Institut für Rechnergestützte Ingenieursysteme

Fakultät Informatik, Elektrotechnik und Informationstechnik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3631

STEP/XML Based Macro Data Representation for Parametric CAD Model Exchange

Firas Zoabi

Studiengang:	Softwaretechnik
Prüfer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Dr. Bernadetta Kwintiana Ane
begonnen am:	10.03.2014
beendet am:	09.09.2014
CR-Klassifikation:	I.2.10, I.4.8, I.5

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

In diesem Kapitel wird eine Einführung aller Punkte wie Hintergründe dieser Arbeit, Aufgabenstellung sowie die jetzige Stand der Forschung vorgestellt bzw. erläutert.

Im weiteren wird diese Abschnitt der Diplomarbeit der Zweck dieses Dokumentes und die Ziele der zu entwickelnde STEP2XML Software beschreiben.

1.1 Forschung Hintergrund

Die Beschreibungssprachen für die Produkt-Modellierung werden heutzutage in der Industrie, wie z.B. in der Automobil-Branche, industriellen Produkt - und Produktteil-Modellierung und in den meisten Bereichen der Produktmodellierung eingesetzt. Dabei werden die wichtigsten Standards IGES, DXF und STEP verwendet.

1.2 Problemstellung

Die Produkt-Modellierung entwickelt sich rasant in allen Bereichen der Industrie und die Anzahl der Anforderungen an die Produktmodellierung steigt. Dadurch erhöht sich auch die Komplexität der Beschreibungssprachen. Mit dieser Entwicklung kommen immer neue Beschreibungssprachen auf den Markt der Produktmodellierung und dadurch steigt auch der Bedarf an Definitionen neuer Standards. Es gibt viele Standards für die Produktmodellierung in allen Bereichen der Industrie, Architektur etc. Im Rahmen der Diplomarbeit werden verschiedene wichtige Standards einheitlich interpretiert und in ein einheitliches Format konvertiert, damit diese wiederum in anderen Beschreibungs- bzw. Programmiersprachen verwendet werden können.

Die Konvertierung der verschiedenen wichtigen STEP-Dateiinhalte, die für die Rekonstruktion der modellierten Objekte relevant sind, wird automatisiert durchgeführt und das Ergebnis der Umwandlung wird im XML-Datenformat strukturiert dargestellt.

1.3 Zielsetzung dieser Diplomarbeit

STEP ist ein neuer Standard und wurde entwickelt, um als ein einheitlicher Standard für die Datenaustauche zwischen verschiedene CAD-Systeme. STEP Standard ist neu und wird nicht von alle Systeme unterstützt. Auf der andere Seite wird Heutzutage XML-Technologie fast in allen Systemen unterstützt und eingesetzt. XML eignet sich als Datenformat für die Datenaustausch zwischen verschiedene Applikationen, weil es estrukturiert und einfach zu interpretieren und zu parsen. Aufgrund dieser Tatsache wird ein Vorgehen für das Umwandlung und Darstellung der STEP- in XML-Daten erforscht.

In dieser Arbeit sind alle Resultate und Vorgehensweisen einer Methodik zum Parsen und Darstellung von STEP-Features in XML-Macro-Data zusammengefasst.

Dieses Dokument hat zwei weitere Ziele. Zum einen soll es die Tool-Software STEP2XML erläutern und als Handbuch dienen.

Zum anderen soll dieses Dokumentes als eine Spezifikation ähnliche Vorlage sein, die alle theoretischen sowie technischen Themen wie Definitionen, Zwecke,

Systemarchitektur, Entwurf sowie Feinentwurf und Implementierung mit fast allen Details und Funktionen beinhaltet.

Die detaillierte Erläuterung in Kapitel Feinentwurf ist für eine wichtige Eigenschaft bei der Entwicklung des Tools und zwar für die Erweiterbarkeit notwendig.

Zudem soll das Dokument als eine Art Einführung und als Grundlage für die Verwendung der Software dienen.

1.4 Methodologie

Im Folgenden handelt es sich um die Methodik und das Vorgehen der Arbeit. Die technischen und praktischen Arbeitsschritte in der Entwicklung werden näher beschrieben.

Figure 1 "Feinentwurf der Systemarchitektur" im Kapitel Feinentwurf ist die Methodik in einzelnen Schritten graphisch dargestellt.

In der Methodologie werden die Arbeitsschritte, die für das Erreichen des Ziels der wisenschaftlichen Arbeit benötigt werden, definiert.

Diese gliedern sich in folgende Schritte:

- Schritt 1: Erstellung der CAD-Produktmodellierung Zeichnungen
- Schritt 2: Entwurf des Vorgehens für die CAD-Feature und Extraktion
- Schritt 3: Entwicklung des STEP-Daten Parsers *STEPTextAnalyser*
- Schritt 4: Erstellung der Java-Klasse *STEPElement*, in der alle Daten aus den STEP-Textzeilen abgebildet werden
- Schritt 5: Eine Objekt-Liste für das Speichern der Java-Klassenobjekte der Klasse *STEPElement.java*
- Schritt 6: Definition und Entwurf der XML-Struktur
- Schritt 7: Entwicklung der XML Parser für die Umwandlung der *STEPElement* - Objektdaten in XML-Elemente und XML-Struktur
- Schritt 8: Programmieren von *PDFGenerator.java* zur Erstellung der Berichte im PDF-Format
- Schritt 9: Erweiterung des XML-Parsers *XMLEngine.java*
- Schritt 10: Erstellung der Funktionen für die 3D-Darstellung bzw. Remodellierung XML-Daten
- Schritt 11: Verbesserung des "graphische-Modell-Erkennung" Algorithmus

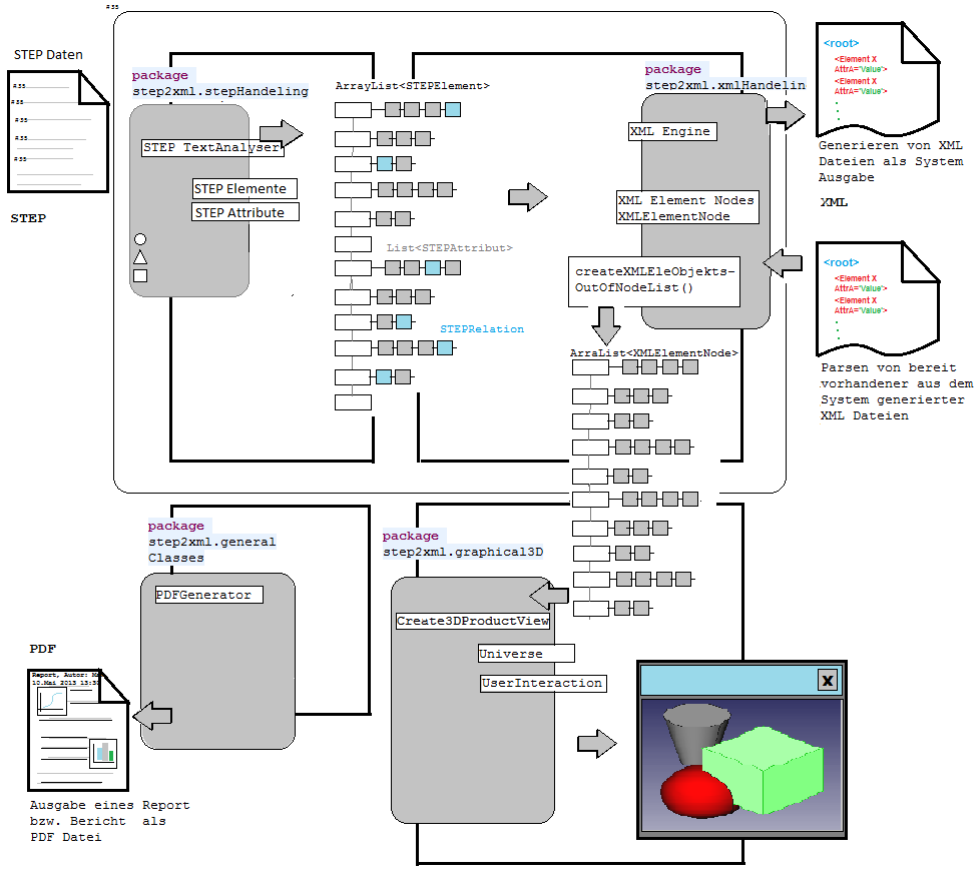

Figure 1: Feinentwurf der Systemarchitektur

1.5 Diplomarbeitsgliederung

Hier wird ein Gliederung der Arbeit von Einleitung über die Problematik, Darstellung des Vorgehens und Lösungsansatz bis hin zu Hervorhebung der Ergebnise und Darstellung der Analyse vorgestellt.

Kapitel 1: Im Kapitel Einleitung soll hier die Darstellung des Vorgehens der Arbeit-sschritte und die Beschreibung der Problematik und Lösungsansatzes kurz dargestellt werden.

Kapitel 2: In diesem Kapitel werden die Technologien, Themenbereiche und wissen-schaftliche Auseinandersetzung wie CAD-Design, Produktdatenaustausch und STEP als ISO-Standard im Rahmen der Recherche behandelt, aufgelistet und einzeln erläutert.

Kapitel 3: Im Folgendem wird das Vorgehen der Arbeit in Arbeitsschritten erklärt.

Kapitel 4: Es erläutert die STEP-Dateistruktur und -inhalte und im Wieteren ist das Kapitel "Implementierung" in Grobentwurf, graphische Modell-Erkennung und Feinentwurf untergliedert. Dabei beinhaltet Grobentwurf die Systemarchitecktur und Erläuterungen wie aus den STEP-Elementen Klassenob-

jekte erstellt werden. Graphische Modell-Erkennung erläutert das Vorgehen der Abstraktion der STEP-Zeilen, STEP-Elemente und wichtiger Daten. Im dritten Teil "Feinentwurf" werden UML-Diagramme, der Programmcode, die XML-Struktur und die Benutzerschnittstellen GUI vorgestellt.

Das 4. Kapitel ist von großer Bedeutung für diese Arbeit, weil es das Technische Vorgehen der Entwicklung und die Ergebnisse spezifiziert und erläutert.

Kapitel 5: Im Kapitel 5 werden die Ergebnisse und die allgemeine Analyse vorgestellt.

Kapitel 6: Eine Zusammenfassung mit offenen Forsch-ungspunkten für zukünftige Arbeiten und Projekte runden diese Arbeit ab.

2 Technologien Überblick

Folgendes Kapitel beschreibt die Technologien, die in dieser Arbeit eingesetzt wurden und darunter der STEP Standard.

2.1 CAD-Design

CAD-Design ist das Produkt einer rechnerunterstützten Konstruktion von Zeichnungen. CAD ist die Abkürzung für "*Computer-Aided Design*" und wird bei der Konstruktion und Zeichnung von 2D bzw. 3D Modellen von Produkten oder Teilprodukten in der Industrie verwendet. Es wird fast in allen Engineering und Produktionsbranchen, wie die Automobilindustrie, Architektur, Maschinen und Flugzeugbau eingesetzt.

2.1.1 Parametrisches CAD-Modell

Für CAD-Modelle existieren viele Modellierungsarten wie Körper-, Flächen- und Volumenmodell und Parametrische Modellierung. Parametrische Modellierung unterscheidet sich von den anderen darin, dass bei diesem Verfahren vordefinierte geometrische Objekte wie Linie, Punkt, Fläche und Kurven im Modell verwendet und ihre Beziehungen zueinander per Parameter beschrieben werden.

Pro geometrisches Objekt werden Werte verschiedene Parameter zugewiesen. Dadurch wird das Modell aus verschiedenen parametrisierten geometrischen Objekten zusammengesetzt und falls Änderungen am Modell angebracht werden sollen, werden nur bestimmten Parametern neue Werte zugewiesen. Somit wird an Zeit, Aufwand und weiteren Resourcen bei der Anpassung oder Remodellierung von weiteren Teilprodukten gespart. FreeCAD (siehe Abschnitt "2.4.8 FreeCAD Editor" unter "Angewandete Technologien") ist ein Beispiel für ein CAD-Programm, das nach dem Prinzip der Parametrischen-CAD-Modellierung konzipiert wurde.

2.1.2 STEP Macro-Daten

Ein wichtiger Aspekt ist die Darstellung der Daten in STEP und ihre Verbindung zueinander. Dies alles wird durch verschachtelte Referenzen realisiert. Fast jedes Schlüsselwort hängt von einem oder mehreren adressierten STEP Skript-Zeilen ab.

Nehmen wir als Beispiel das Schlüsselwort VERTEX_POINT. Es beinhaltet in Klammern eine Raute mit bestimmter Zeilenzahl. Dies ist eine Referenz zu einer anderen Zeile, die den Punkt Definition beinhaltet (siehe Beispiel unten)

#85 = VERTEX_POINT('',#86);

#86 = CARTESIAN_POINT('',(-0.125416859985,8.513317108154,12.11553-478241));

In STEP sind Schlüsselwörter für verschiedene Modellierungsaspekte und -eigenschaften definiert.

AXIS2_PLACEMENT_3D definiert die folgende Zeichnung im 3D-Raum durch die Referenzen von drei XYZ-Punkten, wie im folgenden Beispiel gezeigt:

#101 = AXIS2_PLACEMENT_3D('',#102,#103,#104);

#102 = CARTESIAN_POINT(",(-0.125416859985,8.513317108154,2.11553-478241));

#103 = DIRECTION(",(-1.10393685431E-014,1.,5.60380480761E-015));

#104 = DIRECTION(",(-8.431993804898E-022,-5.60380480761E-015,1.));

2.1.3 Autodesk Inventor & Solidworks :

Solidworks und *Autodesk-Inventor* sind Werkzeuge für das Zeichnen und die 2D-, 3D-Modellierung von Produkten und Teilprodukten. Die Zeichnungsdaten können über diese Programme in verschiedenen Formaten exportiert werden. Im Folgenden werden drei Datenaustausch-Formate *STEP*, *DXF* und *IGS* betrachtet :

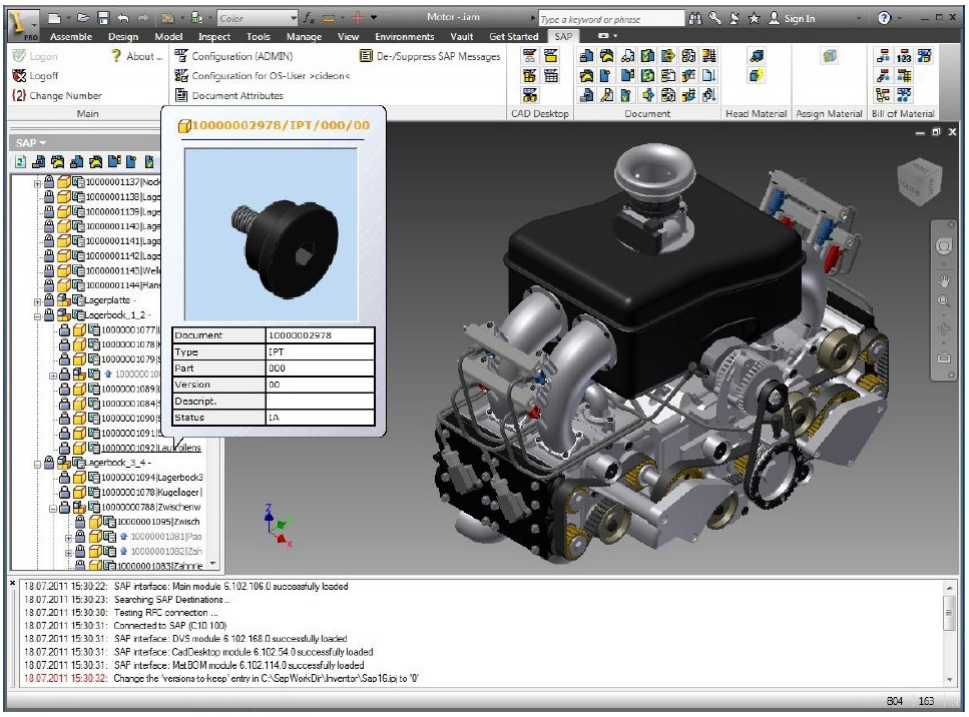

Figure 2: Hauptfenster des Programms *Autodesk Inventor*

DXF steht für ein Standard für den Austausch von Produktmodellierungsdaten und ist für die Abkürzung *Drawing Interchange File Format*. Es wurde von Autodesk für den Datenaustausch in CAD spezifiziert und im Programm AutoCAD als Datenformat integriert.

STEP ist eine ISO-Norm, die verschiedene Funktionen wie Archivierung aber auch eine Hauptfunktion für den Austausch von Produktsmodellierungsdaten darstellt und definiert.

IGS IGES oder IGS ist die Abkürzung für *Initial Graphics Exchange Specification* und die aktuelle Version davon wurde im Jahre 1996 als Standard des ANSI definiert. IGES wurde von der U.S. Product Data Association

entworfen und definiert. IGES steht für ein neutrales und herstellerunabhängiges Datenformat

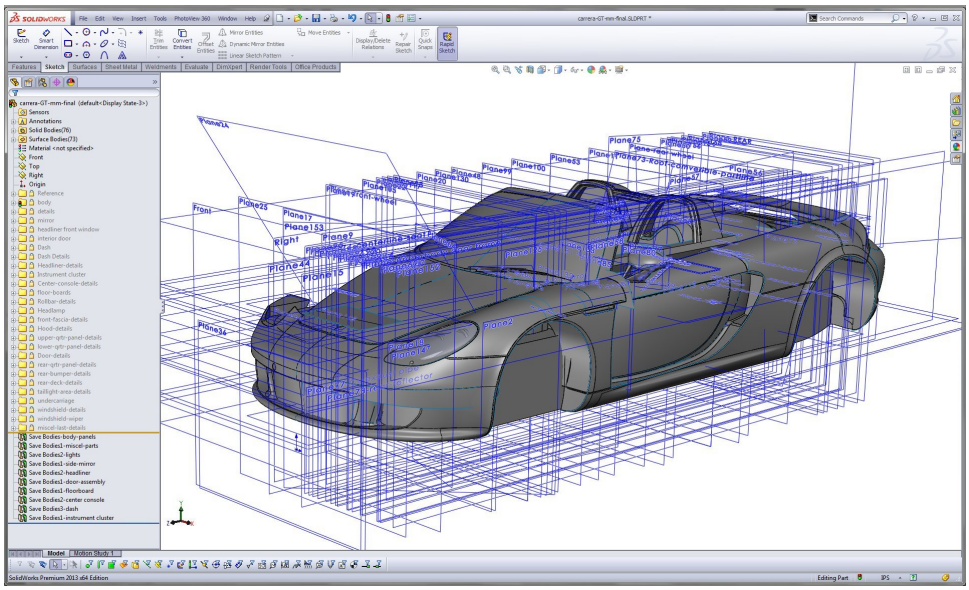

Figure 3: Hauptfenster des Programms *SolidWorks*

Ein wichtiges Vorgehen bei der 3D-Modellierung, der Xtrusion wird in beiden Programmen *Solidworks* und *Inventor* verwendet, um aus einer Zeichnung in 2D-Ebene eine 3D-Zeichnung zu erzeugen. Eine Zeichnung wird zuerst als 2D-Objekt anhand Endkoordinaten, Direction (Richtung auf der Axiome XYZ Achse) oder anhand bestimmter vorgegebener Meßdaten wie Höhe oder Tiefe, Länge und Breite erzeugt und im Nachhinein in 3D über weitere Meßwerte vervollständigt .

Als Beispiel ist folgende Figure gegeben (siehe Figure 4). Hier wird das Vorgehen illustriert, wie graphische Editoren, unter anderem *Autodesk Inventor*, kann ein 2D Objekt in 3D Objekt anhand Start-, Endkoordinaten und Höhe berechnen.

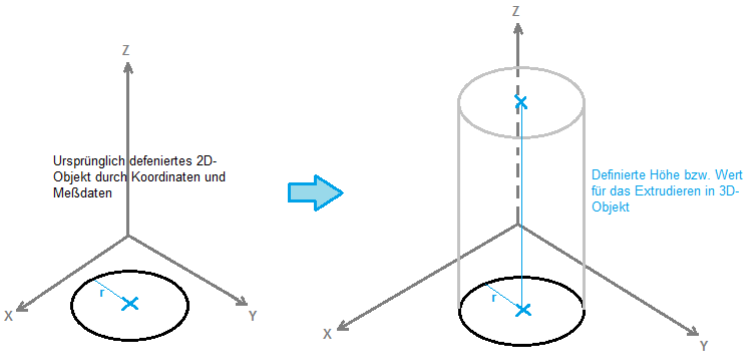

Figure 4: Illustriert die Erstellung von 3D-Objekten

2.2 Produktdatenaustausch

2.2.1 CAD-Feature-Kernel

CAD-Feature-Kernel ist eine Bezeichnung für die Menge an Daten eines CAD Produktdatenaustausch-Standards. In unserem Fall handelt es sich um den STEP Standard, wobei die STEP-Beschreibungs- bzw. Definitionsdaten der geometrischen primitiven Objekte wie Pyramide, Box etc. betrachtet werden.

Für die Art und Definition des Bauteils innerhalb einer Produktmodellierung werden wichtige STEP-Daten wie folgt interpretiert :

- *PRODUCT* bezeichnet den Namen und den Typ des Bauteiles wie im Beispiel zu sehen ist

  PRODUCT('Box','Box','',(#8)); in diesem Fall handelt es sich um einen Würfel. Hier wird das Produkt definiert

  PRODUCT_DEFINITION_SHAPE, #4 = PRODUCT_DEFINITION_-SHAPE('','',#5);

  PRODUCT_DEFINITION, #5 = PRODUCT_DEFINITION('design','',-#6,#9);

  PRODUCT_DEFINITION_FORMATION, #6 = PRODUCT_DEFINITION_FORMATION('','',#7);

  Weitere Schlüsselwörter beinhalten Daten über die Koordinaten und Punkte auf dem Achsensystem und werden mit *POINT* gekennzeichnet:

- *VERTEX_POINT* ist ein Punkt auf der XYZ-Achse und definiert ein Grenzpunkt für das graphische Modellteil,

  #24 = VERTEX_POINT('',#25);

  #25 = CARTESIAN_POINT('',(-0.125416859984,-1.486682891846,12.115-53478241));

- *CARTESIAN_POINT* ist die Basisdefinition für ein Punkt auf dem Achsensystem und stellt das kartesische Produkt aus den drei Koordinaten dar, Beispiel:

  #39 = CARTESIAN_POINT(",(0.E+000,0.E+000));

Schlüsselwörter zur Zeichnung von Linien, Fläschen, Kurven etc. :

- *LINE* ist ein Befehl zur Zeichnung von geraden Linien zwischen definierten bzw. gegebenen Punkten , #38 = LINE(",#39,#40);
- *PLANE* ist für die Erstellung von Flächen,

  #44 = PLANE(",#45); die Interpretation für diese Skript-Zeile ist

  "Zeichne eine Fläche auf dem 3D - Achsensystem mit einem bestimmten XYZ-Punkt, die in Zeile #45 definiert ist.

  #45 = AXIS2_PLACEMENT_3D(",#46,#47,#48);
- *VECTOR* ist für das Zeichnen von Vektoren, #40 = VECTOR(",#41,1.);
- *CURVE* ist zum Zeichnen von Kurven in verschiedenen Arten. Dabei gibt es unter anderem folgende:
  - EDGE_CURVE, #84 = EDGE_CURVE(",#57,#85,#87,.T.);
  - SURFACE_CURVE, #87 = SURFACE_CURVE(",#88,(#92,#99),-.PCURVE_S1.);
  - PCURVE, #92 = PCURVE(",#32,#93);

2.2.2 Produktdaten Übersetzung

Bei der Übersetzung der STEP-Daten werden mehrere STEP-Schlüsselwörter berücksichtigt, die wichtige Daten für die XML-Dateiinhalte und für die spätere graphische Remodellierung des Produkts bzw. Teilprodukts darstellen.

Um das zu illustrieren, wird ein Ausschnitt der Skriptsprache aus STEP mit der Übersetzung in die XML-Struktur in folgender Figure dargestellt "Figure 5: Die Übersetzung der STEP-Daten in XML-Inhalte"

Auf der Figure Links befindet sich ein Abschnitt von STEP-Daten für ein Zylinder und auf der rechten Seite werden Daten in XML-Inhalte übersetzt.

Die wichtigen Inhalte für die Übersetzung in XML-Daten und der späteren graphischen Remodellierung der Produkte sind wie folgt aufgelistet:

- der Name des Produktes, welches unter anderem die STEP-Zeile "PRODUCT('Cylinder', 'Cylinder', " (#254))" beinhaltet
- die Meßdaten für das Objekt in unserem Fall sind das der Radius und die Länge der Zylinder
- die Koordinaten aus dem STEP-Kontext zum Parsen

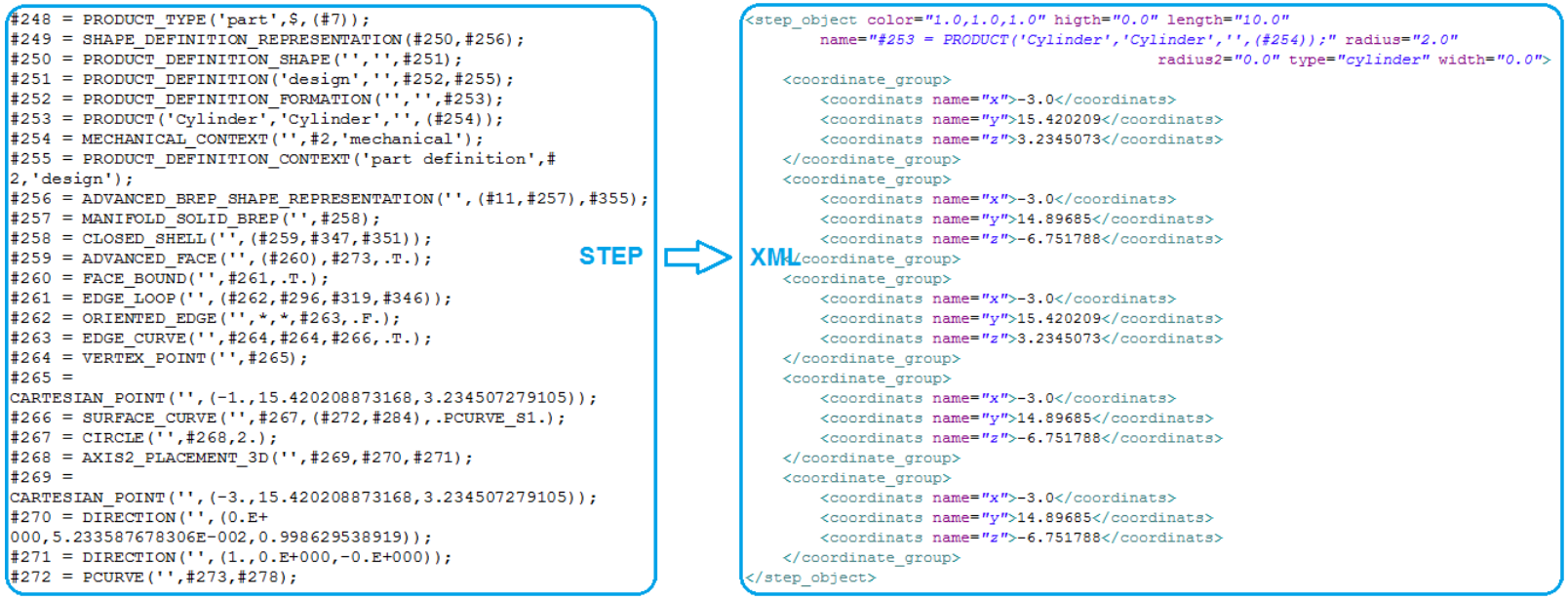

Figure 5: Die Übersetzung der STEP-Daten in XML-Inhalte

2.3 STEP (ISO-10303) Standard

In diesem Kapitel wird der STEP Standard von der Entstehung STEP über technische Aspekte bis hin zur technischen Anwendung in der Entwicklung der Modellierungssprache EXPRESS dargestellt.

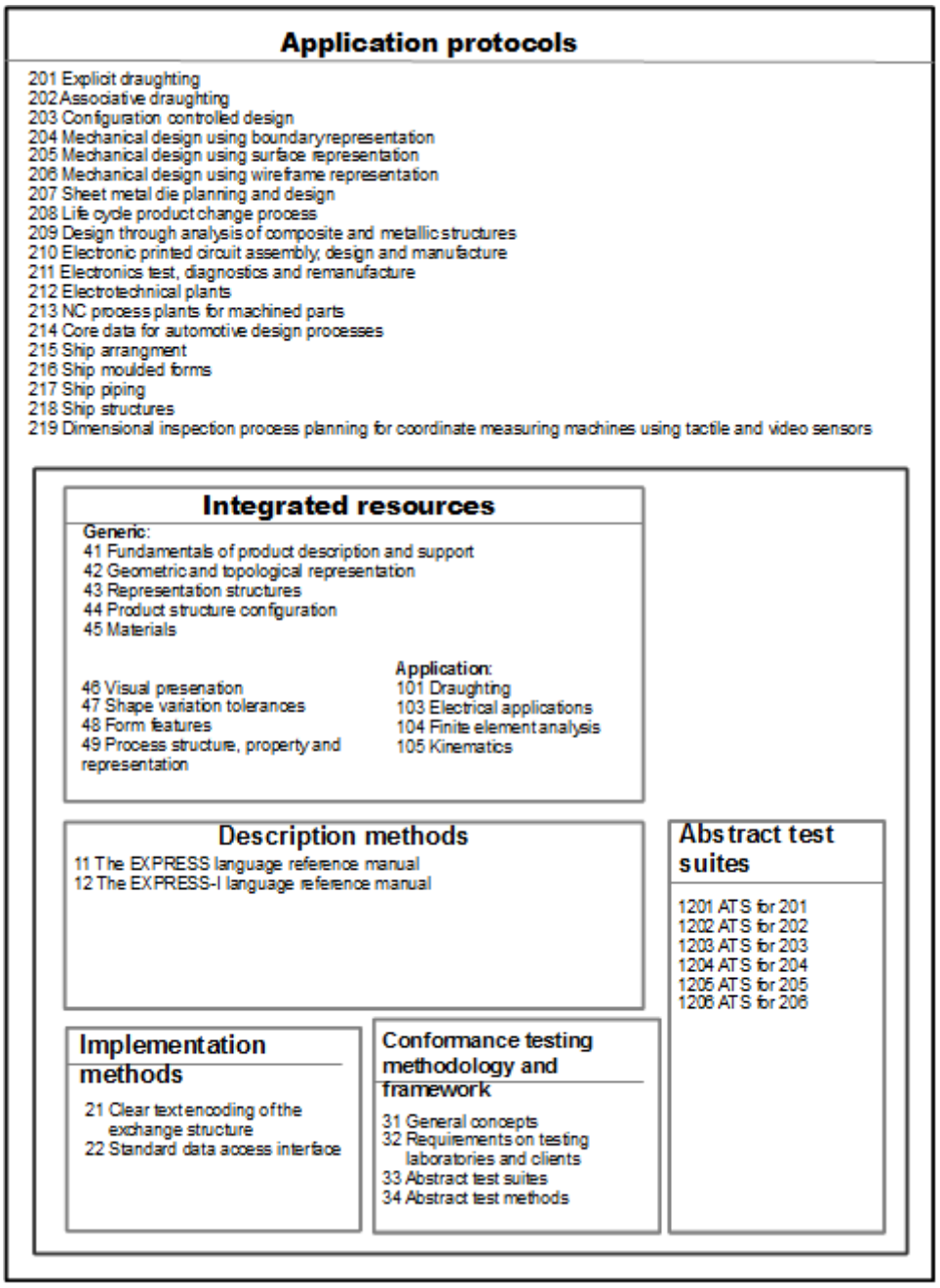

Figure 6: STEP Strukturen. Nachgezeichnet [1]

2.3.1 Die Entstehung und Entwicklung von STEP als ISO-Norm

In der ISO-Norm STEP handelt es sich um verschiedene Funktionen wie Archivierung. Ebenso der Austausch von Produktsmodellierungsdaten ist eine Hauptfunktion dieser ISO-Norm. Im Jahre 1979 wurde in den USA das Austauschformat IGES (Initial Graphics Exchange Specification) entwickelt und für den Austausch und die Archivierung von Produktdaten bzw. technische Zeichnungen verwendet. Mit der weiteren Entwicklung im Laufe der Jahre von CAD-Systemen, wurde IGES immer weiter entwickelt.

Aufgrund von Probleme bei der Übertragung von IGES-Dateien, die wegen den unterschiedlichen Interpretationsmöglichkeiten von IGES verursacht wurden, wurde IGES vom Verband der Automobilindustrie (VDA) in verschiedene Leistungsstufen aufgeteilt. Die Leistungsstufen wurden in zwei Stufen spezifiziert : -VDA-IS -und IGES-Subsets des Verbands der deutschen Automobilindustrie.

Diese Leistungsstufen dienen der Spezifikation des Umfangs der Übertragung von Modelldaten.

Ein weiteres Austauschformat wurde in Frankreich entwickelt, welches SET genannt wurde. SET ist die Abkürzung für Standard d'Exchange et de Transfer.

Im Folgenden sollen die Austauschformate näher erläutert werden:

IGES Die aktuelle Version wurde im Jahre 1996 als Standard des ANSI definiert. IGES wurde von der U.S. Product Data Association entworfen und definiert. IGES steht für ein neutrales und herstellerunabhängiges Datenformat.

VDA VDA steht für ein deutsches Qualität-Management-System und ist eine Initiative der Automobilbranche. Die erste Version aus diesem Standard wurde mit VDA-FS bezeichnet und stand für Datenaustausch für Modelldaten.

SET SET ist ein Standard für den Modell-Datenaustausch für CAD und wird in Frankreich verwendet.

Alle diese Austauschformate haben die Gemeinsamkeit, dass sie für den Austausch von Geometriedaten wie Linien, Volumen und technische Zeichnungen als ein Standard für den nationalen Raum konzipiert wurden. Mit der Entwicklung der Märkte und damit des Produktmodellierung-Sektors, sind diese Standards begrenzt und es entstand ein Bedarf und eine Nachfrage nach internationalem Standard. Deshalb wurde von der ISO entschieden, ein Internationales Standard mit der Bezeichnung STEP (*Standard for the Exchange of Produkt Model Data*) zu entwickeln. STEP wird heutzutage in mehreren großen Bereichen verstärkt angewendet, wie z.B. in der Automobil- und Luftfahrtindustrie.

2.3.2 Die Modellierungssprache EXPRESS

EXPRESS (Expressive Power) ist eine Datendefinitionssprache, die dazu dient, die komplexeren aufwendigen Spezifikationen von Daten für Menschen verständlicher zu machen.

EXPRESS-Schema

In EXPRESS werden die Daten durch mehrere Schemata modelliert. Durch diese Modellierung der Daten werden Metadaten kreiert, die man bei verschiedenen Anwendungen anpassen kann. Es ist sogar möglich, Verweise zwischen Objekten von verschiedenen Schemata zu definieren (siehe Figure 7).

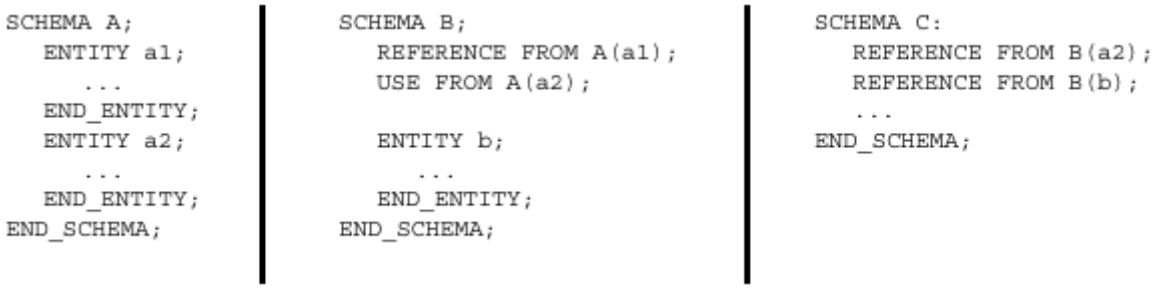

Figure 7: Beispiel für das Referenzieren in EXPRESS Schema zwischen verschiedenen Schemata [Doct-99]

Im Rahmen eines EXPRESS-Schemas werden die Objekte als Entities beschrieben. Die Entities kann man mit Klassen in den objektorientierten Modellen abbilden bzw. definieren.

Mit Entities werden objektorientierte Strukturen durch die Definition von Typklassen gebildet.

Im EXPRESS-Schema wird die Entität mit dem Schlüsselwort ENTITY gekenn-zeichnet danach wird der Name des neuen Typs und Definition des Verweises zu anderen Supertypen oder Subtypen definiert. Damit wird die Definition der Vererbung festgelegt. Anschließend folgt die Beschreibung eigener Attribute und lokale Regeln.

```
SCHEMA Produkt;
   REFERENCE FROM Produzent (Produzent);

   TYPE Art = ENUMERATION OF (komplex,einfach);
   END_TYPE;

   TYPE Farbton = ENUMERATION OF (rot, blau, gelb, grün, weiß);
   END_TYPE;

   TYPE Produkte = SET [2:?] OF Produkt;
   END_TYPE;

   FUNCTION Berechne_Gewicht (in : Produkte) : REAL;
      LOCAL
         result : REAL:=0;
      END_LOCAL;
      REPEAT i:=LOINDEX(in) TO HIINDEX(in);
         result:=result + in[i].Gewicht;
      END_REPEAT;
      RETURN (result);
   END_FUNCTION;

   ENTITY Produkt ABSTRACT SUPERTYPE OF
                     ONEOF (Einfaches_Produkt, Komplexes_Produkt);
      Produkt_Typ : Art;
      Gewicht : REAL;
      Farbe : OPTIONAL Farbton;
   INVERSE
      hergestellt_von : Produzent FOR produziert;
      benötigt_für : SET [0:?] OF Komplexes_Produkt FOR besteht_aus;
   UNIQUE
      Gewicht, Farbe;
   END_ENTITY;

   ENTITY Einfaches_Produkt SUBTYPE OF (Produkt);
   DERIVE
      SELF\Produkt_Typ : Art := einfach;
   END_ENTITY;

   ENTITY Komplexes_Produkt SUBTYPE OF (Produkt);
      besteht_aus : Produkte;
   DERIVE
      SELF\Produkt_Typ : Art := komplex;
      SELF\Gewicht : REAL := Berechne_Gewicht (besteht_aus);
   END_ENTITY;

   ENTITY Auto  SUBTYPE OF (Komplexes_Produkt);
      SELF\Farbe : Farbton; -- jetzt vorgeschriebenes Attribut
   END ENTITY;
```

Figure 8: Beispiel für ein EXPRESS-Definitionen des Schemas *Produkt [Doct-99]*

Typen in EXPRESS

Für die Definition von EXPRESS-Schemata sind folgende verschiedene Typen und Typkonstruktoren vorhanden:

- Basistypen (Simple Types)
- Aggregate (Aggregation Data Types)
- Allgemeine Supertypen (Select Data Type)
- Aufzählungstypen (Enumeration Data Type)
- Objekte (Entity Data Type)
- Benutzerdefinierte Typen (Defined Data Type)

```
-- Kommentare werden durch "--" eingeleitet

-- Defined / Aggregation / Simple Data Type:
TYPE vector = ARRAY [1..3] OF INTEGER;
END_TYPE;

-- Defined / Enumeration Data Type:
TYPE figure = ENUMERATION OF (circle, box);
END_TYPE;

-- Defined / Simple Data Type with Where-Clause:
TYPE positive = INTEGER;
WHERE
   notnegative : SELF >= 0;
END_TYPE;

-- Defined / Select / Simple Data Type:
TYPE own_number = SELECT (NUMBER, positive);
END_TYPE;
```

Figure 9: Typdeklaration in EXPRESS [Doct-99]

2.3.3 Weitere Standards für die Produktmodellierung

Neben STEP als Standard für die Produktmodellierung Datenaustausch werden weitere verschiedene Standards eingesetzt.

Ich werde auf zwei wichtige Standards, die stark in der Produktmodellierung Datenaustausch im Markt benutzt werden, eingehen.

Die unten aufgeführten Standards unterscheiden sich sehr von STEP. Die Unterschiede liegen sowohl in der Struktur und Anordnung der Modellierungsdaten als auch in der Beschreibungssprache bzw. Schlüsselwörter. Aufgrund dieser Unterschiede in den Modellierungsdaten Struktur etc. werden die Standards entsprechend anders abgelesen und verarbeitet.

IGES

IGES ist die Abkürzung für *Initial Graphics Exchange Specification* und die aktuelle Version davon wurde im Jahr 1996 als Standard des ANSI definiert. IGES wurde von der U.S. Product Data Association entworfen und definiert.

IGES steht für ein neutrales und herstellerunabhängiges Datenformat, das dem Datenaustausch von Modellierungsdaten dient. Die IGES-Daten einer Zeich-nung wie Linie, Punkt, Kurve oder Fläche werden in Form von einzelnen Entities oder vordefinierten *Entity*-Typen strukturiert bzw. gespeichert.

DXF

DXF ist ein Standard für den Austausch von Produktmodellierungsdaten und steht für Drawing Interchange File Format. Es wurde von Autodesk für den Datenaustausch in CAD spezifiziert und im Programm AutoCAD als Datenformat integriert.

DXF wurde von AutoDesk gut dokumentiert und spezifiziert. Das war der Grund dafür, dass die Datenstruktur und Datenbeschreibung einfach nachzuvollziehen ist. Aufgrund dessen wird DXF für den programmübergreifenden Datenaustausch bei verschiedenen Programmen verwendet, sodass alle CAD Programme mit DXF umgehen und arbeiten können.

2.4 Direkt angewandte Technologien

In diesem Kapitel wird ein Überblick über die in der Entwicklung verwendeten Technologien und Werkzeuge gegeben. Von der Extensible Markup Language abgekürtzt mit XML über Eclipse als verwendeter Editor mit der Entwicklungssprache JAVA bis hin zu Undefined Modelling Language abgekürzt mit UML.

Die Angewandte Technologien werden in zwei Gruppen angeordnet, die direkt angewandte Technologien und die indirekt angewandte Technologien.

Im Folgendem werden die direkt angewandten Technologien, die die Implementierung direkt be-einflussen, aufgelistet. Die Programmiersprache Java oder die Metasprache XML sind ein Beispiel dafür.

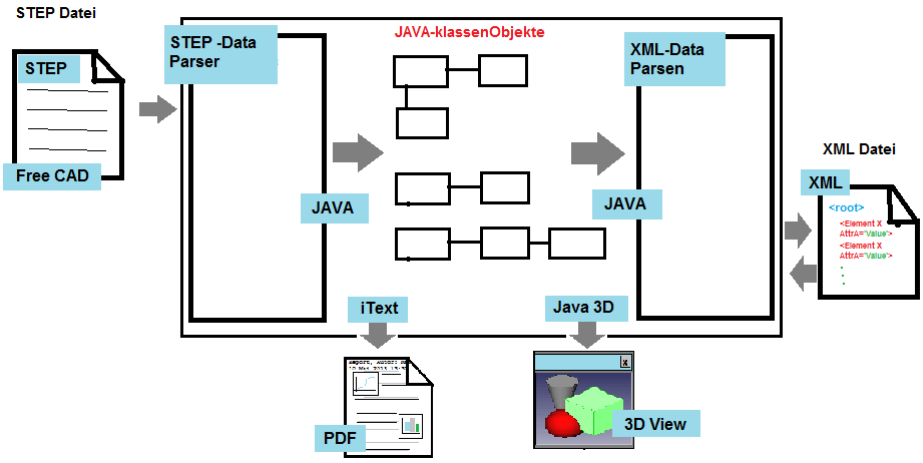

Figure 10: graphische Darstellung der angewandte Technologien in der Entwicklung

2.4.1 Parser

Ein Parser ist ein Objekt, der elemente einer Sprache bzw. Text identifisiert und übersetzt die Daten aus dieser Elemente in eine brauchbare und lesbare Inhalte. Der Parser oder auf Deutsch Zerteiler ist für das Analysieren von beliebigen Text-Eingaben. Dieser Prozess des Analysieren nennt sich auch Syntaxanalyse (*Parsing*), die nicht nur für das Zerlegung von beliebigen Daten dient, sondern für die Untersuchung von natürlichen Sprachen. In der Grammatik *Parsing* eingesetzt, indem es eines Satzes in seiner grammatikalischen Bestandteile (Syntax) zerlegt.

Es existieren verschiedene Arten von Parser und sind im Folgendem aufgelistet:

- Top-Down-Parser: es sind auch als Aufwärtsparsern bezeichnet wie LF-Parser und LL-Parser
- Bottom-Up-Parser: LR-Parser, SLR-Parser und LALR-Parser
- Chart-Parser: es wird Chartparser bezeichnet und wird für das Parsen von kontextfreien Grammatiken
- Left-Corner-Parser: LC-Parser

Ein Parser ist auch eine Art Mechanismus zur Aufbereitung und Abstraktion von Daten aus strukturierten Datenquellen wie XML oder CSV, aber auch zum Aufbereiten bestimmter Daten aus unstrukturierten Datenquellen wie eine Textdatei oder aus einer beliebigen Datenmenge.

Ein Parser ist für die Umwandlung der geparsten Daten in eine spezielles vordefiniertes Format zuständig. Die Umwandlung der Daten in ein bestimmtes Format bzw. Struktur wird mit Hilfe von Funktionen und Prozeduren, die jeweils eigene Aufgaben innerhalb einer Parser-Klasse haben, durchgeführt. Eine wichtige Aufgabe eines Parsers ist die Manipulation und Umschreibung bzw. Strukturierung der Daten von einem Format in ein anderes.

2.4.2 STEP Data Parser

Im Rahmen der Entwicklung wurde ein eigener Parser mit verschiedenen Funktionen entwickelt. Dieser Parser beinhaltet Funktionen, die jeweils für bestimmte Analyseaufgaben zuständig sind, um die komplexen STEP Daten aus der STEP-Datei zu parsen. Die Parser-Funktionen (siehe Figure 11 und 12) wurden mithilfe der Programmiersprache Java entwickelt und bilden eine Struktur, die im engen Zusammenhang miteinander funktionieren.

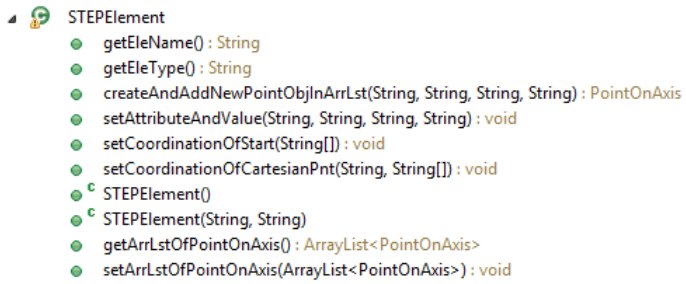

Figure 11: Funktionen und Methoden der Klasse *STEPElement*

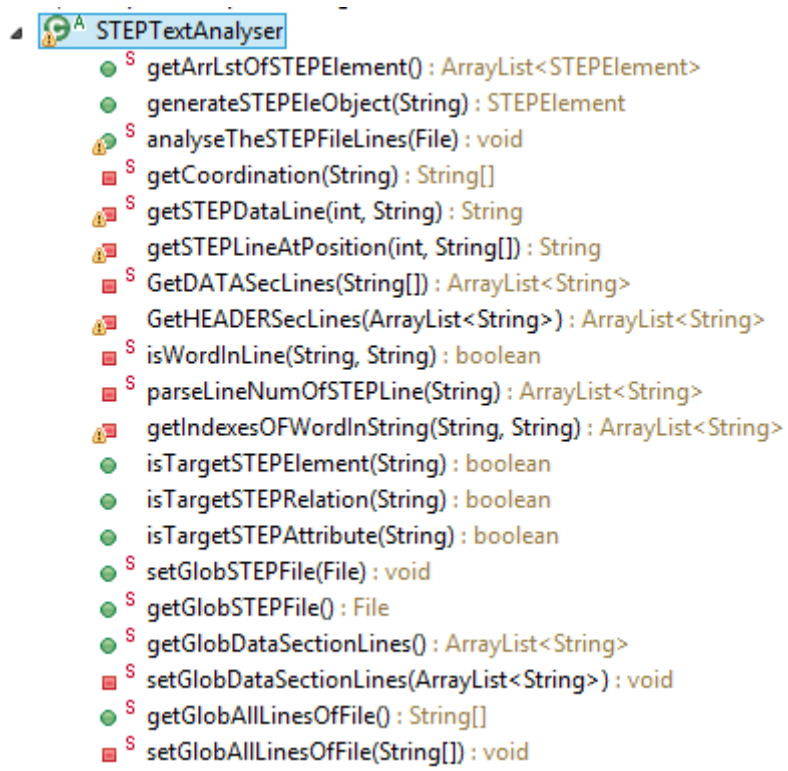

Figure 12: Funktionen und Methoden der Klasse *STEPTextAnalyser* .

2.4.3 Die Extensible Markup Language (XML)

Innerhalb mehrerer Jahre hat sich die Skriptsprache XML immer wieder entwickelt und sich in der Entwicklung von Software-Systemen durchgesetzt. Sie wurde zu einem wichtigen Bestandteil vieler Software Produkte. XML eignet sich für alle Systeme und dank ihres strukturierten und textbasierten Formats ist XML für die meisten Programmiersprachen und -umgebungen kompatibel bzw. unterstützbar.

Eine wichtige Funktion von XML ist der Einsatz für den Datenaustausch von simplen und komplexeren Datenstrukturen. XML erleichtert die Definition von komplexen Datenstrukturen und ermöglicht den Anwendern Flexibilität. Die einheitliche Struktur von XML ermöglicht das Parsen und die Interpretation von XML-Daten in verschiedenen Systemen. Die Definition der XML-Struktur mit Attributen, Elemente etc. wird immer den Entwicklern überlassen, womit jeder Entwickler anhand den Anforderungen und dem Bedarf in seinen zuentwickelnden System die XML- Attribute, Elemente und weitere Bausteine der XML Struktur beliebig anpassen kann.

Die Eclipse Entwicklungsumgebung wurde auch als Editor zur Manipulation von XML Inhalten verwendet.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xsi:schemaLocation="http://www.w3.org/2001/XMLSchema STEP_XMLSchema.xsd ">
    <step_object name="object name" attribute1="data1" length="23"
        width="45" higth="56" mesurment1="21" color="black" >
        This is STEP a XML-Object!
        <coordination name="x"> 20,50 </coordination>
        <coordination name="y"> 13,24 </coordination>
        <coordination name="z"> 120,30 </coordination>
        <relation2next relation2next="top_merge" relation2object="object2">
        </relation2next>
    </step_object>
    <step_object name="object2" attribute1="data2" length="43"
        width="23" higth="76" mesurment1="03" color="red"  ..... attribute="bla">
        This is STEP a XML-Object!
        <coordination name="x"> 20,30 </coordination>
        <coordination name="y"> 34,54 </coordination>
        <coordination name="z"> 82,10 </coordination>
        <relation2next relation2next="side_cut" relation2object="stepobject3">
        </relation2next>
    </step_object>
    <step_object name="stepobject3" attribute1="data_stepobject3" length="43"
        width="87" higth="24" mesurment1="000" color="green"  ..... attribute="bla2">
        This is STEP a XML-Object!
        <coordination name="x"> 20,30 </coordination>
        <coordination name="y"> 24,64 </coordination>
        <coordination name="z"> 90,00 </coordination>
    </step_object>
    <relations>
        <relation name="top_merge" art="mergw" >
            This is the relation named "top_merge" data
            <coordination name="x"> 20,50 </coordination>
            <coordination name="y"> 13,24 </coordination>
            <coordination name="z"> 120,30 </coordination>
        </relation>
        <relation name="side_cut" art="cut" >
            This is the relation named "side_cut" information
            <coordination name="x"> 20,0 </coordination>
            <coordination name="y"> 40,0 </coordination>
            <coordination name="z"> 30,20 </coordination>
        </relation>
    </relations>
</root>
```

Figure 13: Beispiel eines XML Dokumentes

XML-Bausteine

XML-Elemente Elemente sind ein wichtiger Bestandteil in der Bildung von der XML Strukturen. Sie sind die Bausteine für die simplen bis hochkomplexen Strukturen mit einer riesigen Menge an Daten.

In einigen Lektüren wird von Knoten im XML Bereich gesprochen. Damit sind die XML Elemente gemeint, die in einem XML-Baum wie Knoten zu erkennen sind. Wie bei der Behandlung und Vorgehensweise bei den Graphen, geht man von einem Hauptknoten aus, der in XML als "*root*" Knoten bezeichnet wird. Kinderknoten kommen in verschiedene Tiefen und Ebenen vor.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<STEPObject name="object name" attribute1="data1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2001/XMLSchema STEP_XMLSchema.xsd ">

    Das ist ein STEP XML-Object!

</STEPObejct>
```

Figure 14: XML Element als Auschnitt aus einem Beispiel XML-Baum

Kinder Elemente Es gibt auch Kinder Elemente bei XML Strukturen. Jedes XML Element kann beliebig viele Kinder Knoten bzw. Elemente beinhalten. Kinder Elemente sind wiederum XML Elemente, die Attribute, Texte und weitere Kinder-Kinder Knoten beinhalten können.

Ein Element darf Kinder Elemente in beliebiger Verschachtlung bzw. Tiefe besitzen.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xsi:schemaLocation="http://www.w3.org/2001/XMLSchema STEP_XMLSchema.xsd ">>

    <STEPObject name="object name" attribute1="data1" length="23"
        width="45" higth="56" mesurment1="21" color="black" relation2next="xxxx" relation2object="yyyy">

        This is STEP a XML-Object!
        <STEPObject_chiled name="object name" attribute1="data1" length="23"
            width="45" higth="56" mesurment1="21" color="black" relation2next="xxxx">


        this is a STEP XML-Object chiled!

        </STEPObject_chiled>

    </STEPObejct>

</root>
```

Figure 15: Beispeil für Kinder Elemente eines XML Element

Attribute Attribute sind wiederum strukturierte Kinder Elemente, die im Rahmen eines XML Knotens bzw. Elements dargestellt werden. Attribute besitzen Bezeichner und Textwert (siehe Figure 16).

```xml
<?xml version="1.0" encoding="UTF-8"?>

<STEPObject name="object name" attribute1="data1" length="23"
    width="45" higth="56" mesurment1="21" color="black" relation2next="xxxx"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2001/XMLSchema STEP_XMLSchema.xsd ">

    Das ist ein STEP XML-Object!

</STEPObejct>
```

Figure 16: Attribute innerhalb eines Beispiel XML-Elementes

Inhalte/Texte Für jedes XML Element bzw. Knoten und Attribut ist ein Inhalt in Form eines Text-Strings zugeordnet. Es kann aber auch ein leerer String sein. Bei einem XML-Knoten stehen die Textinhalte zwischen dem Anfang XML-Statement *<xmlKnote_name>* und dem Ende XML-Statement *</xmlKnote_name>*. Bei Attributen sind die Inhalte nach jedem Attributnamen zu finden. Sie werden zwischen zwei Gleichungszeichen in Anführungszeichen platziert.

Kommentare Man kann Kommentare in einer XML Datei anbringen. Die Kommentare fangen mit folgende Zeichenfolge "*<!–*" an, gefolgt von den eigentlichen Kommentarzeilen. Um das Kommentar zu beenden, muss folgende Zeichenfolge "*–>*" am Ende hinzugefügt werden.

Die XML Deklaration Eine XML Deklaration steht immer am Anfang jeder XML Datei. Sie besteht aus einer bis mehreren XML Zeilen in folgender Form *<?xml .... ?>*

Solche Deklarationszeilen beinhalten Pseudo-Attribute, die nicht zu Inhalten, die den Zweck einer XML Datei repräsentiert, gehören, sondern sind nur für Deklarationszwecke gedacht.

Die Pseudo-Attribute haben folgende Formatierung :
<?xml version="1.0" encoding="UTF-8"?>

W3C XML Schema Ein Schema im Allgemeinen ist eine Beschreibungssprache für die Struktur von bestimmten Daten bzw. Sprachen. Ein Schema definiert, wie die Struktur von Daten innerhalb einer Sprache aussehen soll. Beim XML-Schema handelt es sich um eine Meta-Meta Sprache. Mit anderen Wortern eine Beschreibung für die Beschreibungssprache und in diesem Fall heißt diese XML-Schema und XML-Inhalte.

Ein XML Dokument beinhaltet die Daten als Strings bzw. Zeichenfolgen in einer Baumstruktur und die Aufgabe eines XML-Schemas ist die Definition diesen baumartigen Struktur. Innerhalb der Software Entwicklung werden XML-Strukturdaten mithilfe eines XML Schemas (Datei) validiert bzw. nach Vollständigkeit und Korrektheit geprüft.

```
<?xml version="1.0"?>
<xs:schema id="root" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
                     xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="coordinate_group">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="coordinats" nillable="true" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent msdata:ColumnName="coordinats_Text" msdata:Ordinal="1">
              <xs:extension base="xs:string">
                <xs:attribute name="name" type="xs:string" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element name="root" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="coordinate_group" />
        <xs:element name="step_object">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="coordinate_group" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="relation2next" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="relation2next" type="xs:string" />
                  <xs:attribute name="relation2object" type="xs:string" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
```

Figure 17: Ausschnitt aus dem XML Schema für die Abbildung der STEP-Daten

2.4.4 Die Entwicklungssprache JAVA

Die Programmiersprache JAVA ist eine objektorientierte Programmiersprache und wurde von der Firma Sun Microsystems entwickelt. Um mit der Programmiersprache Java zu programmieren benötigt man zuerst das JDK "Java Development Kit", die das Java-Entwicklungswerkzeug darstellt, und die Java Laufzeit Umgebung JRE "Java Runtime Environment", um die Java-Programme ausführen zu können.

Die Programmiersprache Java ist plattformunabhängig, d.h Java-Programme sind auf jeder System- und Rechnerarchitektur lauffähig. Diese Eigenschaft der Plattformunabhängigkeit gilt auch für die verschiedenen Betriebssysteme wie Windows, OS X oder Linux.

In Figure 18 sehen wir ein Java-Programmcode Beispiel mit ein paar Programmier- und Java- Programmcod-Richtlinien.

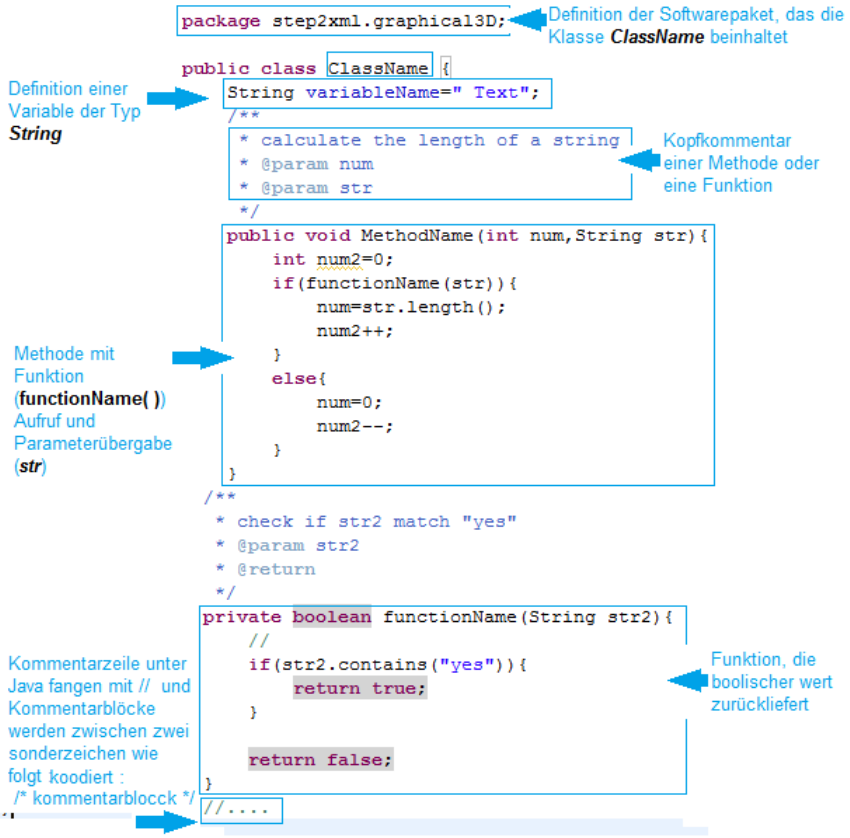

Figure 18: Die Programmiersprache JAVA

2.4.5 Java3D Bibliothek

Java3D ist ein Java-Bibliothek, die mehrere nützliche Funktionen und Klassen zur graphischen Modellierung in 2D und 3D beinhaltet. Die Bibliothek definiert Klassen für die Erzeugung verschiedener geometrischer Elemente wie "*Sphere*", "*Pyramid*" etc., die im Rahmen von vordefinierten (Java3D) *Canvas3D*-Objekten in einer 3D Modellierung *Universe* hinzugefügt bzw. zusammengestellt werden.

Diese vordefinierten geometrischen Elemente von Java3D sind ein wichtiger Bestandteil im STEP2XML-Tool. Damit werden die STEP-Produktmodellierungsdaten, die Teile eines ganzen Produktes darstellen, in eine 3D Modellierung erzeugt und vorgestellt.

Die modellierten Daten werden in einer 3D-Szene hinzugefügt und diese wird mithilfe der Funktion *theScene.Compile( )* gebildet und angezeigt.

2.4.6 iText Java-Bilbliothek zu PDF generierung

iText ist eine frei verfügbare Java-Bibliothek *OpenSource*-Projekt. iText ist für die dynamische Generierung von PDF-Dateien mittels der Programmiersprache Java.

Es wird im Java-Projekt bei der Implementierung als Java-Bibliothek eingebunden und mittels verschiedenen im iText vordefinierten Methoden, Klassen und Funktionen können PDF-Datei automatisch generiert und manipuliert werden.

2.5 Indirekt angewandte Technologien

Bei der indirekt angewandten Technologien handelt es sich um Werkzeuge und Technologien, die der Entwickler bei der Implementierung indirekt unterstützen. z.B. UML oder FreeCAD zur Produktmodellierung und Erstellung von STEP-Inhalten als Versuch bzw. Test Eingabedaten in den entwickelten Systemen.

2.5.1 Eclipse

Eclipse ist eine Entwicklungsumgebung für das Entwickeln in verschiedenen Progr-ammier- und Skriptsprachen wie z.B. Java, Python und Perl.

Für die Entwicklung des Tools STEP2XMLGenerator in Java wurde die Eclipse Variante Eclipse Java EE IDE for Web Developers eingesetzt.

Eclipse ist eine Plattform, welche aus mehreren Subsystemen, die PlugIns heißen, besteht. Die Subsysteme sind über Laufzeitsystem-Plattforma integriert (siehe Figure 19).

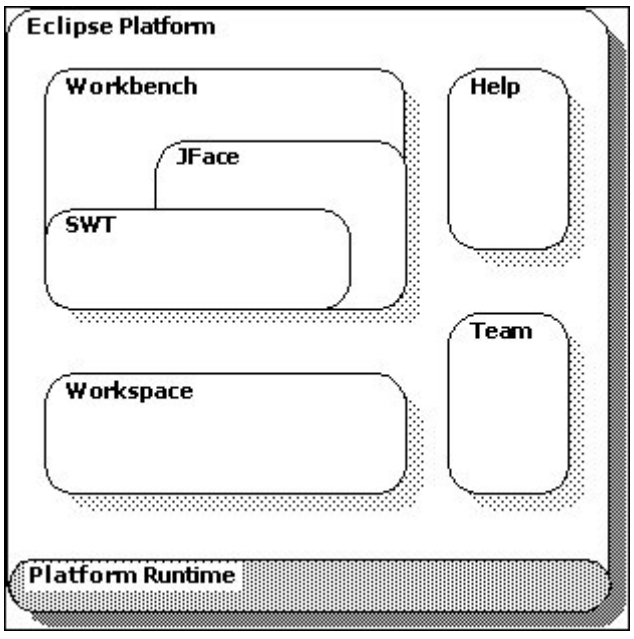

Figure 19: Eclipse als Laufzeitsystem mit verschiedenen Subsystemen[EclipseHD]

Die Arbeitsfläche Workbench von Eclipse, dient zur Verwaltung, Navigation und Erzeugung von Workspace Inhalten. Die Eclipse Arbeitsfläche beinhaltet verschiedene Perspektiven *Perspectives* mit Sichten *Views* mit beliebigen Entwicklungswerkzeugen und Editoren, die flexibel einschalt- oder ausschaltbar

sind.

Solche Werkzeuge und Editoren dienen zur Erleichterung und assistieren den Entwickler bei der Software-Entwicklung .

2.5.2 UML Unified Modelling Language

UML ist eine Abkürzung für Unified Modelling Language. UML wird für die graphische Modellierungssprache und Darstellung von Software- und Systemkomponenten im Rahmen einer Spezifikation und für Konstruktionen wie Softwareentwurf oder -dokumentation verwendet.

UML wurde in den 1990er Jahren definiert und diente damals als Basis und Unterstützung für die objektorientierte Softwareentwicklung. Im August 1999 hat die Firma OMG mit der Entwicklung von UML2 angefangen.

Unter UML sind 14 verschiedene Diagrammarten definiert:

1. das Klassendiagramm
2. das Kompositionsstrukturdiagramm
3. das Verteilungsdiagramm
4. das Komponentendiagramm
5. das Objektdiagramm
6. das Paketdiagramm
7. das Profildiagramm
8. das Aktivitätsdiagramm
9. das Anwendungsfalldiagramm
10. das Interaktionsübersichtsdiagramm
11. das Kommunikationsdiagramm
12. das Sequenzdiagramm
13. das Zeitverlaufsdiagramm
14. das Zustandsdiagramm

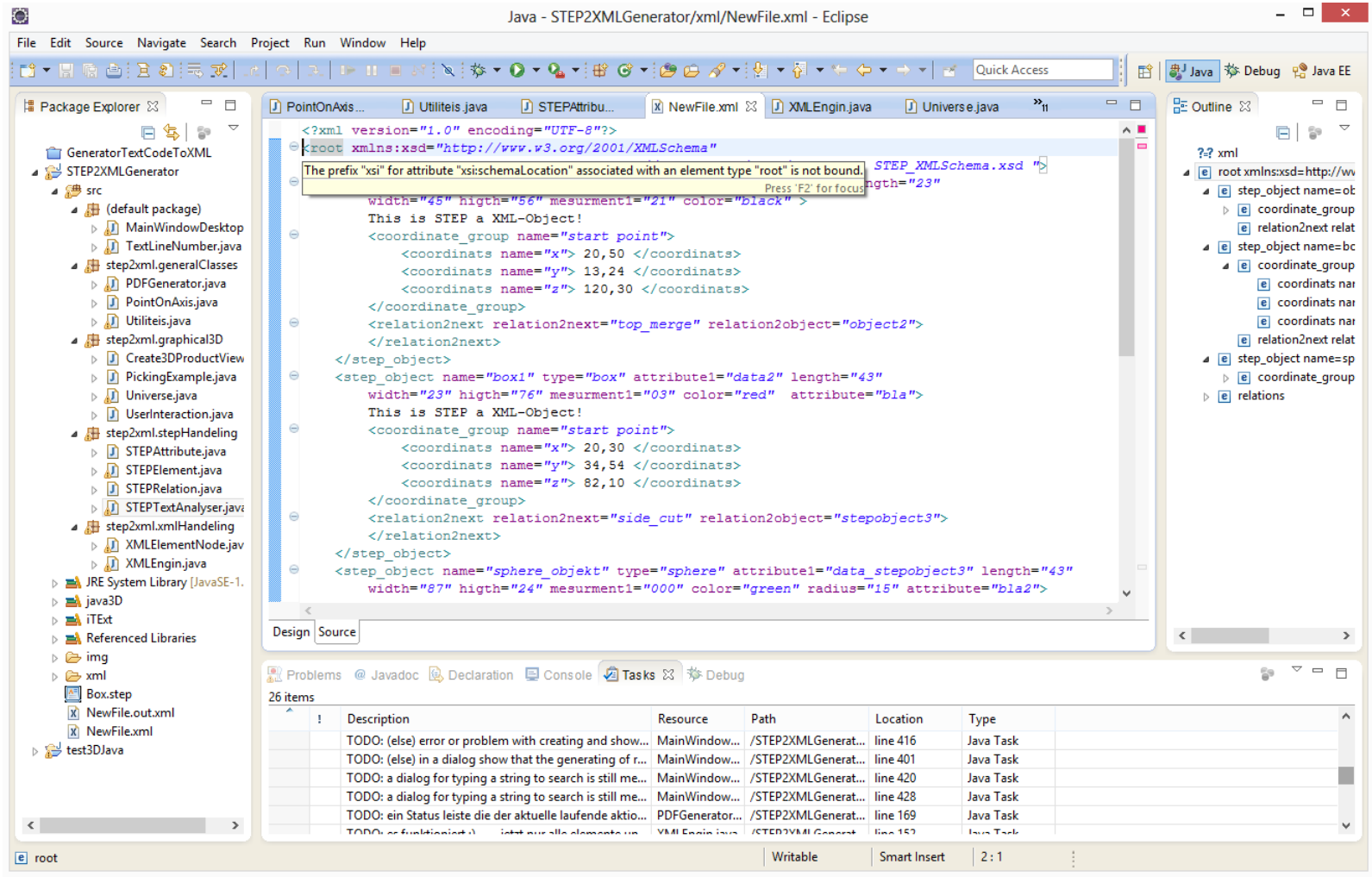

Figure 20: Eclipse Entwicklungsumgebung

Im folgenden sollen einige Diagrammarten näher erläutert werden:

- Klassendiagramm : Bei einem Klassendiagramm werden die Klassen einer Software graphisch dargestellt. In solch einer Darstellung werden die Attributte und Methoden einer Klasse hervorgehoben. Die Beziehungen wie Vererbung, Generalisierung, Spezialisierung etc. zwischen den Klassen können dargestellt werden.

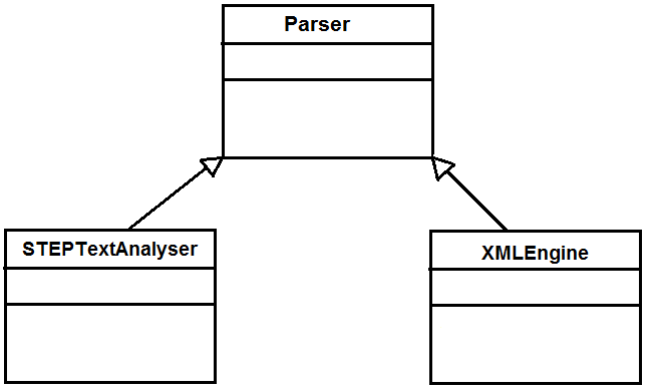

Figure 21: Beispiel für Klassendiagramm

- UseCase-Diagramm : Diese Art von Diagramm wird auch Anwendungsfalldiagramm genannt und es werden damit die Benutzer-System Schnittstellen definiert. Dabei beinhaltet ein Anwendungsfalldiagramm die Anwendungsfälle und ihre Akteure.

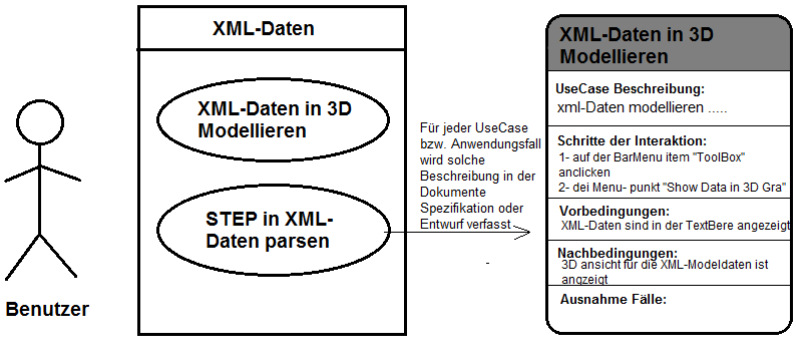

Figure 22: Beispiel für UseCase-Diagramm

- Sequenzdiagramm : Es stellt das Verhalten einer Interaktion in einem System dar. Es werden dynamisch die Aktivitäten einer Interaktion zwischen Anwender und System oder zwischen den Komponenten eines Systems dargestellt. Dabei wird der Nachrichtenaustausch zwischen den Aktivitäten einer Interaktion auf einer Lebenslinie je Aktivi-tät dargestellt.

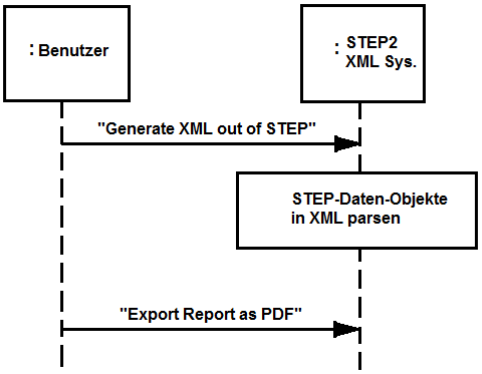

Figure 23: Beispiel für Sequenzdiagramm

2.5.3 FreeCAD Editor

FreeCAD ist ein Opensource Editor zur 2D und 3D Modellierung von verschiedenen Konstruktionen. FreeCAD wurde von Juergen Riegel, Werner Mayer und Yorik van Havre zwischen 2001 und 2011 entwickelt. FreeCAD kann die Konstruktionen in vielen verschiedenen Formaten bzw. Standards wie STEP, IGES Format, Autodesk DXF und Inventor V2 exportieren.

In dieser Arbeit habe ich für die Zeichnung verschiedene 3D Teil-Konstruktionen die Version 0.13 für Windows 32-bit verwendet. Dabei wurden verschiedene 3D Modellierungen in STEP exportiert, und als Beispiel und Referenz für STEP Daten verwendet. Die Modellierungen wurden als Eingabe STEP Daten für den Software *STEP2XMLGenerator* verwendet.

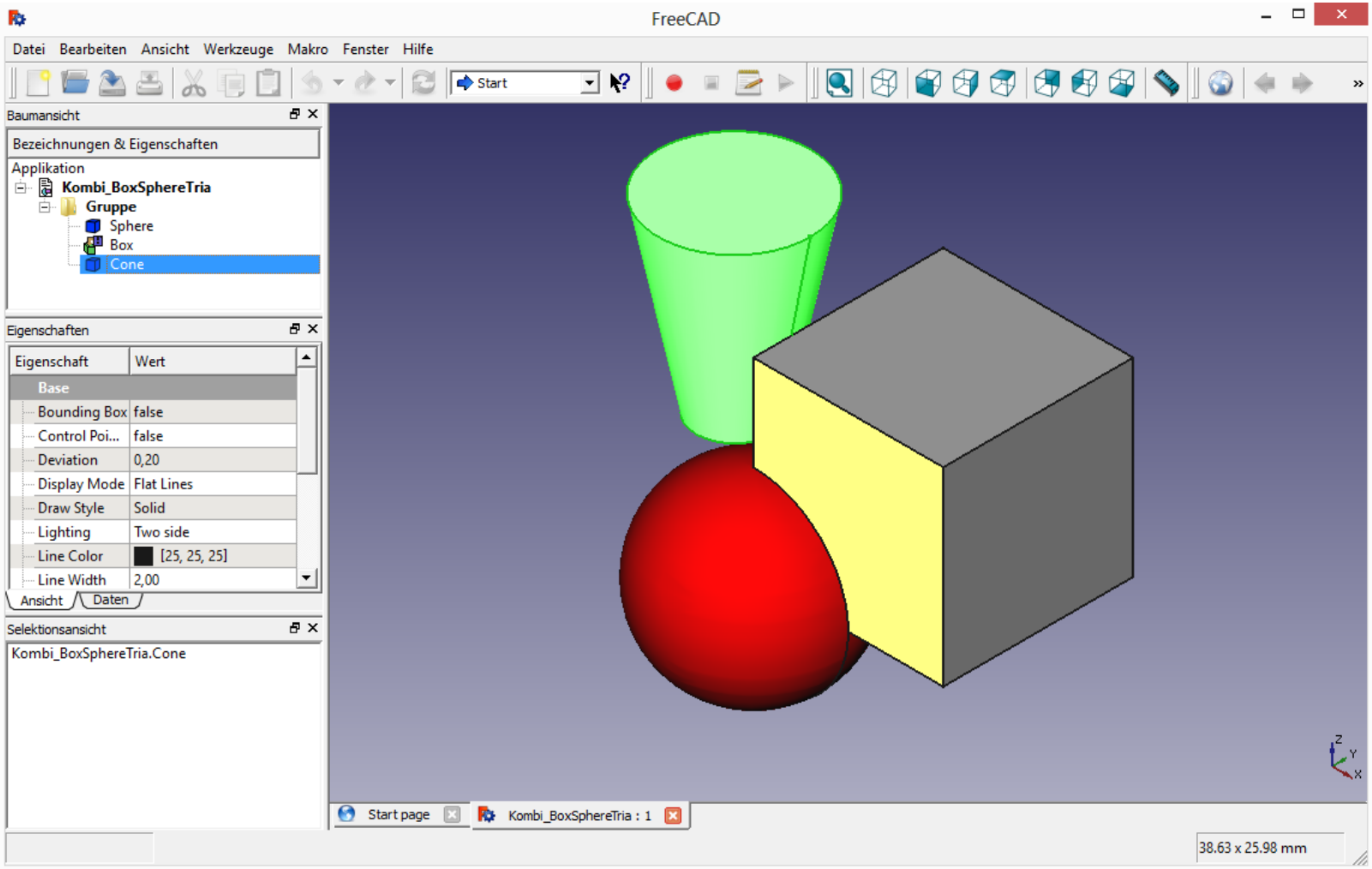

Figure 24: FreeCAD Editor

3 Methodologie

In diesem Kapitel wird auf das Vorgehen in der Entwicklung der Software eingegangen und die entsprechenden Arbeitsschritte erläutert. Die Arbeitsschritte in diesem Kapitel sind technisch orientiert, weshalb werden sie auf einer technischen Ebene behandelt und erläutert.

3.1 Erstellung des CAD-Designs

Mithilfe der OpenSource Software FreeCAD werden Zeichnungen von geometrischen Elementen wie Würfel, Toris, Pyramide etc. zusammen erstellt. Die Zeichnungen werden über FreeCAD als STEP-Datenformat in eine STEP-Datei exportiert. Diese Dateien werden für die Eingabe im STEP2XMLGenerator-Programm verwendet.

Im Rahmen dieser Forschung zu einem Parsen Einsatzes der STEP-Daten in XML-Strukturierten Daten, werden nur primitive geometrische 3D-Objekte behandelt. Die Primitiven sind Pyramide, Kegel, Sphere, Zylinder, Würfel und Torus und werden für eine 3D-Zeichnung mithilfe von FreeCAD modelliert. Es wurden mehrere CAD-Zeichnungen, die zum einen die einzelnen Primi-tiven beinhalten und zum anderen eine Kombination mehrerer primitiver geometrischer 3D-Objekte sind, erstellt.

Die modellierte CAD-Zeichnung wird dann mithilfe von FreeCAD als STEP-Datei exportiert und im Rahmen dieser Studie betrachtet bzw. analysiert um die STEP-Struktur näher anhand von Beispielen zu interpretieren. Diese Dateien werdenu zu einem späteren Zeitpunkt als Eingabedateien für den STEP-Daten Parser bereit gestellt.

3.2 STEP-Daten Parser

In zweiten Schritt wird der Parser der STEP-Daten erzeugt. Die Hauptklasse dieses Parsers heißt *STEPTextAnalyser*, welche die STEP-Daten jeder einzelne geometrischen Produktteil aus der STEP-Daten analysiert wird und entsprechend nach wichtigen Daten wie Messdaten, Koordinaten etc. , die für die spätere Remodellierung benötigt werden, durchsucht.

1. Die Java-Klasse *STEP*Element, bildet alle Daten aus den STEP-Textzeilen ab. *STEPElement* beinhaltet Platzhalter für Messdaten, Koordinatenlisten und weitere Informationen, die einen geometrischen Primitiven aus der STEP-Datei repräsentiert.

2. Objekte-Liste wird für das Speichern der Java-Klassenobjekte der Klasse *STEPElement.java* erzeugt. Die STEP-Daten werden in die Attribute und Arrays innerhalb der Java-Klassenobjekte der Klasse *STEPElement* gespeichert. Alle erzeugten STEPElement-Objekte werden in dieser ArrayList in Java *ArrayList<STEPElement>* abgelegt und für die nächste Schnittstelle bereitgestellt

Für STEP-Daten Parser ist die Funkition *analyseTheSTEPFileLines(File aSTEPFile)* die Hauptfunktion. Er analysiert die STEP-Daten zeilenweise und parst anhand bestimmter Struktur alle wichtigen Daten für die Re-modellierung.

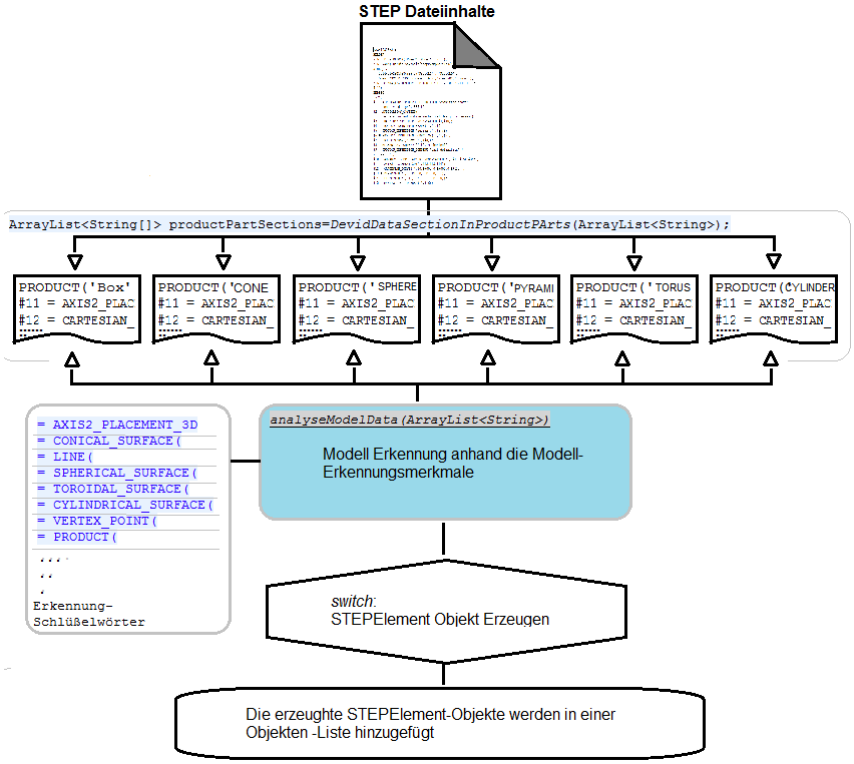

Figure 25: graphische Verlauf der Funktion *analyseSTEPDataLines()*

3.3 Entwurf der XML-Struktur

Es handelt sich in diesem Abschnitt um die Definition und dem Entwurf der XML-Struktur. Damit die STEP-Daten aus der *STEPElement*-Objekten als XML-Struktur modelliert und als eine XML-Datei exportiert werden können, benötigt man zuerst die Definition für ein XML-Baum. Darunter ist die einen Überlegung zu verstehen, wie die STEP-Daten als XML-Elemente, -Attribute etc. zu modellieren sind. Dafür wird ein XML-Schema erstellt und die Java-Klasse *XMLElementNode.java* erzeugt.

- <*step_ object*> Repräsentation für STEP geometrische Primitiven. Die einheitsbezogenen Daten wie Messdaten, Name, Art, Farbe und andere objektbezogene Daten werden in Form von XML-Attributen angelegt.
  - <*coordinate_ group*> Ist ein Vaterknoten für Koordinaten Knoten <*coordination*>
  - <*coordination*> wird zur Darstellung einer Koordinate auf dem Koordinatensystem verwendet. Es beinhaltet ein Attribut "name", welches den Koordinaten der X,Y oder Z-Achse zu zuordnen ist und den Wert der Koordinate, spezifiziert.

3.4 STEP2XML-Parser

Dieser Arbeitsschritt stellt die Entwicklung einer wichtigen Komponente im gesamten System dar. Der XML Parser ist für die Umwandlung der STEPElement - Objektdaten in XML-Elemente und XML-Struktur zuständig. Er nennt sich *XMLEngine.java* und beinhaltet folgende Funktionen:

- parsen der Daten aus *STEPElement*-Objekte in XML-Elemente und Attribute eventuell in eine XML-Struktur
  - Parsen von XML-Daten aus der Eingabe XML-Datei in *XMLElementNode*- Objekte
  - Erstellen und Ablesen von XML-Datei-Inhalten

Für das Erstellen der XML-Datei für die Abbildung der STEP-Daten in XML-Elemente ist die Funktion *generateXMLFileContentOutOfSTEPElements()* zuständig.

Die Funktion erzeugt pro geparster STEP-Datei eine XML-Datei mit dem Name = *Originale STEP-Dateiname + "_generatedXML.xml"*. Danach wird die Funktion *createXMLFileWithXMLContent(String, ArrayList<XMLElement -Node>)* aufgerufen, um für die STEP-Daten in einer zusammenhängenden XML-Struktur die passenden XML-Nodes bzw. XML-Elemente zu generieren.

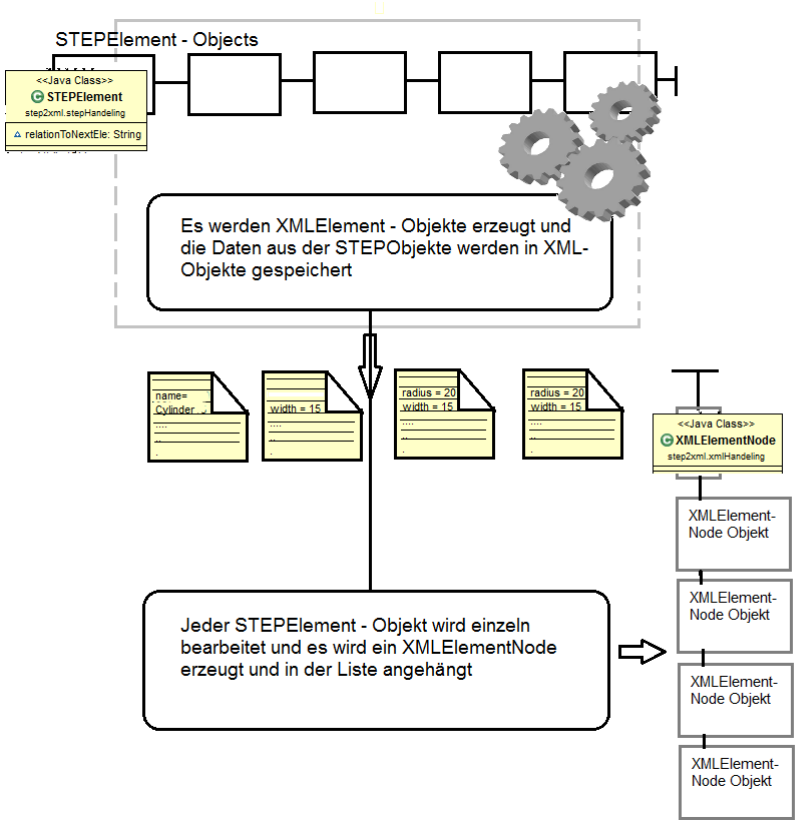

Figure 26: Funktionsweise der Parser *XMLEngine.java*

3.5 Report als PDF Generieren

Der fünfte Arbeitsschritt befasst sich mit dem Programmieren der *PDFGenerator.java* zur Generierung von Berichte in PDF-Format. Die Ergebnisse der Generierung von XML-Daten aus STEP-Daten werden in einem Bericht *Report* aufgelistet. Es beinhaltet hauptsächlich die STEP-Daten sowie die generierten XML-Daten. Desweiteren sind Metadaten wie Name der Dateien mit Dateipfade, Datum und Uhrzeit und eventuell eine graphische Abbildung der Remodellierung der Daten vorhanden.

Für dieses Modul kommt die Funktion *generatePDF (filename,STEPDataStr, XMLDataStr, Titel)* ins Spiel. Es erzeugt ein Bericht mit den Ergebnissen der XML-Daten Generierung. Es beinhaltet auch die STEP Inhalte und dazu Titel, Beschreibung und Dateiname als Einführung am Anfang des Dokumentes.

3.6 Erweitern des XML-Parsers

Im Rahmen der XML-Parser *XMLEngine.java* werden die Funktionen, die für das Parsen von XML-Daten aus der eingegebenen Ergebnis XML-Datei zuständig sind, erweitert. Es werden *XMLElementNode* - Objekte für die XML Elemente und Attribute erstellt. Die Werte der XML Elemente und Attribute werden in Variablen und Arrays in diesen Java-Objekten gespeichert. Zweck der Erzeugung von XMLElementNode - Objekte ist die 3D graphische Modellierung der XML-Daten.

3.7 3D-Modellierung der XML-Daten

Als letzter Schritt werden die Funktionen für die 3D-Darstellung bzw. Remodellierung der XML-Daten entwickelt. Die 3D-Darstellung wird in einem 3D-Viewer angezeigt. Dafür werden Interaktion-Funktionalitäten entwickelt, wie das Rotieren des modellierten Produktes mithilfe der Maus in XYZ-Achsen und das Heran- und Herauszoomen.

Die Modelldaten werden aus der Eingabe XML-Datei geparst. Dannach werden die Daten mithilfe der Funktion *createSceneGraph(SimpleUniverse su)* für die Modellierung der einzelnen Primitiven der STEP-Produktteile neu in 3D-Objekte verwendet. Daraufhin werden 3D-Objekte in der 3D-Szene hinzugefügt und am Ende wird die komplette 3D-Zeichnung angezeigt.

3.8 Verbesserung der CAD-Feature Extraktion

Anschließend wird die CAD-Feature Extraktion verbessert bzw. wird mit der Modell-Erkennung in späteren Kapiteln (Siehe dazu Abschnitt "graphische Modell-Erkennung" unter Implementierung) erläutert.

Mit CAD-Feature ist jedes einzelne STEP-Produktteil gemeint. Es handelt sich hier um die primitiven geometrischen Objekten, wobei das CAD-Feature den Abschnitt an STEP-Daten, die jeder dieser Primitiven beschreibt bzw. definiert.

Dieses Erkennungsverfahren teilt die STEP-Daten in mehrere Abschnitte, die jeweils ein Primitiv wie Box, Sphere, Zylinder etc. repräsentieren. Das Verfahren extrahiert aus jedem dieser Abschnitte die relevanten Daten wie Messdaten, Koordinaten oder weitere Beschreibungsdaten.

4 Implementierung

In diesem Kapitel wird die technische Seite dieser Arbeit vorgestellt. Dabei werden drei Themen behandelt. Als erstes werden die technischen Aspekte sowie Interpretationen des STEP Standards dargestellt und dabei werden folgende Schwerpunkte behandelt

- die Schlüsselwörter innerhalb des STEP Standards
- die Struktur einer STEP-Datei
- die Interpretation und das Ablesen der Daten einiger wichtiger Inhalte innerhalb einer STEP Struktur

Dannach wird auf den Grobentwurf eingegangen in dem STEP-Elemente als Klassenobjekte und als Systemarchitekt behandelt werden. Der Feinentwurf enthält Aspekte wie STEP geometrische Objekte als XML-Elemente $<step\_objekt>$, XML-Struktur, die Kernfunktionen. Anschließend werden die einzelnen Klassen bzw. Module begleitend von der graphischen Abbildung des Klassendiagramms beschrieben.

4.1 HEADER und DATA Abschnitte der STEP Struktur

Jeder STEP Dateiinhalt besteht aus zwei Abschnitten. Der erste Abschnitt beinhaltet allgemeine Informationen über das zu modellierende Produkt bzw. die graphische Zeichnung und wird mit "*HEADER Section*" bezeichnet.

Solche Informationen beinhalten Namen der Editor Software, mit dem die jeweilige Modellierung und die dazugehörigen STEP Daten erzeugt wurden, das Erstellungsdatum der Daten, die Programmversion und der Name des Dateiverzeichnis.

Der zweite Abschnitt der STEP Struktur ist der *DATA* Abschnitt. Es beinhaltet die eigentliche STEP Daten zur Modellierung. Es beinhaltet die Koordinaten auf der XYZ-Achse, die Zeichnungspunkte, wichtige Messdaten wie Länge oder Breite der Modellierungsobjekte.

Diese Abschnitte auch Sections genannt beginnen und enden mit zwei Schlüsselwörtern, es fängt mit dem Name des Abschnittes wie "*HEADER*;" oder "*DATA*;" an und endet mit "*ENDSEC*;" welches das Ende des Abschnittes bezeich-net (siehe Figure 27)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('FreeCAD Model'),'2;1');
FILE_NAME('D:/DiplomArbeit/stepExampleFiles/kegel.step',
  '2014-01-28T14:38:07',('FreeCAD'),('FreeCAD'),
  'Open CASCADE STEP processor 6.3','FreeCAD','Unknown');
FILE_SCHEMA(('AUTOMOTIVE_DESIGN_CC2 { 1 2 10303 214 -1 1 5
4 }'));
ENDSEC;
DATA;
#1 = APPLICATION_PROTOCOL_DEFINITION('committee draft',
  'automotive_design',1997,#2);
#2 = APPLICATION_CONTEXT(
  'core data for automotive mechanical design processes');
#3 = SHAPE_DEFINITION_REPRESENTATION(#4,#10);
#4 = PRODUCT_DEFINITION_SHAPE('','',#5);
#5 = PRODUCT_DEFINITION('design','',#6,#9);
#6 = PRODUCT_DEFINITION_FORMATION('','',#7);
#7 = PRODUCT('Cone','Cone','',(#8));
#8 = MECHANICAL_CONTEXT('',#2,'mechanical');
#9 = PRODUCT_DEFINITION_CONTEXT('part definition',#2,'design');
#10 = ADVANCED_BREP_SHAPE_REPRESENTATION('',(#11,#15),#113);
#11 = AXIS2_PLACEMENT_3D('',#12,#13,#14);
#12 = CARTESIAN_POINT('',(0.E+000,0.E+000,0.E+000));
#13 = DIRECTION('',(0.E+000,0.E+000,1.));
#14 = DIRECTION('',(1.,0.E+000,-0.E+000));
#15 = MANIFOLD_SOLID_BREP('',#16);
....
...
..
.
#127 =
COLOUR_RGB('',0.800000011921,0.800000011921,0.800000011921);
#128 = CURVE_STYLE('',#129,POSITIVE_LENGTH_MEASURE(0.1),#127);
#129 = DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
ENDSEC;
END-ISO-10303-21;
```

Figure 27: Abschnitte “Section” einer Beispiel STEP Datei

Die *DATA-Section* (siehe Unterabschnitt “4.1 *HEADER* und *DATA* Abschnitte der STEP Datei-Struktur”) ist aus mehreren Teilen aufgebaut, wobei jedes Teil STEP-Daten über geometrischen Elemente beinhaltet.

Bei der Erkennung der Basis geometrische Objekte bzw. der Primitiven werden verschiedene wichtige STEP-Schlüsselwörter sowie STEP-Zeilen im Rahmen der Arbeit bzw. Prozess des Parsen untersucht. Beim Parsen werden die STEP-Daten genauer betrachtet, um die Länge, Breite, Tiefe, Radius und Position daraus berechnen zu können.

Bei allen STEP-Elementen werden jeweils erste Definitionen von "*Axis2_placement*" nach dem Produkt-Name wie z.b die Zeile “*#33 = PRODUCT('Box','Box',...*”

verwendet, um die Position der STEP-Element auf der XYZ-Achse zu bestimmen

#11 = AXIS2_PLACEMENT_3D(",#12,#13,#14);

#12 = CARTESIAN_POINT(",(0.E+000,0.E+000,0.E+000));

Die erste Position-Definition im Dokument kann ignoriert werden„ weil es für die Remodellierung nicht relevant ist.

#11 = AXIS2_PLACEMENT_3D(",#12,#13,#14);

#12 = CARTESIAN_POINT(",(0.E+000,0.E+000,0.E+000));

Die zweite AXIS2_PLACEMENT_3D ist für die Berechnung der Position zu beachten. Dieser Fall tritt nur bei den ersten STEP-Elementen auf.

4.2 Die Interpretation wichtiger Daten aus den STEP Dateiinhalten

Bei der Interpretation wichtiger Daten für die geometrischen primitive Objekte aus den STEP Dateiinhalten werden folgende Schlüsselwörter betrachtet:

BOX

Durch eine weitere CARTESIAN_POINT() - Koordinaten-Definition kann man die Länge, Breite der Box berechnen. Beispiel:

#376 = PLANE(",#377);
#377 = AXIS2_PLACEMENT_3D(",#378,#379,#380);
#378 = CARTESIAN_POINT(",(9.,10.,0.E+000));
Um die Maße der BOX abzulesen, betrachten wir Folgendes:

- *CARTESIAN_POINT(",(9.,10.,0.E+000));* mit Startpunkt *CARTESIAN_POINT(",(9.,0.E+000,0.E+000));* besteht Wertunterschied mit der Ursprung Position-Definition an der zweite Stelle und das ergibt den Wert der Breite: *Width* 10-0=10
- *CARTESIAN_POINT(",(19.,0.E+000,0.E+000));* mit Startpunkt *CARTESIAN_POINT(",(9.,0.E+000,0.E+000));* an der erste Stelle daraus ergibt sich die Differenz 19 - 9 = 10 das ist die Länge der BOX
- Das gleiche Verfahren wird wiederholt und dabei ergibt sich die Differenz zum Ursprung an der dritte Stelle

  #403 = AXIS2_PLACEMENT_3D(",#404,#405,#406); mit *#404 = CARTESIAN_POINT(",(9.,0.E+000,10.));* 10-0 = 10 ist die Höhe der Boxes

CONE

Bei *CONE* müssen beide Kreisdefinitionen gesucht werden. Man kann aus dem Schlüsselwort: *CIRCLE(",#396,4.);* den Wert des Radius ablesen, in diesem Fall ist das 4.

#396 ist eine Adresse für eine STEP-Zeile, die die Positionkoordinaten auf der XYZ-Achse beinhaltet.

#396 = AXIS2_PLACEMENT_3D(",#397,#398,#399);
#397 = CARTESIAN_POINT(",(-3.,2.,15.));
Die Höhe des Kegels wird in der STEP-Zeile *LINE()* definiert
#63 = LINE(",#64,#65);
und die Adresse #64 für eine STEP-Zeile, die den Wert der Höhe des Kegels beinhaltet
#64 = CARTESIAN_POINT(",(0.E+000,10.)); =10

SPHERE

Mit Hilfe des Schlüsselworts SPHERICAL_SURFACE kann man den Wert des Radius ablesen

SPHERICAL_SURFACE(",#23,5.); = 5 und gleich in der darauf folgenden Zeile der Definition die Koordinaten der SPHERE

#22 = SPHERICAL_SURFACE(",#23,5.);
#23 = AXIS2_PLACEMENT_3D(",#24,#25,#26);
#24 = CARTESIAN_POINT(",(3.,2.,6.));

TORUS

Bei TORUS sind die zwei Radien aus dem Schlüsselwort *TOROIDAL_SURFACE* zu parsen, also in diesem Fall sind die Werte der Radien 10 und 2.

#22 = TOROIDAL_SURFACE(",#23,10.,2.);
#23 = AXIS2_PLACEMENT_3D(",#24,#25,#26);
#24 = CARTESIAN_POINT(",(-19.,7.,-1.));

CYLINDER

Bei CYLINDER sind Schlüsselwörter bzw. Daten wichtig, die die Länge und den Radius darstellen. Der Radius wird mithilfe von *CYLINDRICAL_SURFACE* definiert,

#195 = CYLINDRICAL_SURFACE(",#196,2.); welches in diesem Fall 2 ist.

Direkt nach der Zeile mit dem Radius folgt die Definition der Zylinderposition

#195 = CYLINDRICAL_SURFACE(",#196,2.);
#196 = AXIS2_PLACEMENT_3D(",#197,#198,#199);
#197 = CARTESIAN_POINT(",(-5.,17.,4.));

Die Höhe des Zylinders wird mithilfe des Schlüsselwortes *LINE* festgelegt und wird in folgende STEP Zeilen definiert

#201 = LINE(",#202,#203);
#202 = CARTESIAN_POINT(",(0.E+000,10.));

PYRAMID

Aufgrund der Tatsache, dass eine Pyramide aus 5 Punkten besteht, wird sie in STEP aus genau fünf Punkten *VERTEX_POINT* in der STEP Struktur definiert.

#24 = VERTEX_POINT(",#25);
#25 = CARTESIAN_POINT(",(15.,27.,30.));

4.3 Grobentwurf

4.3.1 STEP-Elemente als Klassenobjekte

Die STEP-Elemente aus der STEP-Datei werden in Java-Klassenobjekte abgebildet. Die modellierungsrelevanten STEP-Daten wie Messdaten und Koordinaten werden in definierten Variablen innerhalb der Klassenobjekte gespeichert. Die Objekte für die Abbildung der STEP-Daten werden aus der Klasse *STEPElement* instanziiert. Alle erzeugten Klassenobjekte, die die STEP-Objekte

bzw. Produktteile abbilden, werden innerhalb einer Liste abgelegt, um es für spätere Bearbeitungsprozesse im Tool bereitzustellen.

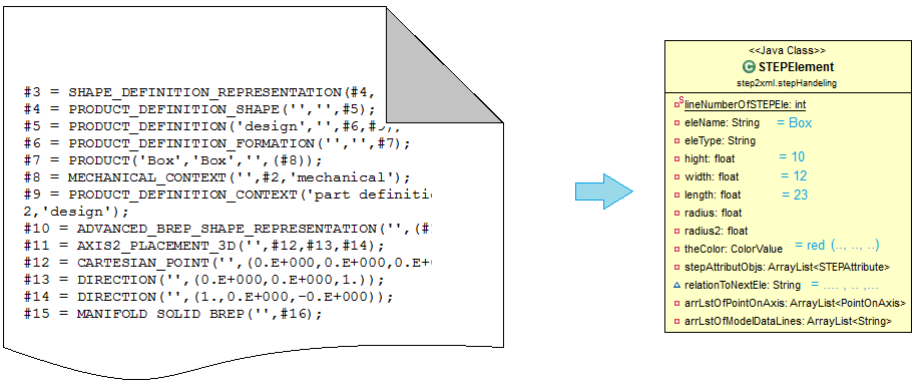

Figure 28: Abbildung der STEP-Objekte in Klassenobjekte *STEPElement*

4.3.2 Systemarchitektur

Wie in der Figure 29 zu sehen ist, wird die Systemarchitektur zusammengefasst dargestellt. Die Architektur teilt sich in fünf Abschnitte:

- Eingabe als STEP Datei zum Parsen und Analysieren der STEP-Daten
- Parser-Modul, um die STEP Daten auf Textebene zu analysieren und um daraus die wichtigen Informationen über die STEP graphische Objekte und Relationen abzulesen.
- Die Informationen aus den STEP Daten werden in JAVA-Klassenobjekte umgewandelt.
- Als letzter Schritt werden diese Klassenobjekte in XML Datenstrukturen geparst bzw. abgebildet.
- Die Ausgabe die XML Daten als XML-Datei.

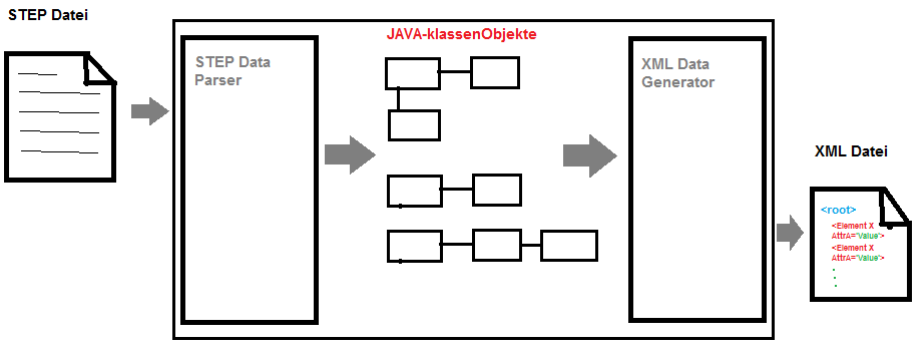

Figure 29: STEP2XMLGenerator Systemarchitektur

STEP Datei

Die Modellierung der Produkte wird im Rahmen STEP-Datei mithilfe der Beschreibungssprache STEP festgehalten. STEP Datei beinhaltet die zu darstellenden geometrische Objekten, die durch STEP Schlüsselwörter, Parameter etc. beschrieben sind.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('FreeCAD Model'),'2;1');
FILE_NAME('D:/DiplomArbeit/stepExampleFiles/Kombi_BoxSphereTri
a.step',
  '2014-01-28T14:36:39',('FreeCAD'),('FreeCAD'),
  'Open CASCADE STEP processor 6.3','FreeCAD','Unknown');
FILE_SCHEMA(('AUTOMOTIVE_DESIGN_CC2 { 1 2 10303 214 -1 1 5
4 }'));
ENDSEC;
DATA;
#1 = APPLICATION_PROTOCOL_DEFINITION('committee draft',
  'automotive_design',1997,#2);
#2 = APPLICATION_CONTEXT(
  'core data for automotive mechanical design processes');
#3 = SHAPE_DEFINITION_REPRESENTATION(#4,#10);
#4 = PRODUCT_DEFINITION_SHAPE('','',#5);
#5 = PRODUCT_DEFINITION('design','',#6,#9);
#6 = PRODUCT_DEFINITION_FORMATION('','',#7);
#7 = PRODUCT('Box','Box','',(#8));
#8 = MECHANICAL_CONTEXT('',#2,'mechanical');
#9 = PRODUCT_DEFINITION_CONTEXT('part definition',#
2,'design');
#10 = ADVANCED_BREP_SHAPE_REPRESENTATION('',(#11,#15),#345);
#11 = AXIS2_PLACEMENT_3D('',#12,#13,#14);
#12 = CARTESIAN_POINT('',(0.E+000,0.E+000,0.E+000));
#13 = DIRECTION('',(0.E+000,0.E+000,1.));
#14 = DIRECTION('',(1.,0.E+000,-0.E+000));
#15 = MANIFOLD_SOLID_BREP('',#16);
```

Figure 30: Ausschnitt aus einer STEP Datei

STEP DATA Parser

Ein Parser ist ein Computerprogramm oder Teilprogramm, das für die Umwandlung, Zerlegung und Strukturierung von beliebig eingegebenen Daten in ein lesbares und sinnvoll brauchbares Format für die Weiterbearbeitung zuständig ist. (siehe auch 2.3.1 Parser für weitere Informationen)

Diese Funktionen des Parsers werden bei diesem STEP-Data Parser auch umgesetzt. Dieser Parser zerlegt und wandelt die komplexeren Daten aus der STEP Datei so um, dass die Daten für die Weiterbearbeitung, die Abstraktion und Interpretation der Daten vorbereitet werden. Die geparsten Daten aus den STEP-Zeilen werden für die Erstellung der *STEPElement* - Objekte verwendet.

Die Funktionalität ist für das Einlesen und Parsen der STEP Dateien. Dieser Parser betrachtet die Elemente, Objekte und Parameter aus der STEP-Datei auf textueller Ebene und extrahiert die wichtigen Daten, Parameter, Werte und weitere beschreibende STEP-Elemente, um die Java-Objekte daraus zu erstellen. Dabei werden die Zusammenhänge und topologische Anordnung durch Daten aus der STEP-Datei bei der Erstellung von Java-Objektlisten berücksichtigt.

JAVA-Klassen Objekte

Eine Java-Klasse definiert eine Verallgemeinung für die zu instanszierten Objekte. Die Klasse beinhaltet eine Beschreibung für die Attribute, Eigenschaften und Funktionen. Eine Java-Klasse wird definiert, um bestimmte Datenobjekte in Java-Objekte abzubilden. In unserem Fall werden Java-Klassen dazu definiert, um die Datensätze in den STEP-Daten, die ein STEP geometrie Objekt in Form von Attributen etc. darstellen, im Rahmen von Java-Klassen Objekte abzubilden. Um mehrere Java-Klassen Objekte für die Weiterbearbeitung in den Programm zwischen zu speichern, werden die Objekte in Form von Objekt-Listen abgelegt.

Bei dieser Schnittstelle handelt es sich um die importierten bzw. geparsten Daten aus der STEP-Datei als Java Objektlisten. Jede dieser Listen beinhaltet Objekte, die folgende Daten darstellen :

- graphische STEP-Elemente
- Attribute, die sich auf die STEP-Elemente beziehen

XML DATA Generator

Der Generator ist ein "*Engine*" für die Übersetzung von Klassenobjekten, die STEP Daten darstellen, in einen XML strukturierten Baum. Der XML-DATA Generator wird mithilfe der Klasse *XMLEngine* realisiert. XMLEngine besteht aus mehreren Methoden, die verschiedene Tätigkeiten für die Bearbeitung diverser Bausteine der ganzen XML Struktur bewältigen.

Dabei werden die *STEPElement* Objekte in XML-Elemente abgebildet und *STEPAttribute* Objekte werden durch XML-Attribute ersetzt.

XML Datei

Die generierende XML-Ausgabe beinhaltet alle wichtigen Daten zur STEP geometrischer Objekte. Die STEP Modellierungsinformationen aus der eingegebenen STEP-Datei werden in drei verschiedenen XML-Elementen abgebildet (siehe Figur 12)

Diese sind:

<***step_ object***> ist zur Speicherung graphischer Objekt- / Einheitsbezogener Daten wie Messdaten, Name, Art, Farbe und andere objektbezogene Daten

<***coordinate_ group***> Ist ein Vaterknoten für Koordinaten Knoten <*coordination*>. Die Aufgabe von <*coordinate_ group*> ist das Zusammenfassen von mehreren Koordinatendaten, die im Zusammenhang stehen und zu einem bestimmten XYZ-Punkt auf dem Achsensystem gehören.

<***coordination***> dient zur Darstellung einer Koordinate auf dem Achsensystem. Es beinhaltet ein Attribut "name", womit definiert wird, auf welche Achse X,Y oder Z die Koordinate und ihre Werte zu zuordnen sind.

4.4 Graphische-Modell-Erkennung

Die STEP-Produktteile bzw-Elemente kann man nicht mit der einfachen Namensbezeichnung wie *box*, *sphere*, *cone* etc. textuell identifizieren, sondern muss anhand bestimmten Erkennungnsmerkmalen identifiziert werden. Solche Erkennungsmerkmalen sollen zwischen der Deklarationszeile bzw. *PRODUCT*-Zeile und der nächste Deklaration zeile der nächsten STEP-Elemente untersucht werden.

Als Beispiel nehmen wir die Zeile mit dem Schlüsselwort *CIRCLE* in *#22 = CIRCLE(",#396,4.)*; hier kann man erkennen, dass es sich um ein Produktteil mit der Definition eines Kreises als ein Bestandteil der graphischen Modellierung handelt.

Beim Würfel *(BOX)* oder bei anderen geometrischen Objekten wie z.B. der Pyramide, sind alle XYZ-Punkte auf der XYZ-Achse zu untersuchen und daraus die Ecken der geometrischen STEP-Teile, Messdaten zu berechnen bzw. zu kalkulieren.

4.4.1 Diagramm zur graphischen Darstellung der Modell-Erkennung Vorgehen innerhalb STEP-Daten

Hier handelt es sich um eine graphische Darstellung der Vorgehensweise der Interpretation der STEP Daten und wie anhand von Mustermerkmalen bzw. STEP-Datazeilen die verschiedenen geometrischen Teile (Box, Cone, Sphere ..etc) wieder erkannt werden können.

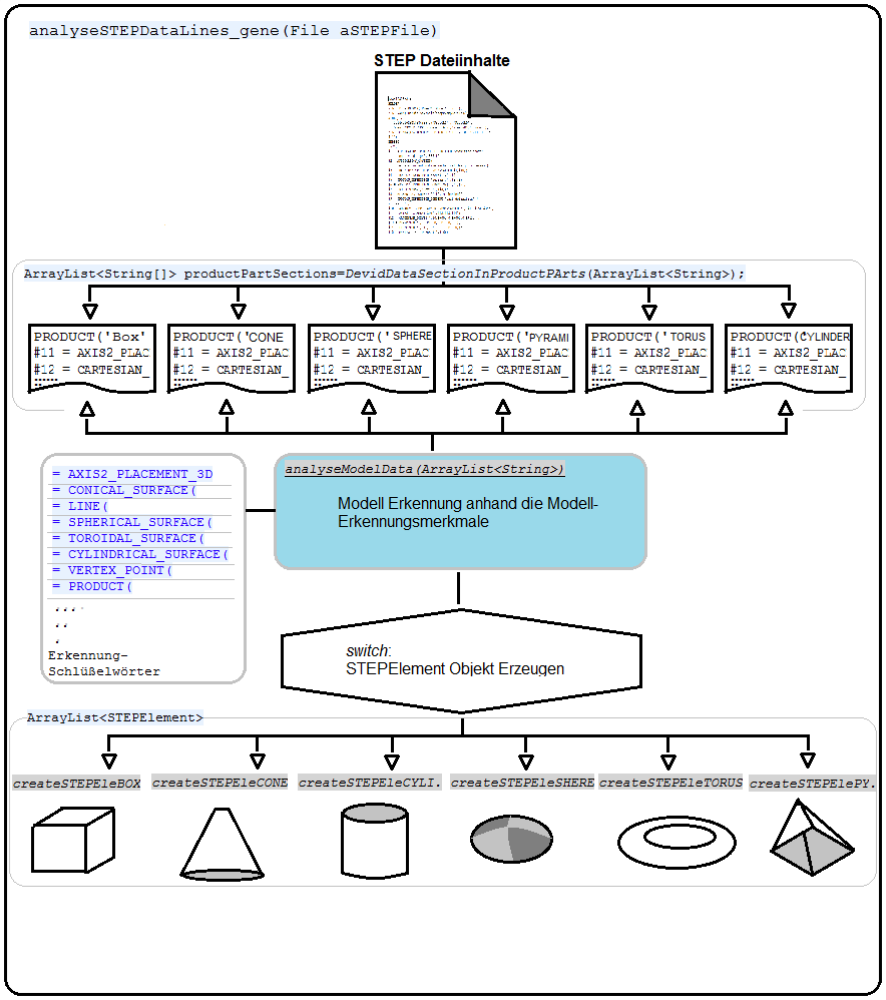

Figure 31: Modell-Erkennung

4.4.2 Architektur des Modell-Erkennungsalgorithmus

Zuerst werden die *STEP DATA Section* Zeilen in Abschnitte geteilt. Jeder dieser Abschnitte beginnt mit der Zeile " *#45 = PRODUCT(* " ..... Und endet genau vor der nächsten "*#78 = PRODUCT(* " - Zeile .

Alle Teile müssen innerhalb eines *ArrayList<ArrayList<String>>* hinterlegt werden. Jeder String wird noch einmal in mehrere Zeilen geteilt und als *ArrayList<String>* angelegt. Die Zeilen jedes Abschnitts (*ArrayList<String>*) werden untersucht und nach bestimmten STEP Zeilen durchsucht. Alle enthaltenen Muster-Merkmale werden in einer ArrayList vorgemerkt. Schließlich wird diese ArrayList analysiert, um heraus zu finden, um welches geometrisches Teil es sich hier handelt.

Nachdem anhand der Merkmale die STEP-Objekte identifiziert wurden, wird jeweils ein Objekt aus der Klasse STEPElement erzeugt und alle wichtigen Daten, die man innerhalb der Erkennung-Merkmale gefunden hat, in Form von

Variablen gespeichert.

4.4.3 Erkennungs - Merkmale:

Es handelt sich um -

- SPHERE: Wenn das STEP-Schlüsselwort " *#34 = SPHERICAL_SURFACE(*" vorhanden ist.
- CONE: Wenn das STEP-Schlüsselwort "*#67 = CONICAL_SURFACE(*" vorhanden ist.
- BOX: Anzahl der VERTEX_POINT ist genau 8 und damit ist eine BOX erkannt. Durch die *CARTESIAN_POINT* der *VERTEXe* werden die Größen der *BOX* berechnet
  - *#377 = AXIS2_PLACEMENT_3D(",#378,#379,#380);*
  - *#378 = CARTESIAN_POINT(",(9.,10.,0.E+000));*

  Damit werden die Maße *length*, *width* und *higth* anhand den XYZ-Punkten berechnet.
- CYLINDER: Wenn das STEP-Schlüsselwort "= *CYLINDRICAL_SURFACE(*" vorhanden ist
- TORUS: wenn die Zeile = *TOROIDAL_SURFACE* vorhanden ist
  - *#22 = TOROIDAL_SURFACE(",#23,10.,2.);*
- PYRAMIDE: Sie besteht aus genau fünf Punkten = *VERTEX_POINT()*
  - *#266 = VERTEX_POINT(",#267);*
  - *#267 = CARTESIAN_POINT(",(0.E+000,10.,0.E+000));*

  Die Größen wie Länge, Breite und Höhe kann man anhand der *VERTEX_POINT* Werte berechnen.

4.5 Feinentwurf

4.5.1 Systemarchitektur

In folgender Abbildung (siehe Figure. 32) wird eine detalliertere Darstellung der Systemarchitektur im Vergleich zur Darstellung im Abschnitt “Grobentwurf ” vorgestellt.

In der Zeichnung werden mehrere Aspekte der Architektur dargestellt:
- die verschiedenen Software Java-Pakete *packages* auf Java Code Ebene
- die Klassen der jeweiligen Java-Pakete
- die verschiedenen Eingabe sowie Ausgabe Dateiformate
- eine grobe Darstellung für die Abbildung der Daten aus STEP Daten oder XML-Elementdaten in Klassenobjekte aus den Klassen *StepElement* und XMLElementNode
- Richtung der Umwandlung und Datenflusses

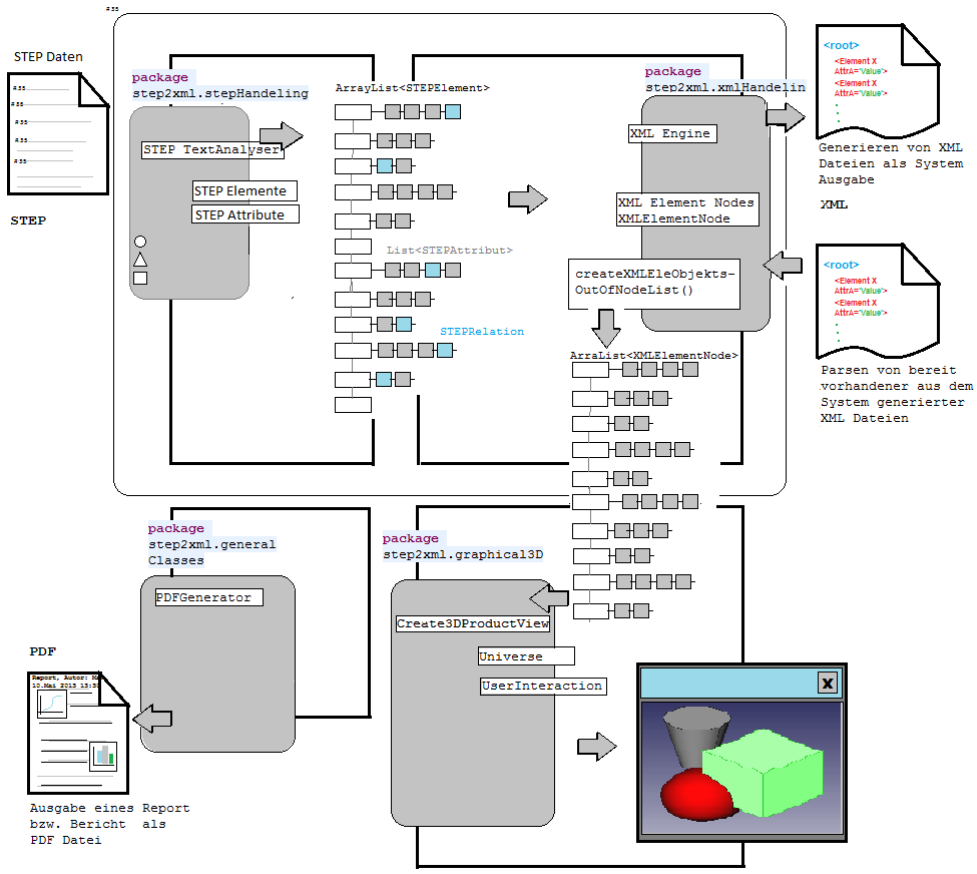

Figure 32: Detalliertere Darstellung der Systemarchitektur

4.5.2 Softwarepakete

- *Java default package*

  In der *default package* sind Klassen für die Erstellung der GUI enthalten. Die Software Arbeitsfläche hat viele wichtige Funktionalitäten wie:

  1. Navigation-Pfade (MenuBar),
  2. Dateienverwaltung wie das Öffnen von STEP- oder XML-Dateien,
  3. Alle Funktionen der Software wie Suchfunktion, XML-Daten graphisch modellieren oder STEP aus XML generieren sind über diese Arbeitsoberfläche ausführbar
  4. Help Bereich
  5. Die UI-Container für STEP- und XML-Inhalte

  Die Klassen des Softwarepakets sind:

  - MainWindowDesktop
  - TextLineNumber

- package *step2xml.stepHandeling*

  Dieses Paket umfasst alle Klassen zum Parsen von STEP Dateien und ihren Strukturen, sowie das Bearbeiten von STEP Dateninhalten.

  - STEPAttribute
  - STEPElement
  - STEPTextAnalyser

- package *step2xml.generalClasses*

  step2xml.generalClasses beinhaltet Klassen, Funktionen, Methoden und Attribute, die in allen Softwarepaketen des Projektes verwendet werden, wie zum Beispiel die Klasse *Utilities*. Die Klasse *Utilities* beinhaltet beispiels-weise die Funktion *read_STEP(File)* zum Öffnen und Lesen einer STEP-Datei. Diese Funktion wird von der Klasse *STEPTextAnalyser*, die in einem anderen Softwarepaket liegt, aufgerufen. Dieses Paket beinhaltet folgende Klassen:

  - PDFGenerator
  - PointOnAxis
  - Utilities
  - ColorValue

- package *step2xml.xmlHandeling*

  Ein weiteres wichtiges Softwarepaket im *STEP2XMLGenerator* Projekt ist dieses Package. Es beinhaltet folgende Klassen, die für das Parsen der XML Inhalte sowie die Erzeugung von XML Elementen bzw. Inhalten zuständig sind.

  - XMLElementNode
  - XMLEngine

- package *step2xml.graphical3D*

  Wie der Name des Softwarepaketes schon sagt, ist dieses Paket für alle Funktionalitäten, die für die graphische Modellierung und das Anzeigen der XML Daten in 3D benötigt werden, zuständig.

  - Universe
  - Create3DProductView
  - CutedCone
  - Pyramid
  - Torus
  - UserInteraction

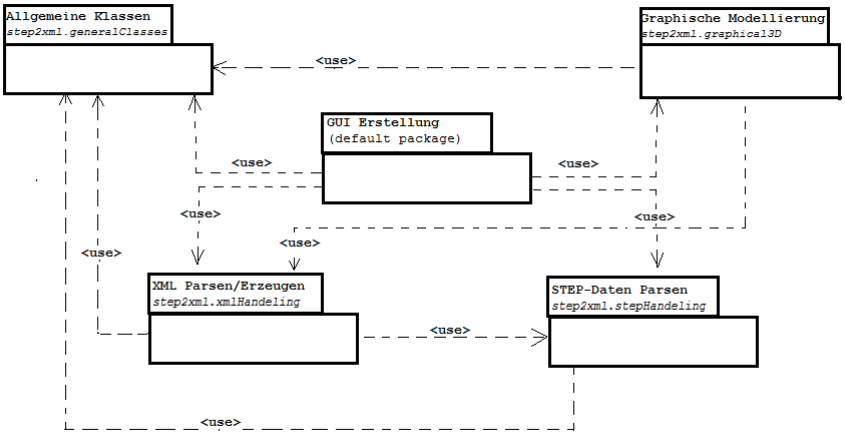

Figure 33: Softwarepakete bzw. *Package* Diagramm

4.5.3 Die Kernfunktionen

1. STEP Data Parser: zum Parsen der STEP Informationen aus der STEP-Datei in Objekte
2. XML-Parser: zum Parsen der XML-Elemente
3. 3D-Modellierung und Vorschau der generierten XML-Daten
4. PDF-Generator: Generieren und Exportieren eines Berichts *"Result Report"* als PDF
5. Suche: Suche und Hervorhebung *Highlight* der Suchbefunde

analyseTheSTEPFileLines(File aSTEPFile)

Diese Funktion ist für die Analyse der STEP-Datenzeilen zuständig. Die Daten werden zeilenweise geparst bzw. eingelesen und die relevanten STEP-Daten für die eigentliche Modellierung werden in XML-Elemente abgebildet.

```
/**
 * read and process every line of the STEP File Lines and
 * @param aSTEPFile
 */
public static void analyseTheSTEPFileLines(File aSTEPFile){
    setGlobAllLinesOfFile(Utiliteis.read_STEP(aSTEPFile));
    setGlobDataSectionLines(GetDATASecLines(getGlobAllLinesOfFile()));
    setGlobSTEPFile(aSTEPFile);

    ArrayList<String> tmpDataSecLines=getGlobDataSectionLines();
    for(int i=0;i<tmpDataSecLines.size();i++){
        parseLineNumOfSTEPLine(tmpDataSecLines.get(i));
        if(tmpDataSecLines.get(i).toLowerCase().contains("= product('box'")){
            stepEleBOX=new STEPElement(Utiliteis.BOX,Utiliteis.BOX);
            String searchedSTEPDataLine=getSTEPDataLine(i,"= AXIS2_PLACEMENT_3

            stepEleBOX.setCoordinationOfCartesianPnt("CARTESIAN_POINT",getCoor
            //handle the box....
            arrLstOfSTEPElement.add(stepEleBOX);
        }

        if(tmpDataSecLines.get(i).toLowerCase().contains("= product('sphere'"){
            //handle the Sphere....
```

Figure 34: Abschnitt vom Quellcode der Funktion *analyseTheSTEPFile-Lines(File aSTEPFile)*

generateXMLFileContentOutOfSTEPElements()

Diese Methode ist für die Bereitstellung einer neuen XML-Datei mit dem Namen *Originale STEP-Dateiname + "_generatedXML.xml"* vorgesehen.

Es wird dabei folgende Funktion für das Erzeugen XML-Node Daten verwendet:

createXMLFileWithXMLContent(String, ArrayList<XMLElementNode>) Die wichtige Aufgabe dieser Funktion ist das Erzeugen von XML-Inhalten anhand von STEP-Daten.

```
/**
 * generating an xml structure out of the array of STEPElement s
 * and saving an xml file with this structure as
 * nameOfSTEPFile + "_generatedXML.xml"
 */
public static void generateXMLFileContentOutOfSTEPElements(){
    ArrayList<STEPElement> arrLstOfSTEPEle=
            STEPTextAnalyser.getArrLstOfSTEPElement();
    //getting the ".step" out of the pathname and file name
    String nameOfXMLFile=
            STEPTextAnalyser.getGlobSTEPFile().getAbsolutePath().substring
            (0,STEPTextAnalyser.getGlobSTEPFile().getAbsolutePath().length()-5)
            +"_generatedXML.xml";
    //attaching to the path+filename "_generatedXML.xml"
    for(int i=0;i<arrLstOfSTEPEle.size();i++){
        getArrLstOfXMLEleNodeOutOfSTEPElements().add(createXMLElementNodes(
                arrLstOfSTEPEle.get(i)));

    }
    boolean isFileSuccCreated=createXMLFileWithXMLContent(nameOfXMLFile,
            getArrLstOfXMLEleNodeOutOfSTEPElements());

}
```

Figure 35: Quellcode der Funktion *generateXMLFileContentOutOfSTEPElements(ArrayList<STEPElement>)*

createSceneGraph(SimpleUniverse su)

Diese Methode ist zuständig für die 3D-graphische Modellierung der XML-Elemente, die die STEP-Daten beinhalten. Dabei wird die Objektliste, die alle generierten *XMLElementNode*-Objekte aus den STEP-Daten beinhalten, verwendet. Womit alle relevanten Daten aus der *XMLElementNode*-Objekte bei der graphischen Modellierung eingesetzt werden.

Für diese Objekte werden anhand der Namen, der Typen, der Koordinaten und der Eigenschaften eine geometrisches 3D-Teilprodukt modelliert und in die graphische Szene hinzugefügt.

```
//In this method, the objects for the scene are generated and added to
//the SimpleUniverse.
public void createSceneGraph(SimpleUniverse su){
  //at this point the xml file should be parsed to get the data for the 3D drawing
    //TODO: get the xml data
  ArrayList<XMLElementNode> xmlElementsList=new ArrayList<XMLElementNode>();
  xmlElementsList=XMLEngin.createXMLEleObjektsOutOfNodeList();
  ArrayList<Object> arrLstOf3DObjects=new ArrayList<Object>();
  Appearance generalApperance = new Appearance();

  for(int i=0;i<xmlElementsList.size();i++){
      //hier the 3d grafic is created according to the xmlElements data
      if(xmlElementsList.get(i).getType().toLowerCase().equals(Utiliteis.BOX)){
          TransformGroup tg = new TransformGroup();
          Transform3D transform = new Transform3D();
           //apperance und color definition
          xmlElementsList.get(i).getColor();
```

Figure 36: Ein Teil vom Quellcode der Funktion *createSceneGraph(SimpleUniverse su)*

generatePDF (filename, STEPDataStr, XMLDataStr, titel)

Eine Methode zur Erzeugung eines Berichts für die Ergebnisse des Umwandlungsprozesses. Dieser Bericht beinhaltet die Inhalte der STEP-Datei sowie Inhalte der daraus generierten XML-Datei mit Titel und Beschreibung.

Der Bericht wird als PDF-Datei gespeichert. Es werden folgende Parameter an die Methode übergeben:

1. *filename* : der vordefinierte Dateiname
2. *STEPDataStr :* die Modellierungsdaten der STEP-Datei
3. *XMLDataStr :* die resultierenden XML-Daten
4. *titel* : die Überschrift des Berichts als string

```
private static Font txtFont1=new Font(Font.FontFamily.TIMES_ROMAN,14);
private static Font txtSmallFont2=new Font(Font.FontFamily.TIMES_ROMAN,10,
        Font.BOLDITALIC);
private static Font titelFont2=new Font(Font.FontFamily.TIMES_ROMAN,17,Font.BOLD);
private static void generatePdf(String filename, String STEPDataStr,
        String XMLDataStr) throws DocumentException,
        IOException {
    //TODO: the strukture of the created pdf still strange and need to be styled
    Document document = new Document();
    PdfWriter writer = PdfWriter.getInstance(document,
            new FileOutputStream(filename));
    document.open();
    addSoftwareLogo(document);
    addMetaDataOfDocument(document);
    addDocumentTitelPage(document,"Document Titel...");
    addReportData(document,STEPDataStr, XMLDataStr);
    document.close();
}
```

Figure 37: Ein Ausschnitt vom Quellcode der Funktion *CreateAndSavePDFFile (File)*

searchAndHightlightText (JTextArea,String,String)

Die Aufgabe dieser Methode ist das Suchen im Text des UI-Bereichs *JTextArea* nach einer bestimmten übergebenen Zeichenfolge.

Die Methode markiert das Ergebnis der Suche farbig.

Die übergebenen Parameter, die diese Methode bekommt, sind:

1. *searchedJTextArea* : ist ein Objekt, das bezeichnet in welchem UI-Textbereich gesucht werden soll
2. *searchWord* : ein String, welcher die Zeichenfolge für den gesuchten Wert beinhaltet
3. *stepORxml_ TA* : anhand diesem Wert der Variablen wird bestimmt in welcher JTextArea gesucht werden soll, in dem *JTextArea* der STEP-Daten oder der XML-Daten

```
/**
 * search a String in the textArea text and highlight it
 * searchAndHighlightText()
 */
private void searchAndHighlightText(JTextArea searchedJTextArea,
                                    String searchWord,String stepORxml_TA){
    Highlighter.HighlightPainter myHighlighter =
        new DefaultHighlighter.DefaultHighlightPainter( Color.YELLOW );
    int indexOfSearchedWord = searchedJTextArea.getText().indexOf(searchWord);
    int lengthOfSearchedWord = searchWord.length();
    while ( indexOfSearchedWord != -1)
    {
        try
        {
            if(stepORxml_TA.equals(STEP_JTA)){
                stepFileTxtArea.getHighlighter().addHighlight(indexOfSearchedWord,
                        indexOfSearchedWord + lengthOfSearchedWord, myHighlighter);
                indexOfSearchedWord = searchedJTextArea.getText().indexOf(searchWord,
                        indexOfSearchedWord+1);
            }
            else if(stepORxml_TA.equals(XML_JTA)){
                xmlFileTxtArea.getHighlighter().addHighlight(indexOfSearchedWord,
                        indexOfSearchedWord + lengthOfSearchedWord, myHighlighter);
                indexOfSearchedWord = searchedJTextArea.getText().
                        indexOf(searchWord,indexOfSearchedWord+1);
```

Figure 38: Quellcode der Funktion *searchAndHightlightText (JTextArea)*

4.5.4 Klassendiagramm

In diesem Unterabschnitt geht es um die Figure 39 "Klassendiagramm". Es werden alle Klassen der Software vorgestellt. Es werden dabei mehrere Aspekte dargestellt:

- die Klassen mit der Bezeichnung und die wichtigen Attribute
- die Software-Pakete (*Packages*) sind dabei durch die grüne Markierung erkennbar. Jedes Paket im Programm wird einzeln mit der dazugehörigen Klasse mit grünem Quadrat markiert.
- die Zugriffs- und Benutzungsbeziehungen zwischen den Klassen und zwischen Software-Pakete sind mit Pfeile abgebildet

4.5.5 *MainWindowDesktop.java*

- Diese Klasse ist für die Erstellung des *Graphical User Interfaces* des *STEP2XML-Generator* Tools zuständig.

  Sie ist das Hauptfenster des Tools. Es beinhaltet folgende Funktionen:

  - *createAndShowGUI()* ist eine Initialisierungsfunktion, um das Hauptfenster abzuleiten und anzuzeigen
  - *createContentPane()* dient zur Erzeugung der Inhalte (*Content*) aus dem Haupt-Panel
  - *createXMLDataJPanel()* erstellt das Panel für der Textbereich *JTextArea* für die generierten XML Informationen

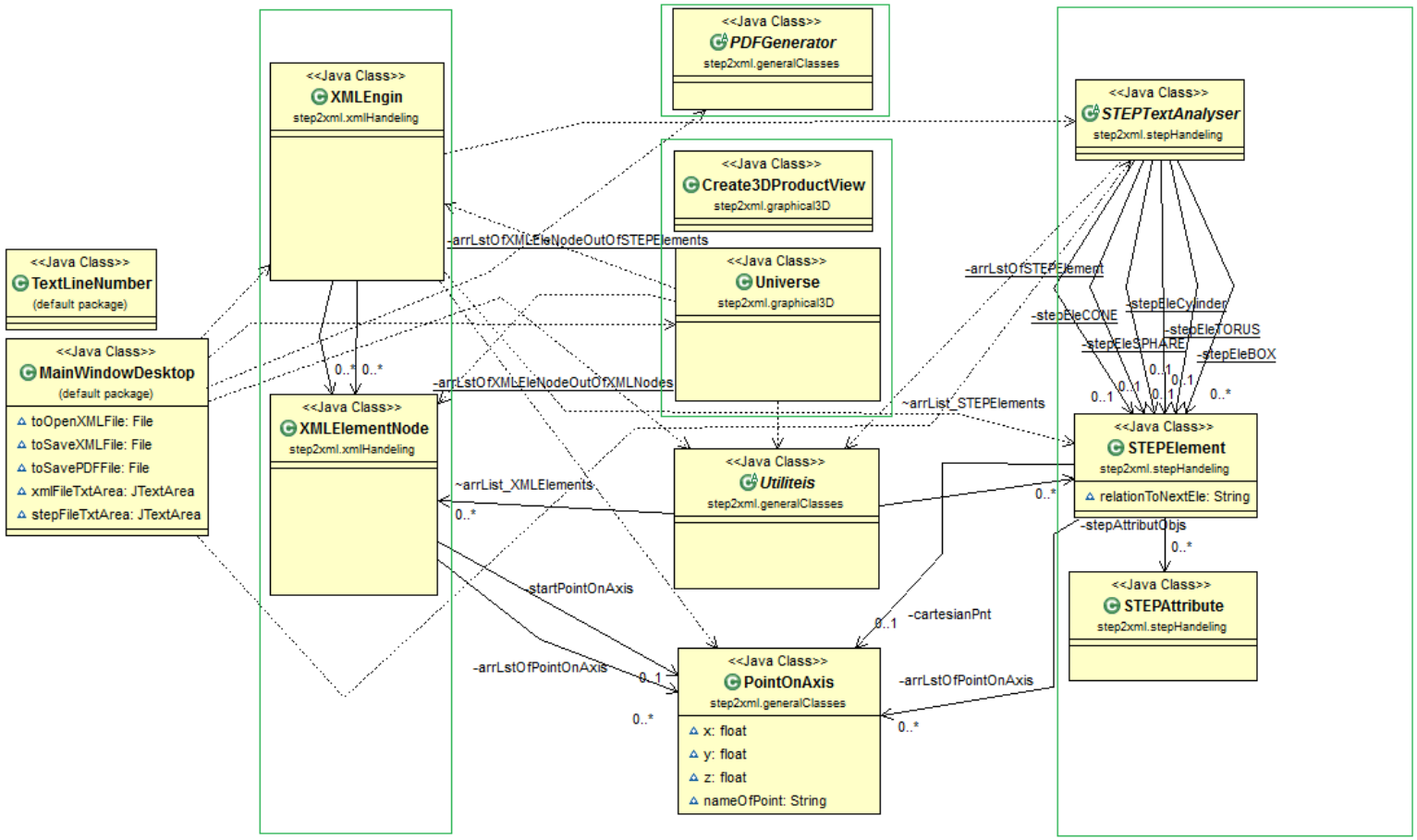

Figure 39: Klassendiagramm

- *createSTEPDataJPanel()* ist für die Erzeugung des Panels für die importierten STEP-Daten zuständig
- *createJScrollPane_withLineNumbers()* erzeugt ein *JScrollPane*, die dem Benutzer ermöglicht, beliebig große Daten anzuzeigen. Dabei kann der Benutzer mit Hilfe der *ScrollBar* navigieren. Eine zusätzliche Funktionalität dieser Funktion ist die Zeilennummerierung.

- Eine weitere Komponente dieser Klasse ist die MenuBar mit allen seinen JMenuItems. Damit werden die Funktionalitäten pro JMenuItem gesteuert und darauf zugegriffen :
  - *createMenuBar()* ist zuständig für die Erstellung und Initialisierung der MenuBar

- Die Klasse beinhaltet sowohl die Funktionen zur Erstellung der graphischen Benutzeroberfläche als auch weitere wichtige Funktionen, wie :
  - *CreateAndSavePDFFile(File)* zur Generierung eines Berichts als PDF-Datei, der STEP-Daten und einen generierten XML Baum beinhaltet
  - *searchAndHighlightText(JTextArea)* für die Textsuche und das Markieren (Highlight) der Ergebnisse innerhalb der *JTextArea*
  - *SaveXMLFileAs(File)* wie der Name der Funktion schon sagt, ist ihre Aufgabe das Speichern des generierten XML Baumes als XML Datei.
  - *OpenXMLFile_inTextField(File)* dient zum Öffnen von XML Dateien über den Dateien-Browser in der XML- JTextArea
  - *OpenSTEPFile_inTextField(File)* ist zum Öffnen und zum Anzeigen von STEP Dateien über den File-Browser im STEP Text-Bereich *JTextArea*

4.5.6 *PDFGenerator.java*

Zur Generierung des Berichts, der die Ergebnisse des Parsens von STEP-Daten in XML formatierte Daten beinhaltet, kommt dieses Modul zum Einsatz. In diesem Modul werden verschiedene *Font* Attribute (Schriftarten) zur Verwendung in der Erstellung des Dokumentes definiert.

Alle Funktionen, die zur Bearbeitung und Erzeugung von PDF-Dateien im ganzen System zuständig sind, werden in die Klasse *PDFGenerator.java* ausgelagert.

- *generatePdf(filename,STEPDataStr,XMLDataStr)* ist zuständig für die Generierung das PDF-Dokument. Mit dieser Funktion werden die Dateinamen *filename* und der Inhalten der STEP Datei *STEPDataStr* und der XML generierten Struktur *XMLDataStr* für den Bericht erzeugt. Außerdem werden die wichtigen *Document* und *PdfWriter* Variablen definiert, welche im ganzen Modul zum Einsatz kommen.

- *addMetaDataOfDocument(Document)* erstellt die Metadaten für das PDF-Dokument. Die Metadaten beinhalten unter anderem Autordaten, Datum, Uhrzeit und *Copy-Rights* bestimmungen.

- *addReportData(Document,sTEPDataStr,xMLDataStr,tietl)* fügt die die - STEP-Daten und XML strukturierten Daten dem Dokument hinzu.
- *addDocumentTitelPage(Document)* erstellt die Titelseite, welche den Titel, Autornamen und einen kurzen Abstract beinhaltet. Dabei werden verschiedene Schriftformate verwendet.
- *createPseudoLine(Paragraph)* kreiert in einem bestimmten vordefinierten Paragraph eine leere Zeile.
- *addSoftwareLogo(Document)* fügt das *STEP2XML* Tool Logo dem Dokument hinzu.
- *setImage(myDocu,imageParagraph,imgPath,scalePercent)* wird benutzt, um eine bestimmte graphische Abbildung bzw. Bild zu einem bestimmten Paragraph mit einer gewünschten Einstellung hinzuzufügen.
- *initializePDF(PDFFilePath,sTEPDataStr,xMLDataStr,titel)* ist die *Frontend* bzw. die Zugriffsschnittstelle im *PDFGenerator* Modull von außen. Durch den Zugriff von außen mit den drei übergebenen Parametern, wird die PDF-Dokument Generierung angestoßen und initialisiert. Deshalb ist die Methode mit der Sichtbarkeit *Public* definiert

4.5.7 *STEPElement.java*

STEPElement.java wird zur Abbildung der STEP Objekte aus der STEP Datei benutzt. Jedes Objekt dieser Klasse repräsentiert ein STEP Element mit all seinen Attributen und weiteren Daten.

4.5.8 *STEPAttribute.java*

Diese Klasse wird benutzt, um alle Attribute, Eigenschaften und Beschreibungsdaten der STEP graphischen Objekte und Relationen abzubilden. Als Beispiel solcher Attribute in den STEP Daten sind die Koordinaten eines bestimmten STEP Elements auf den XYZ-Achsen oder die Farbe eines graphischen STEP Objektes.

Es werden dabei Typ *attType* , Name *attName* und Wert *attValue* des Attributes sowie eine *attDiscription*Variable verwendet, um weitere Daten bzw. Beschreibungen zu speichern. Die verschiedenen Arten der Attribut-Typen sind vordefiniert in der Klasse *Utilities.java*.

4.5.9 *STEPTextAnalyser.java*

STEPTextAnalyser.java ist eine Art Parser für die STEP Daten. Er analysiert die Daten in der STEP Datei auf textueller Ebene und holt alle wichtigen Informationen und Daten für die Erstellung von *STEPElement* und *STEPAttribute* Objekte, die später bei der Erstellung der XML strukturierten Daten verwendet werden.

Diese Klasse hat die Funktion eines Parser, es beinhaltet alle relevanten Methoden für die Analyse des STEP Textes. Eine Beispiel Funktion dieser Methoden ist das Suchen von STEP graphischen Elementen und Attributen bzw. Eigenschaften.

- *isWordInLine(lineOfSTEPFile,targetWord)* überprüft ob ein bestimmtes Wort im Satz vorkommt.
- *getIndexesOFWordInString(strLine,searchedWord)* sucht an welcher Position in einen bestimmten Satz ein Wort sich befindet und gibt den Wert des Indexes zurück.
- *isTargetSTEPElement(String)* dient zum Testen ob eine bestimmte Zeichenfolge ein STEP Element ist
- *isTargetSTEPAttribute(String)* ist zum Testen ob eine bestimmte Zeichenfolge ein STEP Attribut ist
- *generateSTEPEleObject(String)* Diese Funktion erzeugt ein *STEPElement* Objekt
- *analyseTheSTEPFileLines(File)* ist die Hauptmethode, die den Prozess des Parsens der STEP Daten startet
- *getCoordination(String)* liefert aus der STEP Datenzeile die XYZ Koordinaten als *float* Zahlen zurück
- *getSTEPDataLine(int,String)* ist eine Funktion, die eine bestimmte STEP Zeile, die die Werte übergebener Strings und Integer beinhaltet, sucht.
- *getSTEPLineAtPosition(int,String[])* gibt die STEP Zeile mit einer gesuchten Zeilennummer zurück
- *GetDATASecLines(String[])* liefert die *DATA* Abschnitt-zeilen aus der STEP Datei in Form eines Arrays
- *GetHEADERSecLines(ArrayList<String>)* liefert die *HEADER* Abschnittzeilen aus der STEP Datei in Form einer Liste
- *isWordInLine(String,String)* überprüft, ob eine bestimmte Zeichenfolge in einer Zeile enthalten ist
- *parseLineNumOfSTEPLine(String)* parst alle Referenz-Zeilennummern - (#3,#45 ...), die eine Adressierung zu anderen STEP Zeilen darstellen, aus einer STEP Zeile
- *getIndexesOFWordInString(String,String)* sucht Indizes eines Wortes in einer Zeichenfolge

4.5.10 *Utilities.java*

Umfasst alle Methoden, Funktionen und Attribute, die aus allen anderen Modulen bzw. Klassen des Systems zugreifbar sind.

Es handelt sich beispielweise um Attribute und Methoden, die global wie *public* definiert werden und an verschiedenen Stellen im Programmcode gleich verwendbar sind.

4.5.11 *XMLElementNode.java*

Mit dieser Klasse werden alle Daten der graphischen Objekte aus der STEP Beschreibung definiert und gespeichert.

Diese Daten beinhalten den Namen bzw. die Bezeichnung des STEP-Elementes und den Typ des STEP-Elementes.

Des Weiteren enthält sie wichtige Attribute wie Länge, Breite, ... und Radius. Der Radius ist dann von Bedeutung, wenn es sich um ein kreisförmiges, graphisches Teil handelt.

Ausserdem benötigt man die Koordinatendaten für jedes graphische Objekt:

coordi*nateX*, *coordinateY*, *coordinateZ* werden in Form eines *PointOnAxis* Objekts gespeichert.

Für alle diese Daten sind Variablen innerhalb der Klasse *XMLElementNode* definiert.

In der XML hat das XML-Element "step_objekt" ein weiteres Attribut "type", um den graphischen Typ des STEP Objekts zu speichern.

Werte für das Attribut Type können *box*, *sphere*, *cylinder*, *cone* etc. sein. (siehe Figure 40)

```
</step_object>
<step_object name="box1" type="box" attribute1="data2" length="43"
    width="23" higth="76" mesurment1="03" color="red"  attribute="bla">
    This is STEP a XML-Object!
    <coordinats name="x"> 20,30 </coordinats>
    <coordinats name="y"> 34,54 </coordinats>
    <coordinats name="z"> 82,10 </coordinats>
    <relation2next relation2next="side_cut" relation2object="stepobject3">
    </relation2next>
</step_object>
<step_object name="sphere_objekt" type="sphere" attribute1="data_stepobject3" length="43"
    width="87" higth="24" mesurment1="000" color="green" radius="15" attribute="bla2">
    This is STEP a XML-Object!
    <coordinats name="x"> 20,30 </coordinats>
    <coordinats name="y"> 24,64 </coordinats>
    <coordinats name="z"> 90,00 </coordinats>
</step_object>
<relations>
```

Figure 40: *XMLElement*-Daten in einen XML Node

In der Figure 41 ist dargestellt, wie die Klasse *XMLElementNode* strukturiert ist und mit welchen Variablen sie ausgestattet ist. Die Werte jeder XML-Node aus der generierten XML-Datei sind eine Darstellung der Daten aus den Variablen der erzeugten *XMLElementNode* Objekte. (siehe Figure 41)

```java
package step2xml.xmlHandeling;
import java.util.ArrayList;
import step2xml.generalClasses.PointOnAxis;
public class XMLElementNode {
    private String eleName="";
    private PointOnAxis startPointOnAxis=new PointOnAxis();
    private String color="";
    private float hight=0;
    private float width=0;
    private float depth=0;
    //arrLstOfPointOnAxis contains all possible point on the axis for
    //this XMLElementNode object (graphical shape)
    private ArrayList<PointOnAxis> arrLstOfPointOnAxis=new ArrayList<PointOnAxis>();
    /**
     * creates an add new pointOnAxis object into the arrayList
     * @param xPoint
     * @param yPoint
     * @param zPoint
     */
    public PointOnAxis createAndAddNewPointObjInArrLst(String xPoint,String yPoint,String zPoint){
        PointOnAxis newPointOnAxis=new PointOnAxis(Float.parseFloat(xPoint),
                Float.parseFloat(yPoint),Float.parseFloat(zPoint));
        arrLstOfPointOnAxis.add(newPointOnAxis);

        return newPointOnAxis;
    }
    //if it is a sphere or zylinder. ..
    private float radius=0;
```

Figure 41: Die Klasse *XMLElementNode*

4.5.12 *XMLEngine.java*

XMLEngine hat eine zentrale Funktion beim Parsen sowie bei der Erzeugung der XML-Daten. Einige der vielen und wichtigen Methoden und Funktionen dieser Klasse sind im Folgenden aufgelistet:

- *createXMLFileWithXMLContent(String,ArrayList<XMLElementNode>)* - Zur Erzeugung der XML-Datei mit den XML-Daten, die aus den STEP-Element Objekten abstrahiert wurden.
- *generateXMLFileContentOutOfSTEPElements()* erstellt die XML-Elemente, Attribute und Inhalte in Form eines strukturierten XML-Baumes, der später direkt in die XML-Datei geschrieben wird. Die XML-Elemente und Attribute sowie Inhalte werden anhand den Daten aus den *XMLElementNode* Objekten erstellt.
- *createXMLElementNodes(STEPElement)* Die Aufgabe dieser Methode ist die Erzeugung der einzelnen *XMLElementNode*-Objekte mit den Inhalten aus einem *STEPElement*-Objekt
- *createXMLEleObjektsOutOfNodeList()* Es parst die XML-Daten aus der eingegebenen XML-Datei und erzeugt daraus *XMLElementNode*- Objekte.

4.6 XML-Struktur

In diesem Kapitel werden die Themen, die im Zusammenhang mit der XML-Struktur stehen, behandelt. Wichtige Aufgaben und Aspekte sind im Folgenden aufgelistet:

- die Darstellung der STEP geometrische Objektdaten in XML-Elemente,
- Validierung der generierten XML Dateien mit dem XML-Schema
- der allgemein strukturelle Aufbau solcher XML-Bäume.

4.6.1 STEP Objekte als XML-Elemente <*step_objekt*>

<*step_objekt*> ist die XML-Repräsentation der geometrischen Modellierungsdaten einer STEP-Teilproduktszeichnung.

Ein XML-Element, welche alle relevanten geometrischen Daten eines STEP-Teilprodukts beinhaltet.

Der Name und der Typ eines STEP-Elements werden im Rahmen <*step_objekt*> gespeichert, wie im nachfolgenden Beispiel gezeigt wird

- <*step_objekt name=*"NAME OF OBJECT" ... und *type=*"BOX" ...>

Alle wichtigen Attribute und Eigenschaften eines STEP-Elements werden in Form von XML-Attributen in <*step_objekt*> untergebracht.

Weitere STEP-Daten wie die Koordinaten werden als XML-Kinderelemente <*coordinats*> und <*coordinat_group*> abgebildet.

4.6.2 STEP-Eigenschaftsdaten als XML Attribute

Unter STEP-Eigenschaftsdaten sind die Attribute und Eigenschaften eines STEP-Teilprodukts zu verstehen. Die wichtigen Attribute eines STEP-Teilprodukts lassen sich mit Farbe und Messdaten wie Länge etc., zusammenfassen.

- <*step_objekt name=*"NAME OF OBJECT" und *type=*"BOX" *length=*"23" *width=*"45" *hiegth=*"56" *color=*"red" ...>

4.6.3 Koordinatendaten als <*coordinats*> & <*coordinate_group*>

Die Koordinatendaten jedes STEP-Elements werden durch ein speziell vordefiniertes XML-Element abgebildet und zwar durch die <*coordinate_group*>.

<*coordinate_group*> ist ein XML-Vaterelement, den man als eine Art Container XML-Element bezeichnen kann, welches mehrere Koordinaten als XML-Kind-erelemente <*coordinats*> beinhaltet. Die XML-Kinderelemente repräsentieren die einzelnen X-, Y- oder Z-Koordinaten

<*coordinate_group name*"*name of coordinateGroup*">
<*coordinats name=*"*x*"> 20,50 </*coordinats*>
<*coordinats name=*"*y*"> 13,40 </*coordinats*>
<*coordinats name=*"*z*"> 35,50 </*coordinats*>
</*coordinate_group*>

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xsi:schemaLocation="http://www.w3.org/2001/XMLSchema STEP_XMLSchema.xsd ">
    <step_object name="object name" type="box" attribute1="data1" length="23"
        width="45" higth="56" mesurment1="21" color="black" >
        This is STEP a XML-Object!
        <coordinate_group name="start point">
            <coordinats name="x"> 20,50 </coordinats>
            <coordinats name="y"> 13,24 </coordinats>
            <coordinats name="z"> 120,30 </coordinats>
        </coordinate_group>
        <relation2next relation2next="top_merge" relation2object="object2">
        </relation2next>
    </step_object>
    <step_object name="box1" type="box" attribute1="data2" length="43"
        width="23" higth="76" mesurment1="03" color="red"  attribute="bla">
        This is STEP a XML-Object!
        <coordinate_group name="start point">
            <coordinats name="x"> 20,30 </coordinats>
            <coordinats name="y"> 34,54 </coordinats>
            <coordinats name="z"> 82,10 </coordinats>
        </coordinate_group>
        <relation2next relation2next="side_cut" relation2object="stepobject3">
        </relation2next>
    </step_object>
    <relations>
        <relation name="top_merge" art="mergw" >
            This is the relation named "top_merge" data
            <coordinate_group name="start point">
                <coordinats name="x"> 20,50 </coordinats>
                <coordinats name="y"> 13,24 </coordinats>
                <coordinats name="z"> 120,30 </coordinats>
            </coordinate_group>
        </relation>
        <relation name="side_cut" art="cut" >
            This is the relation named "side_cut" information
            <coordinate_group name="start point">
                <coordinats name="x"> 20.0 </coordinats>
<
```

Figure 42: Beispiel für die STEP-Beschreibungsdaten als XML Baum

In der Figure 42 ist eine XML Ausgabe dargestellt, die STEP graphische Produktsmodellierungsdaten beinhaltet und die graphischen Objekte beschreibt. Es fasst alle relevanten Informationen für die Modellierung, wie bestimmte Eigenschaften und welche XYZ-Koordinaten der Objekte, zusammen.

4.7 Benutzerschnittstellen *GUI*

In diesem Unterkapitel werden alle graphischen Aspekte des Programms vorgestellt. Insbesondere wird auf die graphische Benutzungsoberfläche *Grafical User-Interface* eingegangen.

Der Abschnitt besteht aus drei Unterabschnitten:

- die graphische Benutzeroberfläche, die alle Programmfenster und UI-Schnittstellen mit den wichtigsten Funktionen der Programme beinhaltet
- *MenuBar* mit seinen Navigations- und Bedienelementen
- eine Art Handbuch mit Benutzungsszenarien, die als Leitpfaden für den Nutzer dienen sollen

4.7.1 Graphische Benutzeroberfläche

Hauptfenster der Software

Dieses Fenster (siehe Figure 43) erscheint beim Öffnen des Programms. Es ist in zwei Textbereiche für beide STEP- und XML-Inhalte geordnet. Das Programmfenster ist auf der rechten oberen Seite mit drei Schaltelementen, die

für das Schließen, Vergrößern und Verkleinern des Programmfensters zuständig sind, ausgestattet.

Es beinhaltet an der oberen linken Seite eine MenuBar mit drei Menus *File*, *ToolBox* und *Help* für die Navigation. Damit können einzelne Funktionen- wie Datei Öffnen, Such-Funktion, STEP-XML Generierung etc. gestartet werden. Außerdem kann der Benutzer über die ToolBar auf der unteren linken Seite des Fensters auf alle Funktionen des Tools zugreifen und sie ausführen.

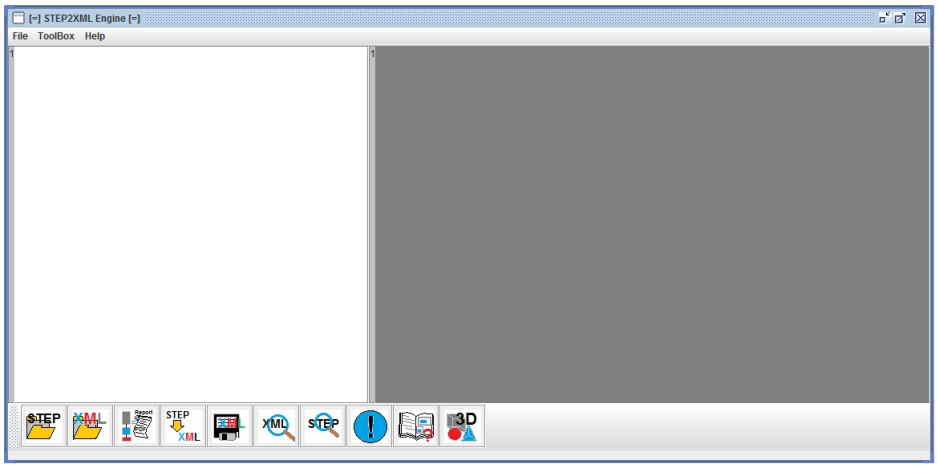

Figure 43: Hauptfenster des Programms

STEP Datei Öffnen

Man kann mithilfe des Programms eine STEP-Datei öffnen und seine Inhalte im linken Textbereich anzeigen lassen. Entweder über die MenuBar Menu *File-Open STEP File* oder über die ToolBar es ist möglich eine STEP-Datei zu öffnen.

Nach dem in der MenuBar das Bedienelement *Open STEP File* gewählt wird oder der Button für das Öffnen einer STEP-Datei mit Hilfe des Mauszeigers angeklickt wird, öffnet sich ein Dialog-Fenster mit der Aufforderung eine STEP-Datei auszusuchen. Daraufhin werden die Inhalte der STEP-Datei im linken Textbereich angezeigt.

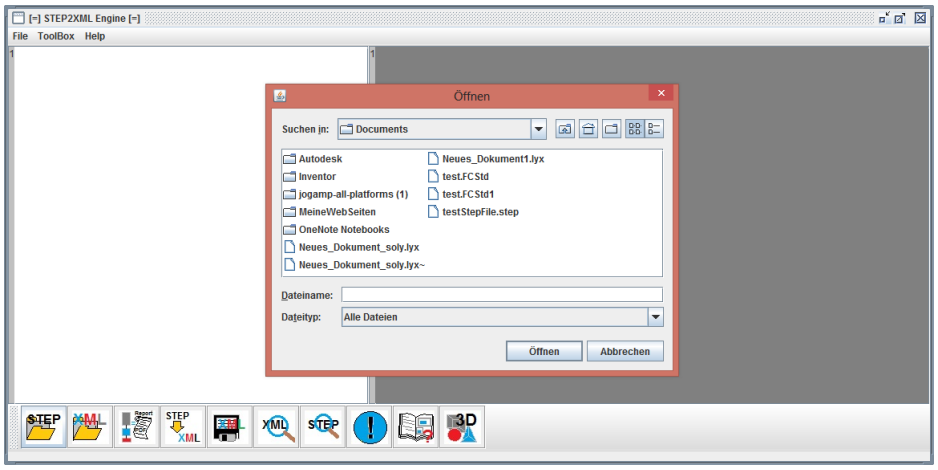

Figure 44: Datei-Browser Fenster zum Öffnen der STEP-Dateien

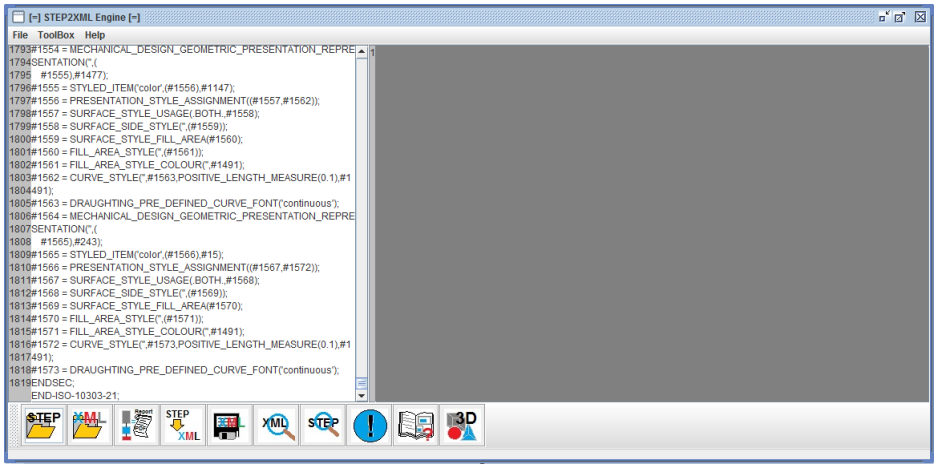

Figure 45: Inhalte einer STEP Datei

XML Datei Öffnen

Man kann mithilfe des Programms eine XML-Dateien öffnen. Sie wird entweder über die MenuBar File das Bedienelement *Open XML File* ausgewählt oder der Button für Öffnen einer STEP-Datei *Open STEP File* wird mit Hilfe des Mauszeigers angeklickt. Es öffnet sich ein Dialogfenster zum Öffnen von XML Dateien. Nachdem man eine Datei ausgewählt und die Schalttfläche “Öffnen” angeklickt hat, werden die XML Dateiinhalte auf der rechten Seite im Text-Bereich angezeigt.

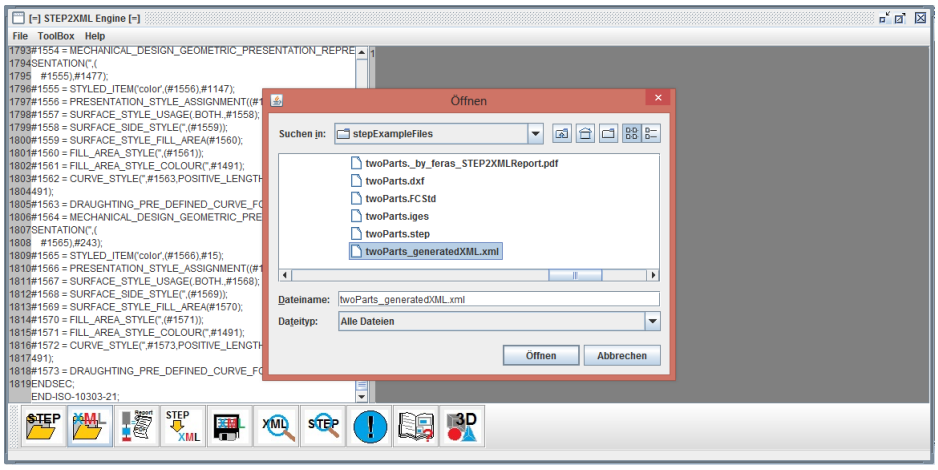

Figure 46: Datei-Browser Fenster zum Öffnen der generierte XML-Datei

Nach dem der Benutzer die XML Datei im Datei-Browser selektiert hat und auf "Öffnen" klickt, werden die Inhalte der Datei im Textbereich auf der rechten Seite im Textbereich angezeigt.

Auf folgender Figure werden die Inhalte der geparsten generierten XML Datei veranschaulicht.

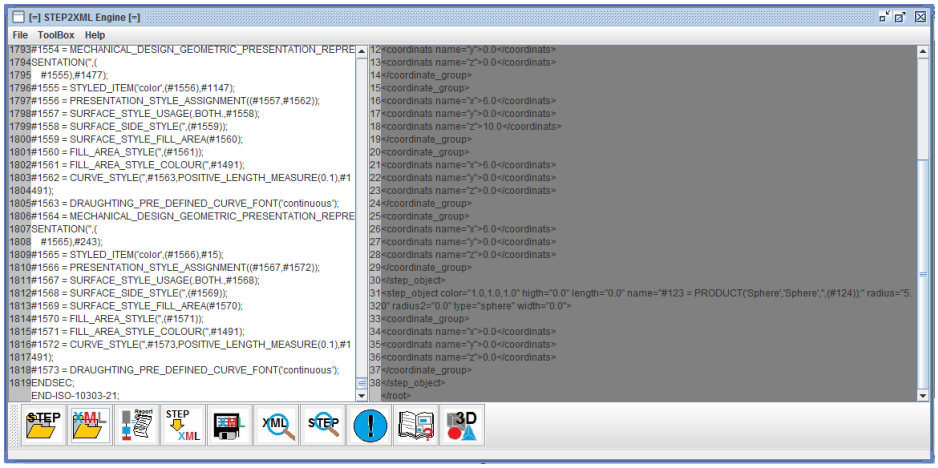

Figure 47: Angezeigte XML-Datei Inhalte

3D Modellierung der XML-Daten

Folgende Figure stellt uns das Fenster mit der 3D-Modellierung der gene-rierten XML-Daten dar. Die Funktion zur Generierung der 3D-Modellierung *"Show DATA in 3D Graphic"* wird aufgerufen in dem man entweder das MenuBar-Bedienelement *ToolBox - Generate DATA in 3D Graphic* anklickt oder der Benutzer benutzt den Button für 3D-Modellierung in der ToolBar. Beim Generierungsprozess werden die relevanten Daten aus dem Resultat der XML-Datei

abgelesen und in der 3D-Modellierung verwendet. Als Ergebnis erscheint ein neues Fenster mit dem graphischen Modell der generierten XML Daten.

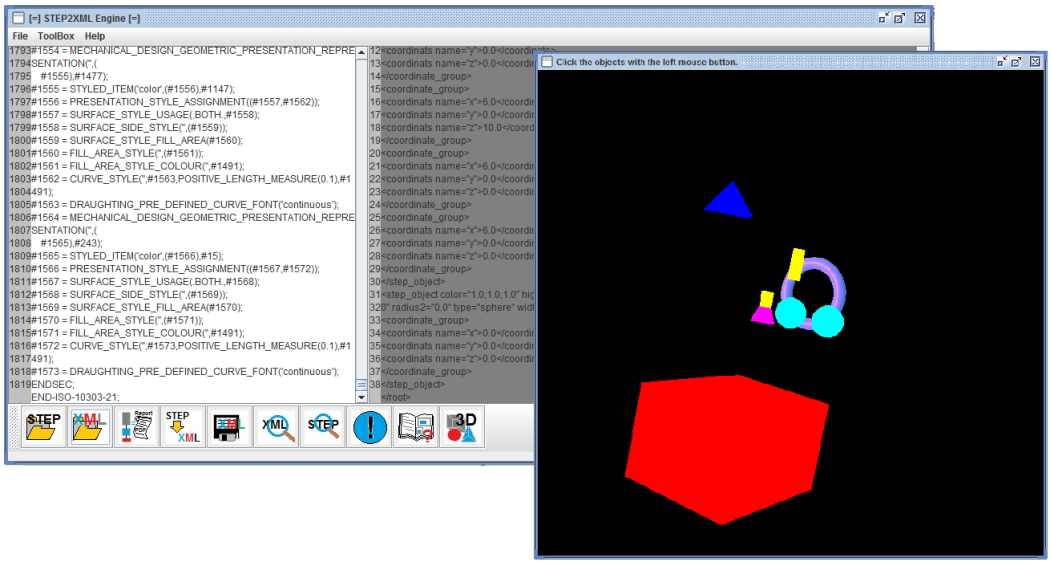

Figure 48: 3D-Modellierung aus generierten XML-Daten

4.7.2 Bedienelemente (*MenuBar*)

Die MenuBar befindet sich auf der linken Seite des Hauptfensters des Programms. Es beinhaltet drei Menüs *File*, *ToolBox* und *Help*.

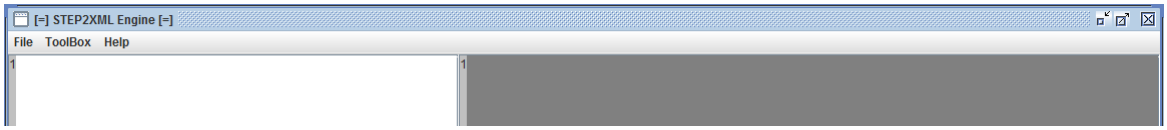

Figure 49: MenuBar des Programm

Die Bedienelemente der MenuBar werden als Navigationspfade betrachtet und sind wie folgt aufgelistet

File

- *File- Open STEP File* : zum Öffnen der STEP-Datei im STEP *TextArea* Textbereich
- *File- Open XML File* : zum Öffnen der XML-Datei im XML *TextArea* Textbereich
- *File- Save XML File As...* : für das Speichern der XML-Daten in eine ausgesuchte XML Datei

- *File- Export Report As Pdf* : zum Generieren eines Berichts als eine PDF Datei, der die STEP-Daten sowie die XML-Daten und weitere Metadaten beinhaltet,

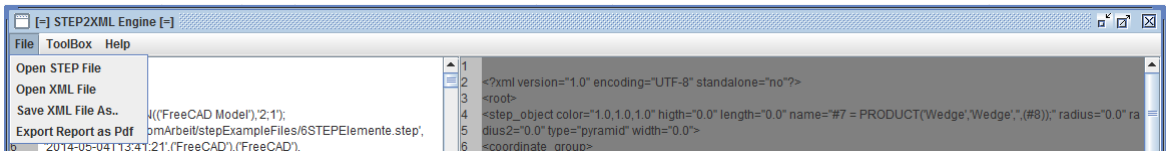

Figure 50: Menu "File" der MenuBars

ToolBox

- *ToolBox- Generate XML out of STEP* : startet die Hauptfunktion dieses Tools, welche für die Generierung von STEP-Daten in XML-Daten zuständig ist
- ToolBox- Search : öffnet ein Untermenü mit zwei verschiedenen Suchfunktionen

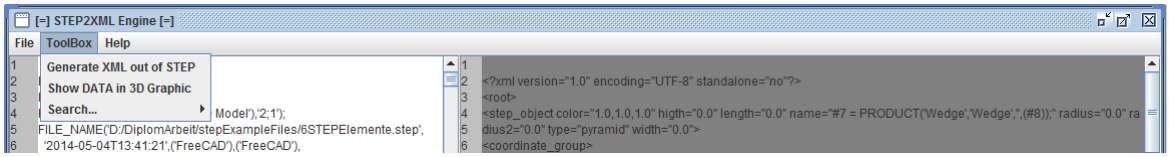

Figure 51: Menu "ToolBox" der MenuBars

- *ToolBox- Search- Search STEP* : öffnet ein Dialog mit einem Eingabefeld, damit der Benutzer eine bestimmte Zeichenfolge in dem STEP-Daten suchen kann
- *ToolBox- Search- Search XML* : öffnet ein Dialog mit einen Eingabefeld, damit der Benutzer eine bestimmte Zeichenfolge in den XML-Daten suchen kann

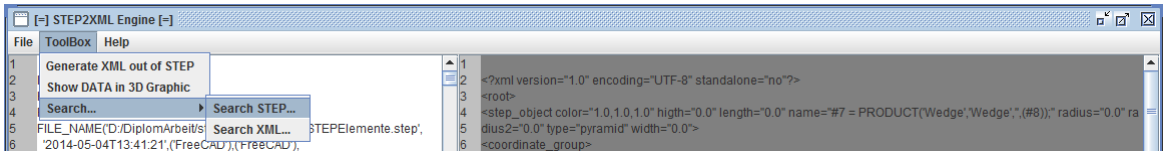

Figure 52: MenuItem "Search..." der Menu "ToolBox"

Help

- *Help- Handbook* : zum Anzeigen des Fenster mit dem Handbuch
- *Help- About* : öffnet ein kleines Dialogfenster mit allgemeinen Informationen über die Software

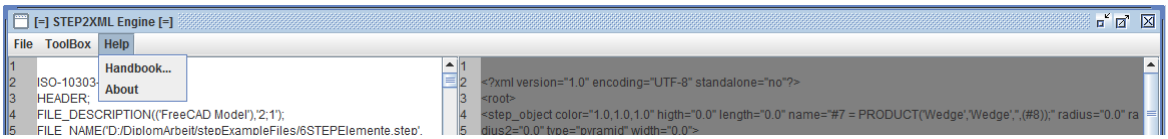

Figure 53: Menu "Help" der MenuBars

4.7.3 Bedienelemente (*ToolBar*)

Die *ToolBar* ist auf der unteren Seite des Hauptfensters platziert und beinhaltet für jede Funktion des Tools einen Button. Es ist möglich mithilfe der Maus die *ToolBar* beliebig auf dem Desktop als separates Fenster zu platzieren sowie rückgängig mithilfe der Maus per *Drag and Drop* in seine ursprüngliche Position im Fenster zu platzieren. Im Folgendem sind die Bedienelemente wie in Figure 54 nummeriert und werden aufgelistet und der Reihe nach erläutert.

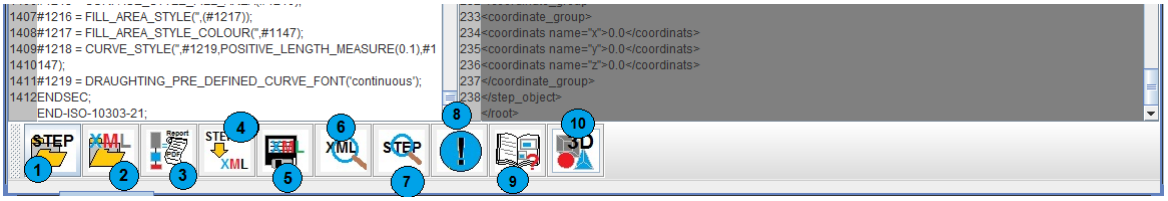

Figure 54: ToolBar Bedienelemente

1. zum Öffnen einer STEP-Datei
2. zum Öffnen einer XML-Datei
3. zum Generieren eines Bericht als PDF-Datei
4. zum Generieren der XML-Daten anhand der STEP-Daten
5. zum Suchen einer Zeichenfolge in XML Textbereich
6. zum Suchen einer Zeichenfolge in STEP Textbereich
7. zum Anzeigen das Info-Fenster, das Informationen über der Software beinhaltet
8. zum Öffnen des Hilfe-Bereichs bzw. das Handbuch
9. für die 3D-Modellierung der XML-Daten

4.7.4 Benutzerhandbuch

In diesen Kapitel werden die wichtigen Funktionen und die Anwendbarkeit des Programms erläutert. Anhand Benutzungsszenarien werden die einzelnne Anwendbarkeiten wie "*XML aus STEP generieren*" oder die 3D-Modellierung erläutert. Es werden verschiedene Szenarien für die Benutzung des Programms Schritt für Schritt erklärt:

Prozess 1- STEP-Datei öffnen

Prozess 2- XML-Datei öffnen

Prozess 3- XML aus STEP Datei generieren
Prozess 4- 3D-Modellierung und Ansicht der generierten XML Daten
Prozess 5- Text bzw. String in den STEP sowie XML Daten suchen
Prozess 6- Ein Status-Bericht als PDF-Datei exportieren

Prozess 1: STEP-Datei öffnen

Der Anwender will eine STEP-Datei öffnen. Dafür sollen folgende Schritte vollzogen werden:

- zuerst soll das Programm gestartet werden
- der Benutzer geht mit dem Mauszeiger auf die MenuBar und klickt auf den Eintrag *"File"*
- es erscheint ein Menü mit mehreren Funktionen zur Auswahl
- der Benutzer soll mit dem Mauszeiger auf den Menüeintrag *"Open STEP File"* fahren
- daraufhin öffnet sich ein Datei-Browser Fenster
- der Benutzer wird aufgefordert eine STEP-Datei zu öffnen
- als Endergebnis wird die Datei geöffnet und Inhalte werden im Textbereich auf der linken Seite angezeigt

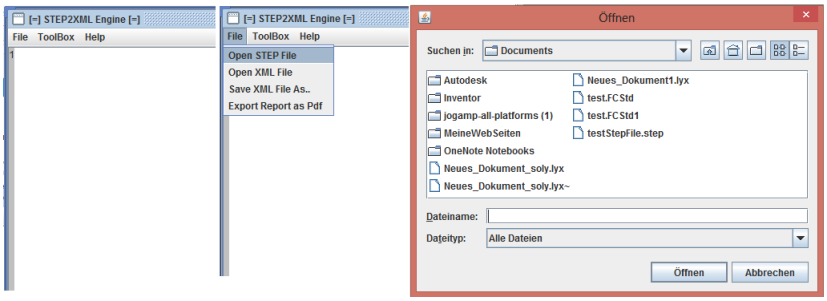

Figure 55: STEP-Datei öffnen

Prozess 2: XML-Datei öffnen

Der Anwender will eine XML-Datei öffnen. Dafür sollen folgende Schritte vollzogen werden:

- zuerst soll das Programm gestartet werden
- der Benutzer geht mit dem Mauszeiger auf die MenuBar und klickt auf den Eintrag *"File"*
- es erscheint ein Menü mit mehreren Funktionen zur Auswahl
- der Benutzer soll mit dem Mauszeiger auf den Menueintrag *"Open XML File"*

- daraufhin öffnet sich ein Datei-Browser Fenster
- der Benutzer wird aufgefordert, eine XML-Datei zu öffnen
- als Endergebnis wird die Datei geöffnet und die Inhalte werden im Textbereich auf der rechten Seite angezeigt

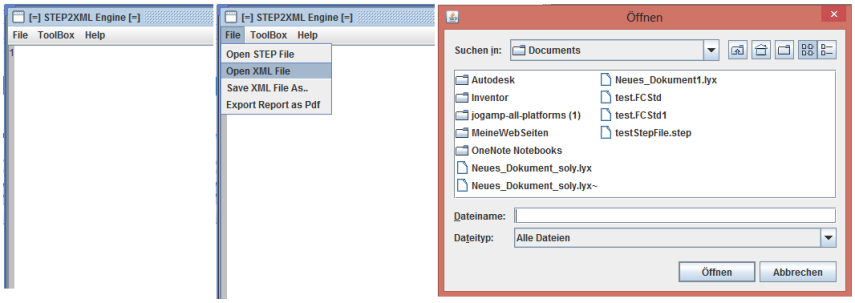

Figure 56: XML-Datei öffnen

Prozess 3: XML aus STEP Datei generieren

Um aus STEP Daten XML-strukturierte Daten zu generieren, müssen folgende Schritten durchgeführt werden:

- zuerst wird im Hauptfenster eine bestimmte STEP-Datei über das Dateibrowser-Dialogfenster geöffnet
- die Inhalte der STEP-Datei werden zeilenweise in den STEP Textbereich hinzugefügt
- über die MenuBar "ToolBox" und nach dem Öffnen einer STEP-Datei, soll die Funktion *"XML generieren aus STEP"* mit dem Mauszeiger angeklickt werden
- dadurch wird die Generierungsfunktion ausgeführt
- am Ende wird eine XML-Datei mit den Produkt Modell Daten erzeugt
- der Name der Datei besteht aus dem Namen der ursprünglichen STEP-Datei und "*..._generatedXML*"
- die Datei wird im gleichen Ordner, aus dem die STEP Datei gelesen wird, abgelegt

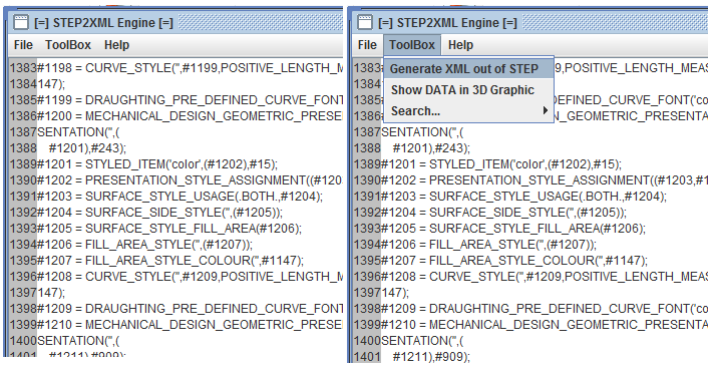

Figure 57: XML aus STEP Datei generieren

Prozess 4: 3D-Modellierung und Ansicht der generierten XML Daten

Der Benutzer hat die Möglichkeit die generierten XML-Daten wieder als graphisches Modell des Produktes zu modellieren

1. Fall 1: Falls die XML-Datei nach dem STEP2XML-Generierungsprozess nicht automatisch im rechten Text-Bereich angezeigt wird, soll zuerst die richtige XML-Datei geöffnet werden

   - als erstes wird über die MenuBar "*File*" der Eintrag "*Open XML File*" ausgesucht und mit der Maus angeklickt
   - danach öffnet sich ein Datei Browser Fenster
   - aus diesem Datei Browser soll die gewünchte XML-Datei ausgesucht und anschließend mit der Maus auf die Schaltfläche "*Open*" angeklickt werden
   - die XML-Dateiinhalte werden im rechten Textbereich angezeigt

2. Fall 2: Falls die XML-Dateiinhalte bereits nach der Generierung im rechten Textbreich geöffnet sind

   - über die MenuBar "*ToolBox*" soll der Eintrag "*Show DATA in 3D Graphics*" ausgewählt werden
   - nach ein paar Sekunden erscheint anhand der XML-Daten ein Fenster mit dem graphischen Modell

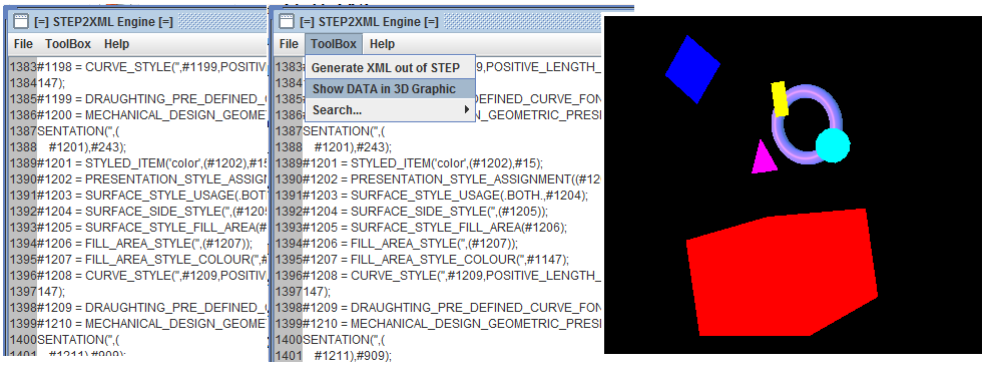

Figure 58: 3D-Modellierung und Ansicht der generierten XML Daten

Prozess 5: Text bzw. String in STEP-Daten sowie XML Daten suchen

Es besteht die Möglichkeit, bestimmte Textabschnitte innerhalb den STEP- sowie XML-Dateiinhalten zu suchen

- als erstes sollen die Inhalte der Dateien im Textbereich angezeigt werden
- über die MenuBar "ToolBox" soll der Eintrag "Search..." angeklickt werden
- daraufhin öffnet sich eine Unter MenuBar mit den Einträgen "Search XML..." und "Search STEP..."
- nach dem der Benutzer einen dieser Einträge auswählt, erscheint ein Dialog-Fenster mit der Aufforderung, das gesuchte Wort bzw. Zeichenfolge einzugeben und zu bestätigen
- die Ergebnisse der Suche werden im Textbereich direkt farbig markiert

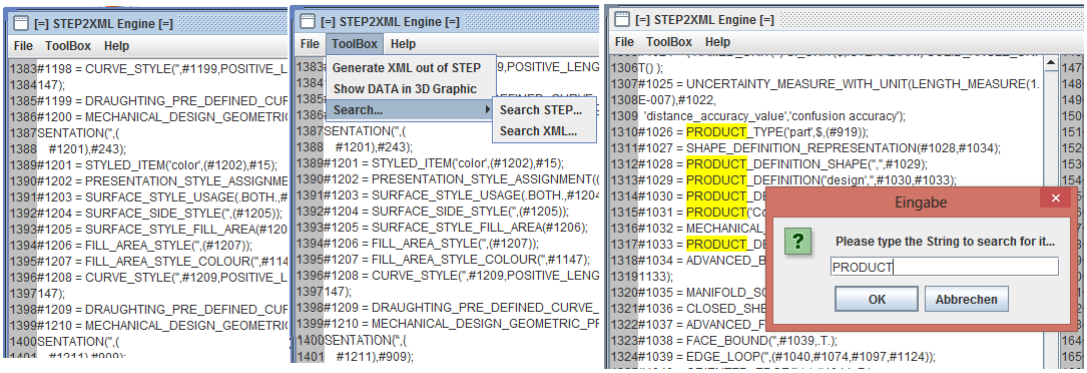

Figure 59: Text bzw. String in STEP-Daten sowie XML Daten suchen

Prozess 6: Ein Status-Bericht als PDF-Datei exportieren

Der Benutzer benötigt einen Bericht als PDF-Datei, der die Ergebnisse und Eingabedaten beinhaltet. Dieses Vorgehen ist durch folgende Schritte möglich:

- zuerst sollte das Programm gestartet und die Inhalte beider Dateien (XML und STEP Inhalte) sollen in der Textbereiche geladen sein
- der Benutzer geht mit dem Mauszeiger auf die MenuBar und klickt auf den Eintrag *"File"*
- es erscheint ein Menü mit mehreren Funktionen zur Auswahl
- der Benutzer soll mit dem Mauszeiger den Menueintrag *"Export Report As PDF"* anklicken
- daraufhin öffnet sich ein Datei-Browser Fenster
- der Benutzer wird aufgefordert, einen Ordner für den zu erstellenden Bericht auszusuchen und dies mit der Bestätigungstaste zu bestätigen
- am Ende schließt sich das Datei-Browser Fenster und der Bericht wird im Hintergrund als PDF-Datei generiert. Der Dateiname wird aus drei Teilen bestehen:
  - der STEP-Dateiname
  - Name des Benutzers (Windows PC-Username)
  - ein String "*_STEP2XMLReport*"

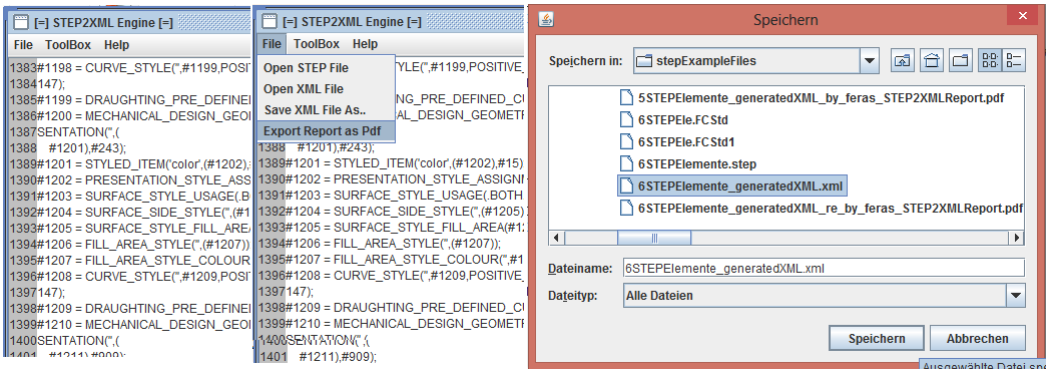

Figure 60: Ein Status-Bericht als PDF-Dokument exportieren

5 Ergebnisse und Analyse

5.1 3D-Remodellierung der Daten

Die graphische Modellierung der XML-Daten in 3D-Zeichnung kann in vielen Aspekten verbessert werden. Ein Beispiel für diese Verbesserungen bei der 3D-Modellierung ist eine korrektere Positionierung der geometrischen Objekte.

Genauso bei der Erzeugung der Primitiven wird z.B. bei Sphere keine glatte Oberfläche erzeugt. Deshalb soll eine bessere Funktion bzw. mächtig-ere 3D Java-Bibliothek beispielsweise für die Zeichnung der Sphere verwendet werden.

5.2 XML - Architektur

Die XML - Architektur ist nicht ausreichend mit Variablen und Elementen ausgestattet, womit weitere geparsste Daten aus der STEP-Modelldaten angelegt werden können.

Die Struktur der XML-Architektur für die STEP-Daten ist nicht optimal genug konzepiert. Es besteht Bedarf für weitere Verbesserungen, sodass detaliertere STEP-Daten abstrahiert werden können.

5.3 Modell-Erkennung und Remodellierung der Produktzeichnug

Bis jetzt werden die geometrischen Primitiven aus den STEP-Daten anhand einiger STEP-Schlüsselwörter identifiziert und mithilfe von fertigen bzw. vordefinierten Klassen remodelliert. Dieses Vorgehen klappt nur bei vordefinierten sechs Primitiven, die sehr simple 3D Modell gehören. Andererseits wenn man etwas komplexere Produkt- oder Teilprodukt-Zeichnungen hat, wird dieses Vorgehen nicht funktionieren.

5.4 Abgrenzungen und Probleme beim Datenaustausch der STEP Standards

Die erste Version von STEP wurde im Jahre 1984 definiert und erst nach 10 Jahren im Jahre 1994 wird der Standard eingesetzt. Innerhalb dieser Zeit haben sich die Hardware und Technologien weiter entwickelt. Durch diese rasante Weiterenwicklung war dieser Standard von 1984 nicht mehr auf der aktuelle Stand von 1994. Diese Zeitlücke von 10 Jahren hat dazu geführt, dass STEP nicht die volle Fähigkeit und Brauchbarkeit für die weiter entwickelten moderne CAD-Systeme besitzt.

Im Rahmen der Recherche und Anwendung vom STEP Standard in meiner Diplomarbeit komme ich zu verschiedenen Erkenntnissen bezüglich den Abgrenzungen und Problemen bei der Anwendung vom STEP Standard für den 3D-Modelldaten-austausch.

Die modernen CAD-Systeme haben verschiedene Aspekte und Vorgehen bei der Modellierung, wie z.B. die Generierung von parametrisch-, *constraint* basierter- und *Feature*-basierter CAD-Modellierung. STEP kann diese diese Modellierungs-aspekte und -Arten nicht definieren, und damit sind parametrische Entitäten, *Constraints* oder *Features* bei STEP nicht repräsentativ.

Ein weiteres großes Problem, das man bei einem weiteren Ausbau von STEP finden wird, ist die Zweigliederung zwischen expliziten oder deklarativen Modellen und impliziten oder Vorgehensmodellen.

Am Anfang wurden die Modellierungen hauptsächlich als Randdarstelungen in Bezug auf Flächen, Kanten, Ecken etc. definiert und mit allen Geometrien im Modell explizit angegeben. Später wurde das Vorgehen der Kunstruktive Solid Geometrie Modellierung um eine Reihe von Operationen und Berechnungen in der Konstruktion definiert. In diesem Fall existieren möglicherwese keine primitive (Low-Level) geometrische oder topologische Elemente im Modell. Heutzutage benutzen einige modernere CAD-Systeme Kombination aus beiden Verfahren.

Ein weiteres Problem bei Version 1 von STEP ist, dass hauptsächlich für den Datenaustausch von statischen und nicht parametrischen (Boudary Representation) Modellen, die free-form Geometry in form NURBS beinhalten, verwendet wurde. Mit diesen Leistungsfähigkeiten kann STEP nur simple Modelle ohne Topologien etc. erstellen

Momentan ist es nur möglich die Basis *BREP*-Daten mit STEP zu modellieren und auszutauschen, wobei alle anderen Modellierungsdaten durch den Datenaustausch verloren gehen.

Eine der Schwächen bzw. Schwierigkeit mit STEP-Daten zu arbeiten, ist die Erkennung von Mustern in den STEP-Zeilen. Als Beispiel dafür betrachten wir folgenden Fall:

In dem Verfahren des Parsens wird jede STEP-Zeile nach den Referenzen zu anderen STEP-Zeilen durchsucht. In den meisten STEP-Zeilen wird Bezug auf bestimmte STEP-Zeilennummern genommen. Diese Zeilenummern sind in Klammern mit Rauten und durch kommas getrennt und am Ende vom STEP-Satz definiert. Diese Definition nehme ich als Muster für das Parsen der Referenzen bzw. der Zeilennummern. Das Problem dabei, dass trotzdem solche STEP-Zeilen existieren, die nicht nach diesem Schreibmuster definiert sind. Somit enstehen Fehler und Lücken bei der Behandlung und dem Parsen der Referenzen aus den STEP-Zeilen.

6 Zusammenfassung und Fortsetzung weiterer Arbeiten

In dieser Arbeit ging es darum, ein Vorgehen für das Parsen und Umwandeln der STEP-Daten in XML strukturierten Daten zu erforschen. Dabei wurde der STEP Standard mit seiner komplexen und aufwendiger Struktur erst analysiert. Es sind verschiedene Schlüßelwörter, Funktionen und Definitionen in STEP-Daten, die untersucht wurden.

Für diesen Zweck wurde im Rahmen dieser Diplomarbeit ein Prototyp (Software) entwickelt, der aus einer Sammlung von mehreren Funktionen und Algorithmen besteht, die in Klassen anhand der Aufgabengebiet aufgeteilt sind. Diese Funktionen und Algorithmen sind für das Parsen und Interpretieren von STEP- und XML-Daten zuständig.

Dabei wurde der Versuch gemacht, Datenmuster in den STEP-Daten zu erkennen und die Daten entsprechend zu interpretieren. Anschließend werden diese Daten in Java-Klassenobjekte in Form von Attributen, ArrayLisiten und Elementen abgebildet.

Die abgebildeten STEP-Daten aus den Objekten werden für weitere Bearbeitungs- bzw. Parsen-Schritte bereitgestellt. Auf anderer Ebene des Parsen-Prozesses werden die Daten, die primitiven geometrischen Objekte wie Box, Pyramide, Cone, Torus etc. repräsentieren, in XML-Objekte, die später als XML-Elemente im Rahmen einer XML-Struktur abgelegt werden, abgebildet.

Im Laufe der Arbeit und mit dem Versuch die STEP-Daten in XML-Daten strukturiert darzustellen fanden sich Problemen und Schwierigkeiten, da der STEP Standard verschiedene Abgrenzungen und Probleme bei dem Datenaustausch aufzeigt. Solche Probleme, die bei der Modellierungsdaten-Definition der STEP-Daten nicht zu 100% richtigen Daten für die Remodellierung der 2D und 3D-Zeichnungen der ursprünglichen CAD-Zeichningen führen.

Einige dieser Probleme befassen sich mit der Abgrenzung von STEP bei der Modellierungsaspekte und -Arten wie sie in moderne CAD-Systemen definiert sind. Damit gehen viele Modellierungsinformationen über die Relationen zwischen Produktteile oder den Zeichnungswinkel der Objekte auf dem Achsensystem etc. verloren.

Ein weiteres Problem gab es bei der Daten-Muster Erkennung in den STEP-Datenzeilen. Es ist schwierig einheitliche Muster in den STEP-Daten zu finden, womit dieses Problem das Parsen und die Interpretation der Daten erschwert. Aufgrund dessen müssen dann viele Ausnahmefälle in den Parsen- und Mustererkennung-Algorithmen behandelt werden.

In den folgenden Punkten können sich zukunftige Arbeiten beschäftigen und verschiedene Aspekte und Forschungspunkte verbessern und erweitern:

- Mit weiteren STEP-Schlüsselwörtern können weitere Produktteile bzw. geo-metrische Objekte, die komplexer aufgebaut sind, erkannt und nachmodelliert werden.

- Die Software-Architektur kann verbessert werden, sodass die Erkennung geometrischer Objekte nicht nur auf den Primitiven begrenzt bleibt, sondarn jedes Produkt oder Teilprodukt anhand XML-Daten in 3D nachmodelliert werden kann.

- Für weiterführende Arbeiten kann die Software Architektur so entworfen werden, dass die Modell-Erkennung und die Remodellierung der STEP-Daten dynamisch und für jede Art Zeichnung funktioniert.

  Eine Idee für ein besseres Vorgehen ist die Entwicklung eines Modellerkennung-Algorithmus, das anhand aller Koordinatenpunkte der Zeichnung , anstelle der Erkennung der Primitiven, funktioniert. Desweiterin gehört auch die Betrachtung der Kurven, Linien und Flächen zwischen den Koordinatenpunkten dazu.

References

[Uc09] Uhllenboom, C.: Java ist eine Insel 2009, Programmieren mit der Java Platform, Standard Edition 6

[SFS06] St.Laurent, S.; Fritzgerald, M.; Deutsche Übersetzung von Schüler, N.: XML kurz&gut. 3.Auflage, 2006

[Krüg] Krüger, Guido; Stark, Thomas: Handbuch der Java-Programmierung : Standard-Edition Version 6. 2009

[Daum08] Daum, Berthold: Java-Entwicklung mit Eclipse 3.3 : Anwendungen, Plugins und Rich Clients, 2008

[Daum03] Daum, Berthold: Java-Entwicklung mit Java 2 : Plugins und Anwendungen implementieren mit SWT und JFace, 1.Aufklage 2003

[Uml2] Rupp, Chris; Queins, Stefan; Zengler, Barbara: UML 2 glasklar: Praxiswissen für die UML-Modellierung. 3., aktualsierte Auflage, 2007

[Dipl06] Berger, Felix: A test and verification enviroment for Java Programmers, Universität Stuttgart 2006

[Assi] Assisi, Ramin: Eclipse : Einführung und Referenz; Java-Entwicklung mit der Open-Source-Plattform

[FchStu08] Haufler, Andreas; Staudenecker, Andreas; Wobser, Alexander: Vergleich von XML-Datenbanken, Universität Stuttgart 2008

[Dipl04] Opletal, Sascha : Einsatz einer XML-Anfragesprache für Transformationszwecke. Universität Stuttgart 2004

[Vlis02] Vlist, Eric van der: XML schema : [the W3C´s object-oriented descriptions for XML]. 2002

[Abit00] Abiteboul, Serge; Bauneman, Peter; Suciu, Dan: Data on the Web : from relations to semistructured data and XML, 2000

[FchStu07] Albiez, Martin; Nothelfer, Frederik; Scherer, Wolfgang: Brauchbare UML-Werkzeuge, Universität Stuttgart 2007

[Blaha05] Blaha, Michael; Rumbaugh, James: Object-oriented modeling and design with UML : [UML2]. 2.edition 2005

[Weik06] Weilkiens, Tim: Systems engineering mit SysML, UML : Modellierung, Analyse, Design. 1. Auflage 2006

[Fowl04] Fowler, Martin: UML distilled : a brief guide to the standard object modelling language. 2004

[Born04] Born, Marc; Holz, Eckardt; Kath, Olaf: Softwareentwicklung mit UML 2 : die "neuen" Entwurfstechnicken UML 2, MOF 2 und MDA ; [die Sprache im Deteil; Metamodell Sprache; Diagramme]. 1.Auflage 2004

[Reus09] Reussner, Ralf; Hasselbring, Wilhelm: Handbuch der Software-Architektur. 2.Auflage 2009

[VACIMNVZ] Vogel, O.; Arnold, I.; Chughtai, A.; Ilhler, E.; Mehling, U.; Neumann, Th.; Völter, M.; Zdun, U.: Software-Architektur Grundlagen - Konzepte - Praxis. 1.Auflage 2005

[Andl] Anderl, R.; Trippner, D.: STEP STandard for the Exchange of Product Model Data, Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP). 2000

[Own-93] Owen, Jon.: STEP : an introduction ,Information Geometers- 1. publ.. - Winchester. 1993

[Doct-99] Mitschang, B.; Ertl, T.: Konzepte und Techniken der Datenversorgung für komponentenbasierte Informationssysteme, Fakultät Informatik der Universität Stuttgart. 1999

[Sell-00] Sellentin, Jürgen: Datenversorgung komponentenbasierter Informationssysteme : [zielgerichtete Einführung in die Standards STEP und CORBA; Klassifikation von Datenquellen und möglichen Zugriffsverfahren; Diskussion von Datenversorgungsstrategien im Intra-/Internet am Beispiel JavaSDAI]. 2000

[Alf-Parser] Alfred, V Aho; JEffrey, D.Ulman: The Theory of Parsing, Translation, and Compiling. 1972

[Wiki-1] Wikipedia: Standard of the exchange of product model data

[1] Steven Metsker: Building Parser with Java. 2001

[Java3D] Pfeiffer, Michael: Java 3D 1.3 für Einsteiger und Fortgeschrittene, Java 3D 1.3.2

[EclipseHD] Eclipse Java EE IDE- Workbench User Guide

[2] Brüderlin, Beat; Roller, Dieter: Geometric Constraint Solving and Applications. 1998

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Firas Zoabi Stuttgart, den 09.09.2014