Beispiel-Modells

In Abbildung 6.4 ist ein Bauteil zu erkennen, welches eine Vielzahl an Manufacturing Features besitzt. Im Zuge des manuellen Testes wurde jedes der 67 Manufacturing Features sowohl durch Klicken auf die Liste der Features als auch durch Klicken auf die Flächen des 3D-Modells ausgewählt (A7, A8, A9). Gleichzeitig wurde stichprobenartig getestet ob durch Klicken auf eine Kante, welche zwischen zwei Manufacturing Features liegt, mehrere Features markiert werden. Beide Tests haben wurden ohne Fehler absolviert. Gleichzeitig wurden die im 3D-Viewer angezeigten Daten der Ontologie (A11) stichprobenartig mit den Daten der OWL-Datei verglichen. Auch dieser Test ergab eine 100%-ige Übereinstimmung.

Von anderen Flächen komplett eingeschlossene Manufacturing Features (A10) können, wie in Abbildung 6.5 dargestellt, durch Aktivieren des transparenten

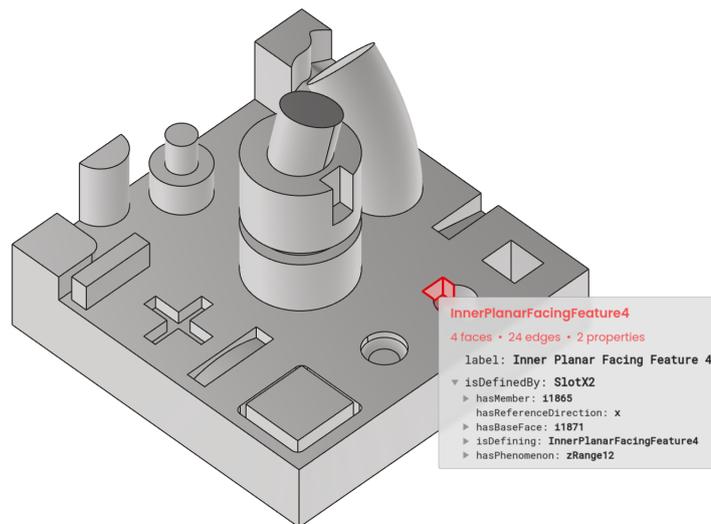


Abbildung 6.4: Beispiel-Bauteil mit einer Vielzahl von Manufacturing Features

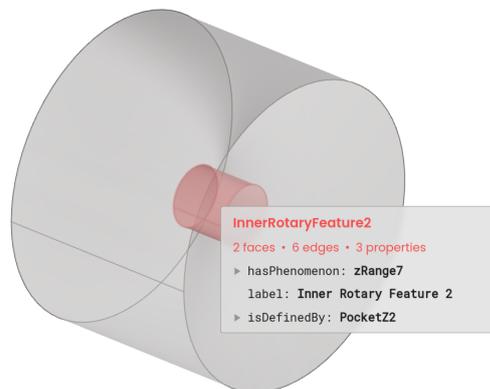


Abbildung 6.5: Beispiel eines Manufacturing Features, welches komplett von anderen Flächen eingeschlossen ist

Modus betrachtet werden.

Das plattformunabhängige Hosting (A12) wird, wie in Abschnitt 5.10 beschrieben, durch die Nutzung von Docker ermöglicht.

Schlussendlich wurde auch die Performanz der Anwendung getestet. Verwendet wurden dazu die in Abbildung 6.6 dargestellten Bauteile. Die Resultate der Messung sind in Tabelle 6.1 zu sehen. Es ist zu erkennen, dass mit zunehmender Anzahl an Features und Größe des STEP-Modells auch die Ladezeit steigt. Die meiste Zeit wird für das Laden der Daten aus der Ontologie benötigt. Trotzdem ist die Wartezeit für das komplexeste Bauteil lediglich eine Sekunde. Allerdings ist die Performanz nicht nur von der Komplexität des Bauteils abhängig, sondern auch von zwei im Code festgelegten Parametern. Der erste Parameter bestimmt die Genauigkeit der Tessellierung durch die lineare Abweichung [45].

In dieser Arbeit wurde dafür ein Wert von 0.05 Maßeinheiten gewählt. Bei den gewählten Beispielen bedeutet das eine maximale lineare Abweichung von 0,05 mm. Je niedriger die lineare Abweichung gewählt wird, desto länger dauert die Tessellierung. Der zweite Parameter ist die Tiefe des Property Paths beim rekursiven Laden der Daten aus der Ontologie. In dieser Arbeit wurde dafür ein Wert von drei gewählt. Je höher dieser Wert, desto länger werden die Ladezeiten sein.



(a) hidden_feature.stp (b) flange_round_straight.stp (c) Probe_01.stp

Abbildung 6.6: Zur Evaluation der Performanz genutzte Bauteile

Modell	6.6a	6.6b	6.6c
Anzahl an Features	13	25	67
Größe des STEP-Modells	14 kB	90 kB	177 kB
Größe des Uploads	120 kB	431 kB	1731 kB
Größe der Antwort	97 kB	858 kB	979 kB
Ladezeit insgesamt	437 ms	665	1011
Netzwerk-Uploads	0,4 ms	1 ms	3,7 ms
Abfragen der Ontologie	411 ms	494 ms	701 ms
Tessellierung	14 ms	131 ms	229 ms
Zuordnung	0 ms	1 ms	32 ms
Netzwerk-Downloads	1 ms	1,8 ms	1,7 ms

Tabelle 6.1: Ergebnisse der Performanz-Messungen von den Bauteilen aus Abbildung 6.6

7 Fazit

Das Ziel dieser Arbeit war es, einen Ontologien-basierten Ansatz für die grafische Darstellung von fertigungsrelevanten Bauteileigenschaften zu entwickeln. Das Konzept der Anwendung entstand auf Basis von zwölf Anforderungen. Daraus resultierte die Entwicklung einer Anwendung mit Client-Server-Architektur. Die Benutzeroberfläche ist dabei sehr benutzerfreundlich aufgebaut und hat im Zentrum einen 3D-Viewer. In diesem 3D-Viewer wird das Bauteil mit seinen Eigenschaften präsentiert, sodass Nutzer die Eigenschaften effizient analysieren können.

Zur Verknüpfung der Bauteileigenschaften mit der 3D-Representation des Bauteils war es nötig, die von den Nutzern hochgeladenen Dateien zu analysieren. Die Informationen zu den Eigenschaften sind dabei in einer Ontologie wiederzufinden und die Geometrie des Bauteils in einem STEP-Modell. Zusätzlich war es nötig, das STEP-Modell in eine diskrete Darstellung des 3D-Modells zu konvertieren. Diese diskrete Darstellung wurde benötigt um das Bauteil grafisch zu rendern. Abschließend wurden Vollständigkeit und Benutzbarkeit der entwickelten Anwendung evaluiert.

Auch wenn die Evaluierung der Anwendung und ihrer Performanz größtenteils positive Ergebnisse aufzeigte, gibt es noch einige Möglichkeiten zur Verbesserung und Erweiterung des entwickelten Ansatzes. Beispielsweise wäre es für die Nutzer hilfreich, wenn sie die Ontologie und damit auch das Bauteil selbst, direkt in der Anwendung modifizieren könnten. Weiterhin würde die Integration von weiteren Product Manufacturing Information (PMI), wie dem Material, der Historie des Modells oder der Oberflächenbehandlung Sinn machen. Dadurch hätten Nutzer ein besseres Verständnis des herzustellenden Bauteils. Außerdem könnte die Benutzbarkeit der Anwendung vereinfacht werden, wenn die Extraktion der Bauteil-Eigenschaft direkt, ohne die Verwendung einer externen Software, erfolgen würde. Dazu müsste die Extraktion der Ei-

enschaften entweder in der Serveranwendung dieser Arbeit erfolgen, oder der 3D-Viewer dieser Arbeit in die Software der Extraktion eingebettet werden.

Allerdings gehen die genannten Erweiterungsmöglichkeiten über den Rahmen dieser Arbeit hinaus. Diese könnten aber als Grundlage für anknüpfende Arbeiten verwendet werden. Schlussendlich wurde das Ziel, die Entwicklung eines CAD-Viewers, welcher durch Ontologien repräsentierte Bauteileigenschaften visualisieren kann, erreicht.

Literaturverzeichnis

- [1] AP 242. *AP 242 Ed2*. 2014. URL: <http://www.ap242.org/edition-2> (besucht am 10.06.2022).
- [2] LEO ALTING und HONGCHAO ZHANG. „Computer Aided Process Planning: the state-of-the-art survey“. In: *International Journal of Production Research* 27.4 (Apr. 1989), S. 553–585. DOI: 10.1080/00207548908942569.
- [3] Edward Angel und Dave Shreiner. *Interactive Computer Graphics with WebGL*. Addison-Wesley Professional, Mai 2016. ISBN: 9781292019338.
- [4] Knut Erik Bang und Tore Markeset. „Impact of Globalization on Model of Competition and Companies' Competitive Situation“. In: *Advances in Production Management Systems. Value Networks: Innovation, Technologies, and Management*. Springer Berlin Heidelberg, 2012, S. 276–286. DOI: 10.1007/978-3-642-33980-6_32.
- [5] M. Bishop und Ed. Akamai. *RFC 9114 HTTP/3*. 2022. URL: <https://www.rfc-editor.org/rfc/rfc9114> (besucht am 14.09.2022).
- [6] Open Cascade. *Open CASCADE Technology 7.6.0*. 2022. URL: <https://dev.opencascade.org/doc/occt-7.6.0/overview/html/index.html> (besucht am 22.08.2022).
- [7] Paolo Cignoni u. a. *MeshLab: an Open-Source Mesh Processing Tool*. en. 2008. DOI: 10.2312/LOCALCHAPTEREVENTS/ITALCHAP/ITALIANCHAPCONF2008/129-136.
- [8] World Wide Web Consortium. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 2012. URL: <https://www.w3.org/TR/owl2-overview/> (besucht am 10.06.2022).
- [9] World Wide Web Consortium. *RDF Schema 1.1*. 2014. URL: <https://www.w3.org/TR/rdf-schema/> (besucht am 06.08.2022).

- [10] Theo D’Hondt, Hrsg. *ECOOP 2010 – Object-Oriented Programming*. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-14107-2.
- [11] Leonard Daly und Don Brutzman. „X3D: Extensible 3D Graphics Standard [Standards in a Nutshell]“. In: *IEEE Signal Processing Magazine* 24.6 (Nov. 2007), S. 130–135. DOI: 10.1109/msp.2007.905889.
- [12] Jos Dirksen. *Three.js essentials*. Packt Publishing, 2014. ISBN: 978-1-78398-087-1.
- [13] Mozilla Foundation. *FormData*. URL: <https://developer.mozilla.org/en-US/docs/Web/API/FormData> (besucht am 14.09.2022).
- [14] Micah Godbolt. *Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites*. O’Reilly Media, 2016. ISBN: 9781491926789.
- [15] Sacha Greif. *The State of CSS 2021: CSS Frameworks*. 2021. URL: <https://2021.stateofcss.com/en-US/technologies/css-frameworks/> (besucht am 14.09.2022).
- [16] Khronos Group. *WebGL 2 Specification*. 2017. URL: <https://registry.khronos.org/webgl/specs/2.0> (besucht am 23.08.2022).
- [17] Khronos Group. *WebGL Specification*. 2014. URL: <https://registry.khronos.org/webgl/specs/1.0> (besucht am 30.08.2022).
- [18] Nicola Guarino, Daniel Oberle und Steffen Staab. „What Is an Ontology?“ In: *Handbook on Ontologies*. Springer Berlin Heidelberg, 2009, S. 1–17. DOI: 10.1007/978-3-540-92673-3_0.
- [19] Tailwind Labs Inc. *Customizing Colors - Tailwind CSS*. URL: <https://tailwindcss.com/docs/customizing-colors> (besucht am 14.09.2022).
- [20] Taufan Fadhilah Iskandar u. a. „Comparison between client-side and server-side rendering in the web development“. In: *IOP Conference Series: Materials Science and Engineering* 801.1 (Mai 2020), S. 012136. DOI: 10.1088/1757-899x/801/1/012136.
- [21] *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*. Standard. International Organization for Standardization, Nov. 2004. URL: <https://www.iso.org/standard/38047.html> (besucht am 07.09.2022).
- [22] Richard Jones, Hrsg. *ECOOP 2014 – Object-Oriented Programming*. Springer Berlin Heidelberg, 2014. DOI: 10.1007/978-3-662-44202-9.

- [23] Arttu Julin u. a. „Characterizing 3D City Modeling Projects: Towards a Harmonized Interoperable System“. In: *ISPRS International Journal of Geo-Information* 7.2 (Feb. 2018), S. 55. DOI: 10.3390/ijgi7020055.
- [24] Sharon J Kemmerer und Sharon J Kemmerer. *STEP, the grand experience*. Techn. Ber. 1999. DOI: 10.6028/nist.sp.939.
- [25] Tobias Köhler u. a. „Development of a Methodology for the Digital Representation of Manufacturing Technology Capabilities“. en. In: (2021). DOI: 10.15488/11259.
- [26] Tobias Köhler u. a. „Geometric Feature Extraction for Additive Manufacturing Parts based on a Knowledge Graph“. Proceeding.
- [27] David Krizaj und Nikola Vukasinovic. „Analysis of Implementation of MBD and STEP AP242 Standard into Modern CAD Tools“. In: *Res. & Sci. Today* 17 (2019), S. 70. URL: <https://www.rstjournal.com/wp-content/uploads/2019/07/MIT-supplement-1-2019.pdf#page=70> (besucht am 22.08.2022).
- [28] Jean-Baptiste Lamy. „Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies“. In: *Artificial Intelligence in Medicine* 80 (Juli 2017), S. 11–28. DOI: 10.1016/j.artmed.2017.07.002.
- [29] David Loffredo. „Fundamentals of STEP implementation“. In: (1999). URL: <https://www.steptools.com/stds/step/fundimpl.pdf> (besucht am 11.07.2022).
- [30] Wang Lu u. a. „Robust and Fast CAD Model Tessellation for Inspection“. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), S. 1–14. DOI: 10.1109/tim.2022.3156988.
- [31] Inc. Meta Platforms. *Hooks FAQ*. URL: <https://reactjs.org/docs/hooks-faq.html> (besucht am 13.09.2022).
- [32] Inc. Meta Platforms. *React Component Documentation*. URL: <https://reactjs.org/docs/react-component.html> (besucht am 13.09.2022).
- [33] Inc. Meta Platforms. *React Context Documentation*. URL: <https://reactjs.org/docs/context.html> (besucht am 13.09.2022).
- [34] Muhammed Oguzhan Mete, Dogus Guler und Tahsin Yomralioglu. „Development of 3D Web GIS Application with Open Source Library“. In: *Selcuk University Journal of Engineering, Science and Technology* 6 (Dez. 2018). DOI: 10.15317/scitech.2018.171.

- [35] Mark A. Musen. „The protégé project“. In: *AI Matters* 1.4 (Juni 2015), S. 4–12. DOI: 10.1145/2757001.2757003.
- [36] Michael A. Park u. a. „Boundary Representation Tolerance Impacts on Mesh Generation and Adaptation“. In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics und Astronautics, Juli 2021. DOI: 10.2514/6.2021-2992.
- [37] Marco Potenziani, Marco Callieri und Roberto Scopigno. „Developing and Maintaining a Web 3D Viewer for the CH Community: an Evaluation of the 3DHOP Framework“. In: *Eurographics Workshop on Graphics and Cultural Heritage* (2018). DOI: 10.2312/GCH.20181356.
- [38] Vinod V Rampur und Dr. Sachhidanand Reur and. „Computer Aided Process Planning using STEP Neutral File for Automotive Parts“. In: *International Journal of Engineering Research and V6.04* (Apr. 2017). DOI: 10.17577/ijertv6is040497.
- [39] Vahid Salehi und Shirui Wang. „Web-Based Visualization of 3D Factory Layout from Hybrid Modeling of CAD and Point Cloud on Virtual Globe DTX Solution“. In: *Computer-Aided Design and Applications* 16.2 (Aug. 2018), S. 243–255. DOI: 10.14733/cadaps.2019.243-255.
- [40] William J. Schroeder und Kenneth M. Martin. „The Visualization Toolkit“. In: *Visualization Handbook*. Elsevier, 2005, S. 593–614. DOI: 10.1016/b978-012387582-2/50032-0.
- [41] Yang Shi u. a. „A Critical Review of Feature Recognition Techniques“. In: *Computer-Aided Design and Applications* 17.5 (Jan. 2020), S. 861–899. DOI: 10.14733/cadaps.2020.861-899.
- [42] Bruno Simoes, Mara del Puy Carretero und Jorge Martnez Santiago. „Photorealism and Kinematics for Web-based CAD data“. In: *The 25th International Conference on 3D Web Technology*. ACM, Nov. 2020. DOI: 10.1145/3424616.3424710.
- [43] Sergey E. Slyadnev, Alexander Malyshev und Vadim Turlapov. „CAD model inspection utility and prototyping framework based on OpenCascade“. In: (2017). URL: <https://dev.opencascade.org/publication/cad-model-inspection-utility-and-prototyping-framework-based-opencascade> (besucht am 30.08.2022).

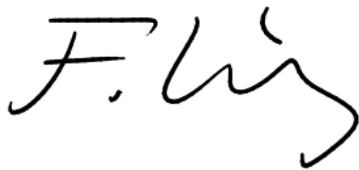
- [44] Inc. STEP Tools. *SCHEMA*. 2020. URL: https://www.steptools.com/stds/stp_aim/html/schema.html#step_merged_ap_schema (besucht am 10.06.2022).
- [45] Open CASCADE Technology. *Modeling Algorithms: Meshing algorithm*. URL: https://dev.opencascade.org/doc/occt-7.1.0/overview/html/occt_user_guides_modeling_algos.html#occt_modalg_11_2 (besucht am 20.09.2022).
- [46] *This is the official source code of FreeCAD, a free and opensource multiplatform 3D parametric modeler*. 2022. URL: <https://github.com/FreeCAD/FreeCAD/blob/dc2bf6c7546e2f079165ca5163a8dd76f4a3a64b/README.md> (besucht am 07.09.2022).
- [47] Xi Vincent Wang und Xun W. Xu. „A collaborative product data exchange environment based on STEP“. In: *International Journal of Computer Integrated Manufacturing* 28.1 (Mai 2013), S. 75–86. DOI: 10.1080/0951192x.2013.785028.
- [48] C. Weber. „What is a Feature and What is its Use: Results of FEMEX Working Group I“. In: *D. Roller (Ed.), 29th International Symposium on Automotive Technology and Automation: Florence, Italy 3* (Juni 1996).
- [49] webpack. *Tree Shaking*. URL: <https://webpack.js.org/guides/tree-shaking/> (besucht am 13.09.2022).
- [50] Leo S. Wierda. „Linking Design, Process Planning and Cost Information by Feature-based Modelling“. In: *Journal of Engineering Design* 2.1 (Jan. 1991), S. 3–19. DOI: 10.1080/09544829108901667.
- [51] Mazin Al-wswasi, Atanas Ivanov und Harris Makatsoris. „A survey on smart automated computer-aided process planning (ACAPP) techniques“. In: *The International Journal of Advanced Manufacturing Technology* 97.1-4 (Apr. 2018), S. 809–832. DOI: 10.1007/s00170-018-1966-1.
- [52] Jinhua Xiao u. a. „Information exchange standards for design, tolerancing and Additive Manufacturing: a research review“. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 12.2 (Mai 2017), S. 495–504. DOI: 10.1007/s12008-017-0401-4.
- [53] Xun Xu, Lihui Wang und Stephen T. Newman. „Computer-aided process planning – A critical review of recent developments and future trends“. In: *International Journal of Computer Integrated Manufacturing* 24.1 (Jan. 2011), S. 1–31. DOI: 10.1080/0951192x.2010.518632.

- [54] Yusri Yusof und Kamran Latif. „Survey on computer-aided process planning“. In: *The International Journal of Advanced Manufacturing Technology* 75.1-4 (Juli 2014), S. 77–89. DOI: 10.1007/s00170-014-6073-3.
- [55] Xionghui Zhou u. a. „A feasible approach to the integration of CAD and CAPP“. In: *Computer-Aided Design* 39.4 (Apr. 2007), S. 324–338. DOI: 10.1016/j.cad.2007.01.005.

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Seitens des Verfassers bestehen keine Einwände die vorliegende Masterarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

A handwritten signature in black ink, appearing to read 'F. Witz'. The letters are cursive and fluidly connected.

Jena, 22. September 2022