Manuel Stitz

Entwicklung und Anwendung einer Sprache zur Beschreibung von CAD-Wiederholteilen

Bachelorarbeit

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fakultät Maschinenbau

Mittweida, 2010

Manuel Stitz

Entwicklung und Anwendung einer Sprache zur Beschreibung von CAD-Wiederholteilen

eingereicht als

Bachelorarbeit

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Maschinenbau

Chemnitz, 2010

Erstprüfer: Prof. Dr.-Ing. Martin Zimmermann
Zweitprüfer: Dipl. Inf. Chris Hübsch

vorgelegte Arbeit wurde verteidigt am: 29. Juni 2010

Bibliografische Beschreibung:

Stitz, Manuel:
Entwicklung und Anwendung einer Sprache zur Beschreibung von CAD-Wiederholteilen. - 2010. - 67 S. Mittweida, Hochschule Mittweida (FH), Fakultät Maschinenbau, Bachelorarbeit, 2010

Referat:

Development and application of a language to describe CAD reusable parts.

Ziel der Bachelorarbeit ist die Erstellung einer Sprache, um parametrische CAD-Wiederholteile beschreiben zu können. Schwerpunkt ist die Implementierung CAD-System-übergreifender Konstruktionselemente. Gleichzeitig soll diese Sprache von keinem CAD-System abhängig sein.
Der langfristige Zweck dieser Entwicklung ist die Ablösung und Umwandlung der vorhandenen Geometriedateien inklusive der in den Dateien verwendeten Methoden der Norm ISO 13584-31.
Für die Umsetzung ist ein geeignetes Werkzeug zu finden. Dabei muss das Umfeld des Geometrieerstellungsprozesses beachtet werden. Beeinflusst wird die Wahl auch durch die Programmierung eines prototypischen und für die Anwendung notwendigen Generators. Mit diesem werden durch Verwendung der API eines CAD-Systems Teile erzeugt, nachdem die Dateien der neuen Sprache eingelesen und interpretiert wurden. Die Sprache wird letztendlich auf Basis von Xtext, einem Framework zur Erstellung von textuellen Sprachen, umgesetzt.
Weiterhin wird anhand eines Beispiels die Tauglichkeit der Sprache bewiesen.

Inhaltsverzeichnis

Quelltextverzeichnis iv

Abbildungsverzeichnis v

Tabellenverzeichnis vi

Abkürzungsverzeichnis vii

1 Einleitung 1

1.1 Motivation 1
1.2 Beauftragendes Unternehmen 4
1.3 Zielstellung 4
1.4 Abgrenzung 5

2 Derzeitiger Stand der Geometrieerzeugung 7

2.1 Überblick 7
2.2 Merkmaldateien 8
2.3 Instanzauswahl 9
2.4 Temporäre Datei 10
2.5 ISO 13584-31 und Geometriedateien 11
2.6 Geometrieerzeugungsprogramm 13
2.7 Problemanalyse und Formulierung der Aufgabenstellung 14

3 Entwicklung einer neuen Sprache 18

3.1 Anforderungen an die neue Sprache 18
3.2 Aufbau der neuen Sprache 21
3.3 Vergleich mit Fortran und ISO 13584-31 27

4 Umsetzung 30

4.1 Java und Eclipse 30
4.2 Eclipse Modeling Framework 31
4.3 Xtext 34
4.4 ANTLR 36

5 Werkzeuge zur Anwendung **37**
5.1 Initialisierung der Übergabewerte 37
5.2 Steuerung der Geometrieerzeugung 37
5.3 Generator zur Geometrieerzeugung 39
5.4 Optionen 39

6 Komplexbeispiel **40**

7 Zusammenfassung und Ausblick **48**

Anhang **52**

A Auflistung der Anreicherungen **53**

B vollständige Geometriedatei **54**

C vollständige Xtext-Grammatik des Neutral-Formats **58**

Literaturverzeichnis **65**

Quelltextverzeichnis

2.1 Auszug aus den Merkmaldatensätzen einer Merkmaldatei. Jede Zeile repräsentiert eine Instanz (ein * ist die Fortsetzung einer Zeile). . . . 9
2.2 Auszug aus einer Geometriedatei. Methoden der ISO 13584-31 befinden sich z. B. in den Zeilen 1, 14, 15, 19 und 30. 11

6.1 Geometriedatei zur Erstellung einer Rotation 40
6.2 Xtext-Grammatik von Rotation und Skizze 42
6.3 Eine Neutral-Format-Datei zur Beschreibung einer Rotation 43
6.4 Auszug aus einer Geometriedatei. Die ISO 13584-31-Methode *LIN_CHAMFER_2_LIN* definiert die Erstellung einer Fase. 44
6.5 Xtext-Grammatik für die Definition einer Fase aus einem Wert und einem Winkel 45
6.6 Neutral-Format-Quellcode zur Beschreibung einer Fase 45

Abbildungsverzeichnis

1.1 Abbildungsverzeichnis 2
1.2 Familientabelle 3

2.1 Schema der derzeitigen Geometrieerzeugung 8
2.2 REMARC Mapping-Editor 10
2.3 Aufruf der F-Routinen 14

3.1 Einteilung der Features 22
3.2 Einteilung der Skizzenelemente 25

4.1 EMF vereinigt die Technologien Java, XML und UML 32
4.2 EMF-Modelleditor 33
4.3 Texteditor von Xtext 35

6.1 Grundkörper einer Schraube, aus einer nf-Datei erstellt 44
6.2 Grundkörper einer Schraube mit Fase, aus zwei nf-Dateien erstellt . . 46
6.3 aus Geometriedateien erzeugte Schraube 47
6.4 aus nf-Dateien erzeugte Schraube 47

Tabellenverzeichnis

3.1 Vergleich zwischen Fortran mit der ISO 13584-31 und dem Neutral-Format 29

4.1 Überblick über die API-Programmiersprachen der CAD-Systeme Pro/E und NX 31

Abkürzungsverzeichnis

ANTLR	ANother Tool for Language Recognition
API	Application Programming Interface
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CATIA	Computer Aided Three-Dimensional Interactive Application
CSG	Constructive Solid Geometry
DIN	Deutsches Institut für Normung e. V.
DLL	Dynamic Link Library
DSL	Domain Specific Language
EMF	Eclipse Modeling Framework
EN	Europäische Norm
ISO	Internationale Organisation für Normung
nf	Neutral-Format
Pro/E	Pro/ENGINEER
SCS	Sketch Coordinate System
XML	eXtensible Markup Language

1 Einleitung

Diese Arbeit beschäftigt sich mit einem Thema aus dem Bereich des Maschinenbaus. Es werden Grundkenntnisse der Programmierung und das Verständnis von Begriffen und Techniken aus dem Gebiet des Computer Aided Design (CAD) vorausgesetzt.

1.1 Motivation

Für die Fertigung von Fahrzeugen, Maschinen u. ä. gibt es von der Idee bis zum Vertrieb viele verschiedene Schritte. Nach der Planung erfolgt die Entwicklung, wobei diese Phase oft mit der Konstruktion zusammenfällt. Für die Konstruktion werden die genannten Erzeugnisse von Konstrukteuren in CAD-Systemen modelliert, in dem sie Einzelteile erstellen und diese zu Baugruppen zusammensetzen. Diese Einzelteile wiederum können Einzelstücke sein, also einmal konstruierte und gefertigte Teile, aber auch Wiederholteile. Wiederholteile sind entweder Variantenteile, die sich in Normteile, Werknormteile und andere gliedern, oder Gleichteile, wie Abbildung 1.1 zeigt.

- **Normteile** werden durch nationale und internationale Organisationen, wie das Deutsche Institut für Normung e. V. (DIN) oder die Internationale Organisation für Normung (ISO), in ihren Formen und Abmessungen vorgegeben.
- **Werksnormen** sind innerbetriebliche Normen.
- **Gleichteile** sind ungenormte Bauteile mit gleichbleibender Gestalt, die aber oft verwendet werden (z. B. Airbag).

Die Baugruppen werden lange vor ihrer Herstellung von Konstrukteuren an Computern in CAD-Systemen modelliert. Für ein effizientes Konstruieren werden einmal erstellte Bauteile abgespeichert, um sie bei erneutem Gebrauch sofort verwenden zu

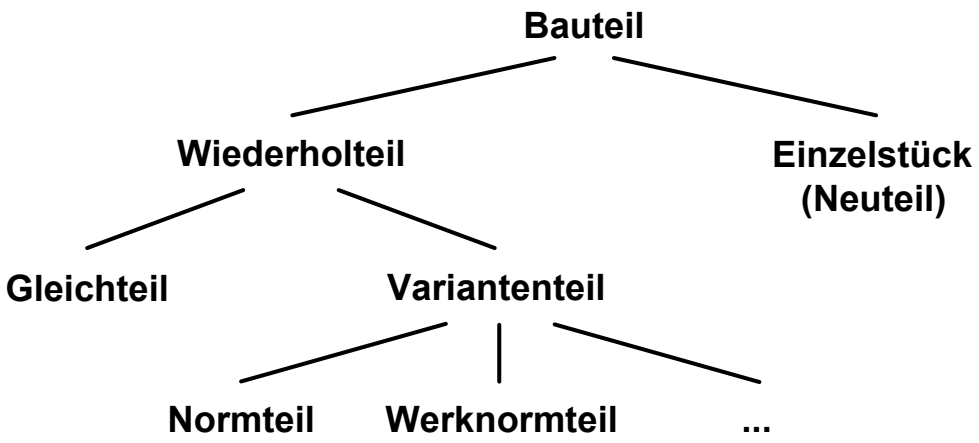

Abbildung 1.1: Einteilung von Bauteilen

können. Vor allem bei Normteilen kommt es jedoch oft vor, dass neu modelliert werden muss, weil die gesuchte Ausprägung[1] noch nicht vorhanden ist. Dies bedeutet eine wachsende Anzahl von sich ähnelnden Bauteilen. Die Folge ist eine große und wachsende Datenbank, was auch zu einem erhöhten Verwaltungsaufwand führt. Letzteres ist allerdings auch eine Frage einer funktionierenden Verwaltung, denn dafür gibt es speziell für den Bereich der rechnergestützten Entwicklung (CAE) entsprechende Datenverwaltungssysteme. Die Verbesserung liegt darin, jede Ausprägung einer Norm bzw. Werksnorm nicht einzeln zu modellieren und zu speichern, sondern parametrische Modelle zu entwerfen und alle Ausprägungen davon abzuleiten. Dabei kann zwischen drei Möglichkeiten ausgewählt werden:

1. Es wird ein Template erstellt, also eine geometrische Vorlage, in der die Ausprägungen über Parameter gesteuert werden. Die Veränderung der Parameter erfolgt entweder in dem betreffenden CAD-System selbst oder wird mit Hilfe einer Programmier-Schnittstelle (API) angesprochen.

2. Es wird eine Familientabelle erstellt. Der Unterschied zu dem Template besteht darin, dass hier eine Liste mit allen möglichen Ausprägungen enthalten ist, wie in Abbildung 1.2 zu sehen ist.
   Der Nachteil von Templates und Familientabellen ist, dass sie an ein bestimmtes CAD-System gebunden sind - sie sind nativ. Das bedeutet: Will ein Kon-

[1] Die verschiedenen Teile einer Normreihe können auch als Ausprägungen oder Instanzen bezeichnet werden.

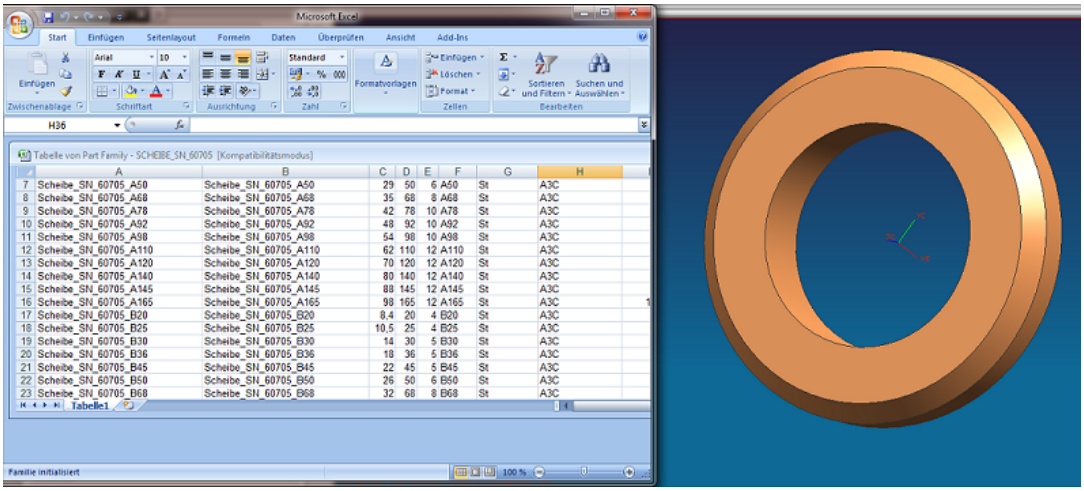

Abbildung 1.2: Familientabelle; Rechts ist das Teil dargestellt, wie es auch in einem Template enthalten ist, links die Tabelle mit allen Instanzen.

strukteur ein Normteil in einem anderen CAD-System verwenden, so kann er das Template des ersten CAD-Systems nicht nutzen. Die einzigen Lösungen sind, dass er das Template in dem anderen CAD-System neu modelliert oder ein Werkzeug verwendet, welches die nativen Daten in die des Ziel-Systems umwandelt. Solche Werkzeuge gibt es[2], diese sind jedoch entweder sehr teuer (50.000 € und mehr), nicht für parametrische Modelle geeignet oder können nicht alle Daten korrekt konvertieren. Eine eigene Programmierung würde sich nur lohnen, wenn die Anzahl der umzuwandelnden Vorlagen bzw. Daten dafür entsprechend hoch ist und somit die Programmierung effizienter als die Neu-Modellierung wäre.

3. Es wird eine nicht-native Abbildung der Norm bzw. Werksnorm erstellt. Das heißt, dass der Aufbau des Modells in einer Reihe von Anweisungen gespeichert wird. So könnten beispielsweise zuerst alle Punkte definiert werden, die danach zu Linien verbunden und anschließend zu Flächen und Volumenkörpern zusammengefügt werden. Die Erzeugung des konkreten Modells im CAD-System erfolgt über deren API.

[Bug96]

[2] zum Beispiel *Asfalis* von CAMTEX (http://www.camtex.de/asfalis-Schnittstellen.html) oder *Feature Based* von CoreTechnologie (http://www.coretechnologie.com/p_depf.htm)
Allerdings ist laut Aussage von CAMTEX die Verwendung rückläufig.

1.2 Beauftragendes Unternehmen

Die ARC Solutions GmbH ist ein mittelständiges Unternehmen in Chemnitz. Sie bietet verschiedene Dienstleistungen im Bereich Maschinenbau an und vertreibt neben mehreren CAD-Systemen auch das firmeneigene Produkt REMARC. Dieses dient zur Verwaltung und Erzeugung von Wiederholteilen und ist somit ein Beispiel für die im vorigen Abschnitt erwähnten Datenverwaltungssysteme. Zur Teileerstellung nutzt REMARC alle drei der oben erwähnten Varianten der Bereitstellung von Modellen. Den größten Teil macht jedoch die dritte Möglichkeit aus, also die nicht-native Abbildung einer Norm.

1.3 Zielstellung

Diese Arbeit beschäftigt sich mit der dritten genannten Möglichkeit: eine Sprache zur Beschreibung von CAD-Normteilen. Eine solche Sprache gibt es bereits und sie ist in der ISO 13584-31 definiert. Diese Norm ist aber inzwischen mehr als 15 Jahre alt und entspricht nicht mehr den derzeitigen Anforderungen. Eine Erweiterung wäre zwar möglich, ist aber nicht sinnvoll. Die Implementierung der Norm ist für die Programmiersprache Fortran gedacht. Dadurch ist der Wortschatz der ISO um den von Fortran erweitert. Es ist jedoch besser, wenn alle verwendbaren Funktionen der Sprache vorgegeben sind. Weiterhin hat sich die Art der Konstruktion verändert. Methoden der ISO müssten umgestaltet, hinzugefügt und weggelassen werden, was aber schwierig ist, da der komplette Satz von Methoden aufeinander aufgebaut ist. Das Ziel ist daher die Entwicklung einer neuen, verständlicheren Sprache, die die alte ersetzen soll. Dabei soll sich an aktuelle Modellierungsrichtlinien gehalten werden.
Diese neue Sprache soll alle benutzbaren Funktionen und deren Verwendung und Kombination vorgeben, nicht aber die Reihenfolge bei der konkreten Anwendung. Sie gibt sozusagen ein Gerüst vor, mit dem individuelle Modelle erzeugt werden können. Dies ist mit einer Programmiersprache vergleichbar, bei der Schlüsselwörter und Syntax vorgegeben werden und der Anwender daraus seine eigenen, spezifischen Quelltexte schreibt. Der Funktionssatz der Sprache muss im Hinblick auf das Anwendungsgebiet und diese Arbeit eingegrenzt werden, was im nachfolgenden Kapitel erfolgen wird.

Weiterhin wird ein Programm benötigt, welches die Beschreibung eines Modells einliest, interpretiert und die entsprechenden Funktionen der API ausführt. Hierbei kann das bestehende Programm von REMARC nur als Hilfestellung dienen. Eine Erweiterung bzw. Veränderung ist nicht zielführend, da das bisherige Programm an die alte Sprache angepasst ist. Es muss daher komplett ersetzt werden.

1.4 Abgrenzung

Im Rahmen dieser Arbeit soll keine vollständige Sprache entstehen, mit der alle erdenklichen Wiederholteile beschrieben werden können. Die Sprache soll aber so entwickelt werden, dass Erweiterungen möglich sind. Für den Anfang soll sich auf die Beschreibung von Normteilen beschränkt werden. Es sollen nur einfache Geometrien und Features beschrieben werden, was z. B. Freiformflächen und Extrusionen entlang frei definierter Kurven ausschließt.

Die entstehende Sprache dient zunächst einmal dazu, die alte Sprache abzulösen, jedoch noch nicht die Templates und Familientabellen. In welcher Form dies geschehen soll und ob eine vollautomatische Umwandlung in die neue Sprache möglich ist, hängt teilweise von der API des betreffenden CAD-Systems ab und wird an dieser Stelle nicht betrachtet. Andererseits soll es nicht möglich sein, neue Templates und Familientabellen zu erzeugen.
Weiterhin sollen mit der Sprache einzelne Bauteile aber keine Baugruppen beschrieben werden.

In der Sprache wurden Bestandteile weggelassen, deren nachträgliche Einbindung mit geringem Aufwand erfolgen kann. Zum Beispiel ist die Erweiterung der Sprache um die Definition eines Zylinders trivial, wenn schon ein Block enthalten ist.

Auch wird sich in dieser Arbeit auf ein einziges CAD-System beschränkt, da die Betrachtung mehrerer Systeme den Rahmen der Arbeit sprengen würde. Das bringt mit sich, dass sich die Sprache nur an das verwendete System und dessen API anlehnt. Bei Einbeziehung mehrerer CAD-Systeme, ist die Gefahr geringer, bei späterer Implementierung weiterer CAD-Systeme die Sprache korrigieren zu müssen, damit sie auf alle CAD-Systeme abbildbar ist. Da die neue Sprache jedoch weitgehend die

grundlegenden Fähigkeiten eines CAD-Systems berücksichtigt, sollte eine spätere Korrektur minimal sein.
Das verwendete CAD-System ist *NX* von *Siemens PLM Software*.

2 Derzeitiger Stand der Geometrieerzeugung

Das folgende Kapitel zeigt die derzeitige Erzeugung von Teilen und geht auf einzelne Komponenten ausführlich ein. [Sti09] Zum Schluss werden Probleme beschrieben und eine Aufgabe formuliert, die die Ausgangssituation für das nachfolgende Kapitel darstellt.

2.1 Überblick

Der erste Schritt zu einem Bauteil ist die Auswahl einer Norm in REMARC. Im Mapping-Editor von REMARC (siehe Kapitel 2.3) wird als Nächstes die gewünschte Instanz ausgewählt, wobei dafür ein Interpreter (*BSV*) angesprochen wird, der alle Werte aus einer Merkmaldatei liest. Danach wird eine temporäre Datei (*d99999.dat*) erstellt, die von dem Geometrieerzeugungsprogramm *ARCComponentEngine* eingelesen wird. In dieser Datei sind Parameter enthalten, die für den Aufbau des Bauteils erforderlich sind. Die ARCComponentEngine ruft nun *Geometriedateien* auf und übergibt an diese die Parameter. Diese Geometriedateien stellen bei ARC Solutions die in der Einleitung erwähnte Beschreibung des Modellaufbaus dar. Sie beinhalten Anweisungen für die Erzeugung von *Elementen* wie Punkte, Linien, Kreise u. a., aus denen dann mit Hilfe von geometrischen Operationen (*Features*), wie Extrudieren oder Rotieren, Volumenkörper entstehen. Die übergebenen Parameter steuern einerseits die Größe und Position, andererseits die optionale Einbindung von Elementen und Features. Das Bauteil wird schließlich durch den Aufruf von API-Funktionen einer Bibliothek erstellt.
Der Zusammenhang aller Parameter und Programme ist in Abbildung 2.1 dargestellt.

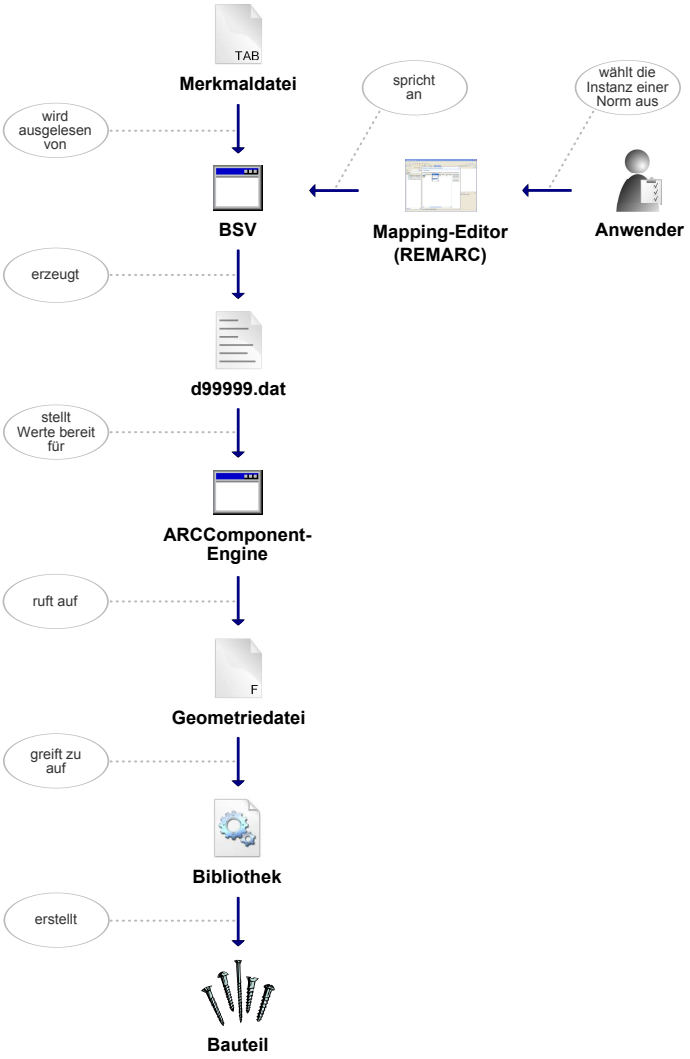

Abbildung 2.1: Schematische Darstellung der derzeitigen Geometrieerzeugung. Der Fluss von oben nach unten zeigt den Weg der Parameter aus der Merkmaldatei. Der Anwender beeinflusst dabei den Satz der Parameter.

2.2 Merkmaldateien

Die Merkmaldateien werden von der DIN Software GmbH, einer Tochtergesellschaft des DIN, bezogen. „Sie hat die Aufgabe, Dateien und Programme zu beschaffen, herzustellen und zu vertreiben, die bei der Herstellung DIN-normgerechter Erzeugnisse helfen und die Anwendung DIN-normgerechter Verfahren fördern.“ [Inn] Die

Merkmaldateien sind nach DIN V 4001 genormt und beinhalten „die Merkmale zur Beschreibung der verschiedenen Ausprägungen eines Normgegenstandes“ ([din90], S.69). Für die Auswahl der im vorigen Kapitel genannten Parameter sind die Merkmaldatensätze (= Gesamtheit aller Instanzen) ausschlaggebend. Quelltext 2.1 zeigt einen Auszug aus den Merkmaldatensätzen, wobei jede Zeile eine Instanz repräsentiert (ein * ist die Fortsetzung einer Zeile).
Die Merkmaldateien enthalten weitere Angaben wie Normnummer oder Komplettteil-Nummer, die für die richtige Zuordnung von Norm und Geometriedatei zuständig sind. (Zu Komplettteil: siehe Kapitel 2.5)

```
1  C  AE  IDNR FA FA2 EAA  A02  A03  A04  B  C    E1   F  ABF
2  C    ABG   BAK
3  DA, 1, 10, 1, 1, 'H', 'M', 1.6, 0.35, 3, 1.30, 3.0, 1, 1.6,
4  *   0.90, 0.31
5  DA, 1, 20, 1, 1, 'H', 'M', 1.6, 0.35, 4, 2.30, 3.0, 1, 1.6,
6  *   0.90, 0.31
7  DA, 1, 30, 1, 1, 'H', 'M', 1.6, 0.35, 5, 3.30, 3.0, 1, 1.6,
8  *   0.90, 0.31
9  DA, 1, 40, 1, 1, 'H', 'M', 1.6, 0.35, 6, 4.30, 3.0, 1, 1.6,
10 *   0.90, 0.31
11 DA, 1, 50, 1, 1, 'H', 'M', 1.6, 0.35, 8, 6.30, 3.0, 1, 1.6,
12 *   0.90, 0.31
```

Quelltext 2.1: Auszug aus den Merkmaldatensätzen einer Merkmaldatei. Jede Zeile repräsentiert eine Instanz (ein * ist die Fortsetzung einer Zeile).

2.3 Instanzauswahl

Der Mapping-Editor von REMARC ist die grafische Oberfläche zum Selektieren der gewünschten Instanz (siehe Abbildung 2.2). Dabei muss jede Spalte mit einem Wert aus der angezeigten Liste gefüllt sein. Die Menge der verfügbaren Werte ist

durch Analysieren der zu der Norm zugehörigen Merkmaldatei durch den Interpreter vorgegeben. Jede Auswahl eines Wertes durch den Anwender wird sofort verarbeitet und es erfolgt eine Aktualisierung der selektierbaren Werte der anderen Spalten.

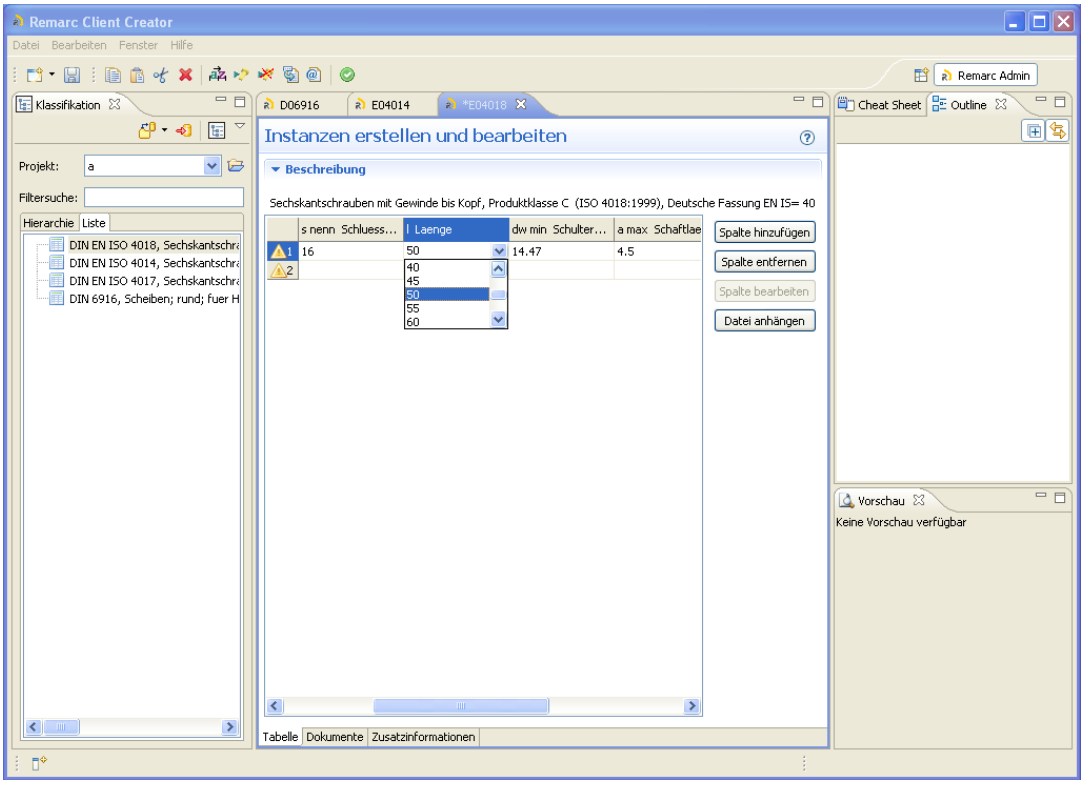

Abbildung 2.2: Das Programm REMARC zur Erzeugung von Teilen. In der Mitte ist der Mapping-Editor, in dem die Instanz einer Norm ausgewählt wird.

2.4 Temporäre Datei

Bei einer vollständig definierten Instanz, also wenn jede Spalte mit einem Wert gefüllt ist, erzeugt der Interpreter die d99999.dat. Diese enthält neben den gewünschten Parametern weitere norm- und instanzspezifische Werte. Dazu gehören die Komplettteil-Nummer und die Normnummer, was für das weitere Verständnis wichtig ist.

2.5 ISO 13584-31 und Geometriedateien

Die ISO 13584-31 gibt eine logische Beschreibung einer Schnittstelle zwischen einem Teilebibliotheksystem und einem Produktmodellierungssystem an. Die Schnittstelle ermöglicht den Austausch von parametrischen Formteilen, welche die implizite Geometrie von Teilefamilien im Format eines parametrischen Programms beschreiben. [nor12] Diese Norm definiert jedoch nur das „Gerüst“ der Methoden und nicht deren Inhalt. Das heißt, die Funktionalität, die Übergabeparameter und der Rückgabewert sind vorgegeben. Nicht aber, was bei einem Aufruf der Funktion geschieht. Das liegt im Ermessen des Anwenders der Norm. Bei ARC Solutions wurde dafür eine Bibliothek programmiert, die kompiliert als *remarc.dll* zur Verfügung steht.

Die in *Fortran* geschriebenen Geometriedateien beschreiben den geometrischen Aufbau von Teilen und werden neben den Merkmaldateien und weiteren Daten von der DIN Software GmbH bezogen. Sie enthalten die Methoden der ISO 13584-31, wie in Quelltext 2.2 dargestellt ist.
Die Geometriedateien werden mit Hilfe eines Fortran-Compilers in DLLs (im Weiteren auch „Geometrie-DLLs“ genannt) umgewandelt und in einzelne Normen gegliedert.

```
1  IPN(1)=PNT_CARTESIAN_ABSOLUTE(X,Y,Z,KFIX)
2  DX=-2.0D0*P2
3  DY=0.0D0
4  DZ=0.0D0
5  IPN(2)=PNT_CARTESIAN_RELATIVE(IPN(1),DX,DY,DZ,KFIX)
6  DX=-0.5D0*P3*DTAN(28.0D0*PI/180.0D0)
7  DY=0.5D0*P3
8  DZ=0.0D0
9  IPN(3)=PNT_CARTESIAN_RELATIVE(IPN(1),DX,DY,DZ,KFIX)
10 DX=-P2
11 DY=0.5D0*P1
12 DZ=0.0D0
13 IPN(4)=PNT_CARTESIAN_RELATIVE(IPN(1),DX,DY,DZ,KFIX)
```

```
14  IGR(1)=CREATE_GRP()
15  ILN(1)=LIN_2_PNT(IPN(1),IPN(3),KFIX)
16  ILN(2)=LIN_2_PNT(IPN(3),IPN(4),KFIX)
17  ILN(3)=LIN_2_PNT(IPN(2),IPN(2),KFIX)
18  ILN(4)=LIN_2_PNT(IPN(2),IPN(1),KFIX)
19  CALLCLOSE_GRP()
20  N=1
21  ICTR(1)=CTR_GEN(N,IGR(1),KFIX)
22  N=0
23  IAPS(1)=APS_GEN(ICTR(1),N,NULL,KFIX)
24  X=1.0D0
25  Y=0.0D0
26  Z=0.0D0
27  IDIR(1)=DIR_COMPONENT(X,Y,Z,KFIX)
28  IACHS1=A1P_GEN(IPN(1),IDIR(1),KFIX)
29  ANG1=360.0D0
30  KORPER=SLD_REVOLUTION(IAPS(1),ANG1,IACHS1,KFIX)
```

Quelltext 2.2: Auszug aus einer Geometriedatei. Methoden der ISO 13584-31 befinden sich z. B. in den Zeilen 1, 14, 15, 19 und 30.

Die Geometriedateien sind nach einer definierten Systematik aufgeteilt, die sich nach dem Schema richtet, mit dem die Geometrien der Teile geordnet sind.
Hat ein Bauteil eine individuelle und parametrische Geometrie, so wird es als Komplettteil bezeichnet und durch eine von der DIN festgelegten Identifikationsnummer bestimmt. Hierbei können mehrere Normen dasselbe Komplettteil verwenden. Andererseits kann eine Norm aber auch aus mehreren Komplettteilen bestehen. Welches Komplettteil in diesem Falle für die Geometrieerzeugung verwendet wird, ist in den Merkmaldateien durch Unterscheidung von Merkmalen definiert.
Jedes Komplettteil besteht aus mindestens einem Geometriebaustein, wobei ein Geometriebaustein einen ganz bestimmten, oft einfachen Volumenkörper darstellt, z. B. ein Quader, Zylinder, Kugelkuppe usw.
Bei den Geometriedateien ist jedem Komplettteil eine Hauptdatei (Hauptroutine)

zugeordnet. Diese dient zur Überprüfung der übergebenen Parameter auf Fehler und zur Unterscheidung von 2D- und 3D-Ansichten. Je nach Ansicht gibt es eine entsprechende Hauptroutine, in der es Verweise auf andere Unterroutinen geben kann. Die Trennung in mehrere Routinen bzw. Dateien hat den Vorteil, dass eine bestimmte, eindeutige Geometrie nur einmal erstellt werden braucht. Die Geometriedateien für andere Normen müssen nun nicht komplett neu geschrieben werden, sondern können teilweise oder vollständig auf vorhandene Routinen referenzieren.
Diese Einteilung ist von der DIN Software GmbH vorgegeben, um viel Redundanz zu vermeiden. Dabei wurde ein Kompromiss zwischen Anzahl der Dateien und Komplexität der Geometrie gefunden.

Zum Verständnis wird der Aufruf der Routinen beispielhaft anhand der Erstellung einer Schraube der Norm DIN EN ISO 7046-1 für eine 3D-Darstellung erklärt (siehe Abbildung 2.3): Abhängig von der gewählten Instanz wird entweder das Komplettteil T000K276 oder T000K277 erzeugt. Die aufzurufende Datei für ein T000K277-Teil ist die D90691.f, welche für die 3D-Darstellung einen Verweis auf die T000K277.f enthält. Dies ist die Hauptroutine und ruft im Laufe der Abarbeitung zwei Unterroutinen auf.

2.6 Geometrieerzeugungsprogramm

Die *ARCComponentEngine* ist das Programm, welches für die Geometrieerzeugung zuständig ist. Die Aufgabe wurde schon in der Einleitung geschildert, als von einem Programm gesprochen wurde, welches die Beschreibung einliest, interpretiert und die entsprechenden Funktionen der API ausführt. Dazu wird zuerst die d99999.dat eingelesen. Darin sind die Normnummer und die Komplettteil-Nummer enthalten, mit denen das Programm weiß, welche DLL aufgerufen und welches Komplettteil erzeugt werden muss. Mit dem Abarbeiten der aufgerufenen Geometrie-DLL wird eine Liste mit Elementen erstellt, die zum Schluss abgearbeitet und dadurch das Bauteil - mit Hilfe der NX-API - erstellt und abgespeichert wird.

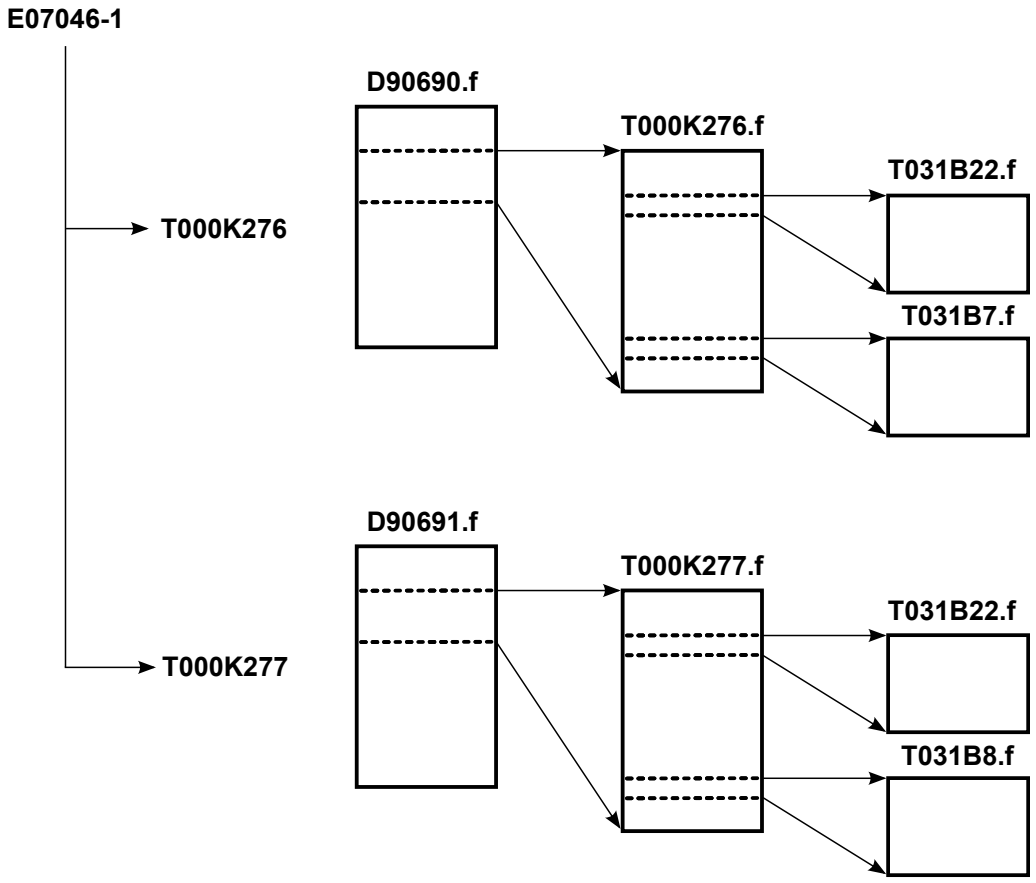

Abbildung 2.3: Aufruf der Fortran-Routinen für die Erzeugung einer Schraube der Norm DIN EN ISO 7046-1. Die Fortrandateien rufen wiederum andere auf. Die Datei „T031B22.f“ wird in beiden Komplettteilen (T000K276 und T000K277) verwendet.

2.7 Problemanalyse und Formulierung der Aufgabenstellung

Im Umgang mit der alten Sprache und den Geometriedateien bestehen folgende Probleme:

Datenlieferung: Die Lieferung der Geometriedateien von der DIN Software GmbH wurde eingestellt. Daher werden keine neuen Normdaten mehr geliefert und die bestehenden nicht gepflegt. ARC Solutions muss sich also selbst um die Datenbeschaffung und -pflege kümmern und für diese neue Aufgabe auch Mitarbeiter

qualifizieren. An Stelle der Ausbildung in Fortran und die ISO 13584-31 kann das Kennenlernen der neuen Sprache und die Einarbeitung in diese stehen.

Geometriedateien: Der Umgang mit den Geometriedateien ist zu unflexibel.

- Da die Geometriedateien in der Programmiersprache Fortran geschrieben sind, muss der Quelltext bei jeder Änderung neu kompiliert werden. Das bedeutet vor allem beim Auftreten von Fehlern einen zusätzlichen Aufwand, da vor jeder Kompilierung eine Prüfung auf fehlerfreie Ausführung erforderlich ist.
- Die Übergabereihenfolge der Parameter muss immer gleich sein. Eine Verbesserung dieses Punktes ist optional, stellt allerdings eine Vereinfachung dar.

ISO 13584-31: Der Standard ISO 13584-31 ist technisch veraltet. Nach [pli] wurde die Initiative zur Erarbeitung der Norm 1990 in Gang gesetzt und ging 1994 / 1995 zu Ende. Zwar wurde seitdem eine weitere Ausgabe veröffentlicht, der Inhalt hat sich jedoch kaum verändert. In den letzten 16 Jahren haben sich die CAD-Systeme stark weiter entwickelt und viele neue Features sind hinzugekommen. Zur Erklärung sollen hier nur ein paar ausgewählte Sachverhalte genannt werden:

- Grundsätzlich ist der Satz an Methoden zwar ausreichend, um sämtliche Normteil-Geometrien abzubilden, jedoch fehlen Definitionen für Anreicherungen, durch die sich die Erweiterungen der Bauteile um Eigenschaften (Metadaten), Berechnungen und Features darstellen lassen. Zu den Eigenschaften gehören z. B. Farbangaben, Werkstoff und Layereinstellungen; Features sind beispielsweise Gewinde und Ankerpunkte. Eine genaue Einteilung ist im Anhang A dargestellt. Diese Anreicherungen werden auch von vielen Kunden gewünscht. Um diesem Interesse nachzukommen wurden *model-Dateien* angelegt, welche normspezifische Anreicherungen beinhalten und mit einem Text-Editor bearbeitet werden. Allerdings ist die Erstellung und die Verarbeitung sehr umständlich und führt immer wieder zu Problemen, deren Lösung zeitaufwändig ist. Eine neue Sprache, welche Anreicherungen einschließt, würde an dieser Stelle eine enorme Verbesserung bedeuten.

- Es gibt inzwischen Features, die in der Norm fehlen, wie Fase, Rundung, Bohrung usw. Zwar sind Fasen und Rundungen in den Geometriedateien enthalten, aber sie sind als 2D-Elemente definiert und werden heutzutage nicht mehr in die Kontur eingebaut, sondern sind eigenständige Features.

- Linien, Kreise, usw. liegen nicht mehr im Raum, sondern werden auf Skizzen platziert.

- Skizzen können inzwischen entlang vorgegebener Kurven extrudiert werden.

- Bei der Konstruktion wird sehr oft mit Hilfselementen gearbeitet. Dadurch könnten verschiedene Methoden der ISO 13584-31 wegfallen, die durch die neuen Funktionen nicht mehr benötigt werden.

- Die Technik zum Beschreiben von Körpern ist überholt. Nach aktuellem Stand der Technik wird nicht mehr mit CSG[3] sondern featurebasiert [Sen94] modelliert. Zum Beispiel wird bei einer Schraube zuerst der Körper und danach der Kopf erstellt. Mit der CSG-Technik gibt es nun eine zusätzliche Operation, die den Kopf vom Körper abzieht. Bei der feature-basierten Variante wird der Kopf direkt vom Körper abgezogen.

Einsatz von REMARC unter Linux: Für den Einsatz von REMARC unter Unix-Systemen müssen derzeit alle Quelltexte (gleich welche Programmiersprache) extra kompiliert werden. Durch Umstellung auf eine vom Betriebssystem unabhängige Sprache würde dieser Aufwand wegfallen.

Ersatz der ARCComponentEngine: Für die Erstellung von Teilen für NX wird derzeit die ARCComponentEngine verwendet. Dieses Programm wurde vor mehr als 15 Jahren mit C programmiert und ist sehr unübersichtlich. Daher wäre der Aufwand viel zu groß, das Programm umzuschreiben, damit es für die neue Sprache eingesetzt werden kann. Darüber hinaus sind die alte und die neue Sprache zu verschieden, sodass sich eine Anpassung nicht lohnt und die Programmierung eines neuen Interpreters weniger Aufwand bedeutet.

[3] Constructive Solid Geometry (CSG) ist eine Art des Modellierens von Teilen durch die Verwendung von Primitiven wie Zylinder, Block, Kegel, usw. und deren Kombinierung mit Booleschen Operationen (Vereinigen, Schnittmenge und Subtrahieren). Siehe dazu auch [Kun02].

Weiterhin gibt es keine Dokumentation zur ARCComponentEngine, sodass die momentane Wartung schwierig ist.

Die Aufgabe der neuen Sprache ist die Ablösung und Erweiterung der alten. Das bedeutet, dass viele Methoden der alten Sprache auf die neue abgebildet werden müssen, damit alle bestehenden Geometriedateien fehlerfrei umgewandelt werden können.

3 Entwicklung einer neuen Sprache

3.1 Anforderungen an die neue Sprache

Umfeld / Aufbau:

- Die neue Sprache muss so ausgelegt sein, um die Form des Bauteils mit einem Satz an Parametern beeinflussen zu können. Dieser stammt aus den Merkmaldateien und ist dort definiert. Damit müssen auch in der neuen Sprache die gleichen Parameter definiert sein. Die Benennung kann dabei vom Namen aus den Merkmaldateien abweichen, wenn gewährleistet wird, dass beide Namen korrekt einander zugeordnet werden.
- Die neue Sprache muss alle Funktionen neutral beschreiben, d. h. unabhängig von einem CAD-System. Dabei muss sich jede Funktion eindeutig von einer anderen abgrenzen. Auch muss jede Funktion in sich eindeutig sein, sodass es keine zwei unterschiedlichen Lösungsmöglichkeiten gibt.
- Die neue Sprache muss erweiterbar sein, damit eine Veränderung oder das Einfügen neuer Funktionen möglich ist.

Inhalt:

- Die neue Sprache muss aufgrund der Ablösung der Geometriedateien Folgendes beinhalten:

  Funktionen zur Veränderung der Geometrie: Zu den Elementen in einer Skizze gehören: Punkt, Linie, Kreis, Bogen, Ellipse, Hyperbel und Parabel sowie die Möglichkeit Elemente zu spiegeln; und zu den Features gehören: Kugel, Kegel, Zylinder, Torus, Block, Extrusion, Rotation sowie die Booleschen Operationen: Vereinigen, Schnittmenge und Subtrahieren.

Verweise auf andere Dateien: Zur Nachbildung der Struktur der Geometriedateien werden Verweise benötigt, damit eine Routine (Datei) eine andere aufrufen kann.

Variablen: Diese dienen zur Vereinfachung - um Berechnungen, deren Ergebnis mehrmals verwendet wird, nur einmal ausführen zu müssen - und zur Übersichtlichkeit.

Bedingungen: Dabei werden Parameter, Variablen oder Berechnungsausdrücke miteinander verglichen. Bei erfüllter Bedingung sollen Variablen Werte zugewiesen werden.

Referenzen: Es muss möglich sein, an bestimmten Stellen auf bestehende Elemente und Features zu verweisen. Zum Beispiel werden bei einer Booleschen Operation zwei bestehende Features verwendet.

Berechnungen: An jeder Stelle, an der ein Wert gefordert ist, muss es möglich sein ganze Zahlen, Dezimalzahlen, definierte Parameter oder Variablen inklusive Vorzeichen anzugeben. Weiterhin müssen Berechnungen mit den vier Grundrechenarten, sowie Wurzel-, Sinus-, Kosinus- und Tangensausdrücke und absolute Beträge enthalten sein. Auch Klammerausdrücke sind eine Bedingung an die Sprache. Die Verwendung von π ist nicht zwingend notwendig, bietet jedoch eine Vereinfachung.

- Die neue Sprache soll sich, wie zu Beginn der Arbeit erwähnt, an aktuelle Modellierungsrichtlinien halten, die durch Befragung der Mitarbeiter zusammengetragen wurden.
  Da mit Skizzen und Features gearbeitet wird, darf es somit keine 2D-Geometrie außerhalb von Skizzen geben. Die Verwendung von Bohrung, Mustern usw., also Features, die nicht in der ISO 13584-31 vorkommen, soll möglich sein. Der Gebrauch von Booleschen Operationen sollte nur in Fällen geschehen, wenn keine andere Lösung möglich ist. Da aber die Geometriedateien darauf aufbauen, muss der Gebrauch fernerhin möglich sein. Und es kommen weitere Funktionen hinzu, um alte Methoden aus der ISO 13584-31 abzulösen. Den Skizzen zugehörig sind Hilfselemente (Hilfslinie, Hilfsbogen, Hilfskreis, Hilfsellipse, Hilfshyperbel) sowie die Möglichkeit, Elemente zu löschen. Zu den Features kommen die Spiegelung von Features, Fase und Rundung hinzu.

- Weiterhin sollen Anreicherungen definiert werden können. Details zu den Anreicherungen sind im Kapitel 2.7 auf Seite 15 erläutert.

Optional:

- Es wäre von Vorteil, wenn für die Umsetzung eine (Programmier-) Sprache verwendet wird, bei der mit jeder Änderung des Quelltextes nicht neu kompiliert werden muss.
- Eine Vereinfachung stellt das Arbeiten mit Skizzenkoordinaten dar. Das bedeutet, dass in Skizzen nicht mit Weltkoordinaten gearbeitet wird, sondern mit lokalen 2D-Koordinaten. Zwar wird in den Geometriedateien auch mit Skizzenkoordinaten gearbeitet, jedoch ist dort eine Eingabe einer dritten Koordinate möglich, was zu Fehlern führen kann.
- Methoden, deren Abbildung mit Hilfe anderer Methoden möglich ist, können zur Vereinfachung weggelassen werden. Zum Beispiel können Polarkoordinaten mit kartesischen Koordinaten berechnet werden.
- Andererseits können zusätzlich zu den grundlegenden Skizzenelementen komplexere Elemente definiert werden, wie z. B. Polygone.
- Falls möglich, sollte die Reihenfolge der Parameter bei der Übergabe beliebig sein. Das würde eine einfachere Verarbeitung der Parameter bewirken. Diese Option hängt mit der internen Verarbeitung der Parameter zusammen, da im Mapping-Editor von REMARC die Spalten beliebig verändert werden können.
- Darüber hinaus kann eine Übergabe von Optionen (Flags) hinzukommen, um verschiedene Abarbeitungen des Teileerstellungsprozesses zu steuern.

Nicht enthalten:

- In der neuen Sprache sollen nicht verwendete und nicht benötigte Methoden und Anweisungen weggelassen werden. Das sind zum Beispiel Schleifen und Gruppen, aber auch Methoden zur Veränderung der Geometrie wie *HYP_GEN*, *WDG_GEN* u. a.
- Die Kontrolle der übergebenen Parameter auf positive Zahlen, wie es in den Geometriedateien stattfindet, soll nicht übernommen werden.

- Die neue Sprache soll folgende Elemente nicht enthalten: Freiformflächen, Splines.

3.2 Aufbau der neuen Sprache

Als Name für die neue Sprache wurde aus verschiedenen Vorschlägen *nf* ausgewählt. Diese Abkürzung steht für *Neutral-Format*. Die Wörter *Neutral* und *Format* gibt es auch im Englischen, sodass zwischen der deutschen und der englischen Bezeichnung nicht unterschieden werden braucht.

Ein Teil (und damit auch eine Datei) beginnt immer mit einer Bezeichnung. Das kann eine Komplettteil-Nummer sein, aber auch eine Unterroutine oder ein sonstiger Name. Wie bei den Geometriedateien ist auch hier eine Grundeinteilung nach Komplettteil-Nummern gedacht. Eine Gliederung nach Normen, wie es derzeit bei den Geometrie-DLLs erfolgt, ist nicht sinnvoll, weil z. T. mehrere Normen auf dasselbe Komplettteil verweisen und das daher zu unnötigen Überschneidungen führen würde.
Danach folgen eine Reihe von Features, deren Reihenfolge nicht vorgegeben ist. Die Features werden unterteilt in volumenbildende und sonstige, was in Abbildung 3.1 dargestellt ist. Zu den volumenbildenden Features gehören Extrusion, Rotation, Block, Boolesche Operation, Spiegel-Feature und Import-Feature. Die restlichen Features bilden Fase, Ebene, Achse, Parameterdefinition, Variable und Bedingung. Jedes Feature, außer der Bedingung, hat einen Namen zur Identifikation.

- Eine **Extrusion** besteht aus einer Skizze, einer Richtung, in die extrudiert werden soll, einem Startwert sowie einem Endwert. Zur Vereinfachung kann die Richtung nicht selbst gewählt werden, sondern ist entweder eine Grundrichtung (x, y oder z) oder eine bestehende Achse. Der Startwert kann auch wegfallen, dann liegt dieser bei Null. Optional ist weiterhin die Angabe einer Booleschen Operation, wobei hier zwischen drei Typen ausgewählt werden kann (siehe Boolesche Operation weiter unten). In diesem Fall muss auch ein Grundkörper gewählt werden. Zur Auswahl stehen dabei nur volumenbildende Features.

▷ Eine *Skizze* beinhaltet einen Namen, eine Lage, sowie eine Liste von Skizzenelementen. Eine Lage ist entweder eine bestehende Ebene oder eine Grundebene (xy, xz oder yz). Optional zu der Grundebene kann die Position einer Fläche angegeben werden. Dabei muss der Abstand angegeben werden, der zwischen dem Nullpunkt (Ursprung) und der gesuchten Fläche in der Richtung liegt, die senkrecht zur Lage der Grundebene ist. Weiterhin kann auch ein bestehendes volumenbildendes Feature gewählt werden, zu dem die gesuchte Fläche gehört, da es vorkommen kann, dass bei der angegebenen Position zwei Flächen von zwei verschiedenen Features liegen.
Der Sinn der Auswahl einer Fläche liegt darin, dass die Skizze somit direkt auf eine bestehende Fläche eines Features platziert werden kann und nicht vorher eine Bezugsebene erstellt werden muss.
Die Erklärung der Skizzenelemente erfolgt weiter unten.

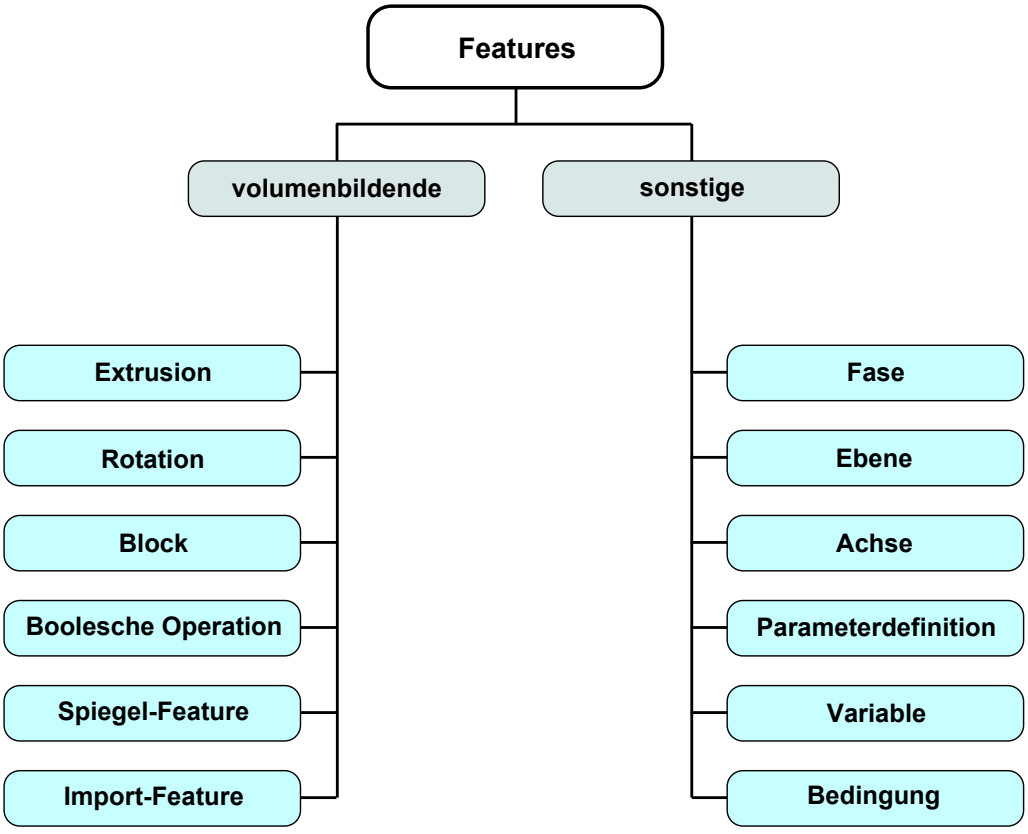

Abbildung 3.1: Einteilung der Features

- Eine **Rotation** umfasst die gleichen Attribute wie die Extrusion. An Stelle des Startwertes muss jedoch ein Startwinkel, und anstatt des Endwertes ein Endwinkel angegeben werden.

- Ein **Block** kann mit Hilfe verschiedener Definitionen entstehen. Als Beispiel wurde ein Block aus zwei Punkten implementiert. Jeder Punkt besteht aus den drei Koordinaten x, y und z. Optional ist auch hier wieder die Angabe einer Booleschen Operation.

- Eine **Boolesche Operation** als Feature setzt sich zusammen aus einem Grundkörper, einem Werkzeug und einem Typ. Grundkörper und Werkzeug sind volumenbildende Features. Als Typ kann entweder Schnittmenge, Subtrahieren oder Vereinigen bestimmt werden.

- Ein **Spiegel-Feature** besteht aus einer Spiegelebene, die entweder eine bestehende oder eine Grundebene ist, und einer Liste von volumenbildenden Features.

- Ein **Import-Feature** enthält einen weiteren Namen für den Aufruf einer anderen nf-Datei sowie eine Liste von Parametern, die der Unterroutine, also der anderen nf-Datei, übergeben werden.

- Bei der **Fase** kann zwischen zwei Varianten gewählt werden: Fase aus zwei Werten oder Fase aus Wert und Winkel. Die *Fase aus zwei Werten* besteht aus einem Namen, zwei Seitenlängen und einem Kantentyp. Die *Fase aus Wert und Winkel* beinhaltet einen Namen, eine Seitenlänge, einen Winkel und einen Kantentyp.
  Es gibt zwar mehrere Kantentypen, doch wurde sich hier auf die wichtigsten zwei beschränkt: *Linear* und *Circular* (Kreisrund). Ist die Kante kreisrund, so muss eine Koordinate (entweder x, y oder z) ausgewählt und der Abstand zwischen Nullpunkt (Ursprung) und der Kante angegeben werden. Bei einer linearen Kante müssen zwei Koordinaten und dazugehörige Abstände angeführt werden. Optional kann am Schluss eine ganzzahlige positive Nummer angegebenen werden. Der Grund ist, dass zwei oder mehr Kanten auf derselben Ebene bzw. Achse liegen können. Daher dient diese Zahl als Unterscheidungsmöglichkeit. Wird die Zahl nicht angegeben, so wird stets die erste Kante verwendet,

auf die die angegebenen Attribute zutreffen.

- Eine **Ebene** besitzt einen Namen und ist entweder eine parallele, gedrehte oder eine Grundebene. Eine *Grundebene* liegt entweder auf xy, xz oder yz, welche die Ebenentypen darstellen. Eine *parallele Ebene* referenziert auf eine bestehende Ebene oder einen Ebenentyp und hat einen Abstand zum Nullpunkt. Eine *gedrehte Ebene* referenziert auch auf eine bestehende Ebene oder einen Ebenentyp, hat einen Winkel und eine Achse, um die die Ebene gedreht wird. Die Achse ist eine bestehende Achse oder eine der Grundachsen.

- Als **Achse** ist nur eine Grundachse implementiert worden. Eine Grundachse setzt sich zusammen aus einem Namen und einer der Richtungen x, y oder z, welche die Achstypen darstellen.

- Eine **Parameterdefinition** enthält einen Namen sowie weitere Zeichen, bei denen Zahlen und Buchstaben als Erläuterung für den Parameter dienen. Der Sinn der Zahlen ist es, die Reihenfolge der Parameter zu zeigen. Die Buchstaben fungieren als Beschreibung des Parameters. Weil die Übergabe der Parameter unabhängig von der Reihenfolge erfolgen soll, ist die Angabe der Zeichen optional und dient nur als Hilfestellung.

- Eine **Variable** besitzt einen Namen sowie eine Berechnung, die jedoch nicht angegeben werden muss, da auch bei einer Bedingung der Variable ein Wert zugeordnet werden kann.

- Der Aufbau einer **Bedingung** hat folgende Form: Nach dem Schlüsselwort *if* steht eine Bedingung und mindestens eine Anweisung. Danach kann entweder ein *else if* - mit einer Bedingung und mindestens einer Anweisung - oder ein *else* - ohne Bedingung - stehen. Nach einem *else if* kann wieder ein *else if* oder ein *else* kommen.

Ein Skizzenelement ist entweder ein geometrisches Element, eine Skizzenbeziehung oder das Löschen von mindestens einem geometrischen Element. Die Einteilung ist in Abbildung 3.2 dargestellt.

- Ein geometrisches Element hat einen Namen und ist entweder ein Punkt, eine Linie, ein Kreis, ein Bogen oder eine Spiegelung.

- Ein **Punkt** ist entweder absolut, relativ oder entsteht aus einem Schnittpunkt von zwei bestimmten Elementen. Darüber hinaus gibt es die Option, den Punkt im CAD-System anzeigen zu lassen.
  ▷ Ein *absoluter Punkt* besteht aus einem x- und einem y-Wert.
  ▷ Ein *relativer Punkt* besitzt für x und für y einen Bezugspunkt und eine Wertangabe.
  ▷ Bei einem *Punkt aus einem Schnittpunkt von zwei Elementen* gibt es einmal den Schnittpunkt aus zwei Linien und den Schnittpunkt aus einem Kreis oder einem Bogen und einer Linie.
- Eine **Linie** beinhaltet Referenzen auf zwei Punkte sowie die Optionen eine Hilfslinie zu erzeugen und die Linie im CAD-System nicht anzuzeigen.

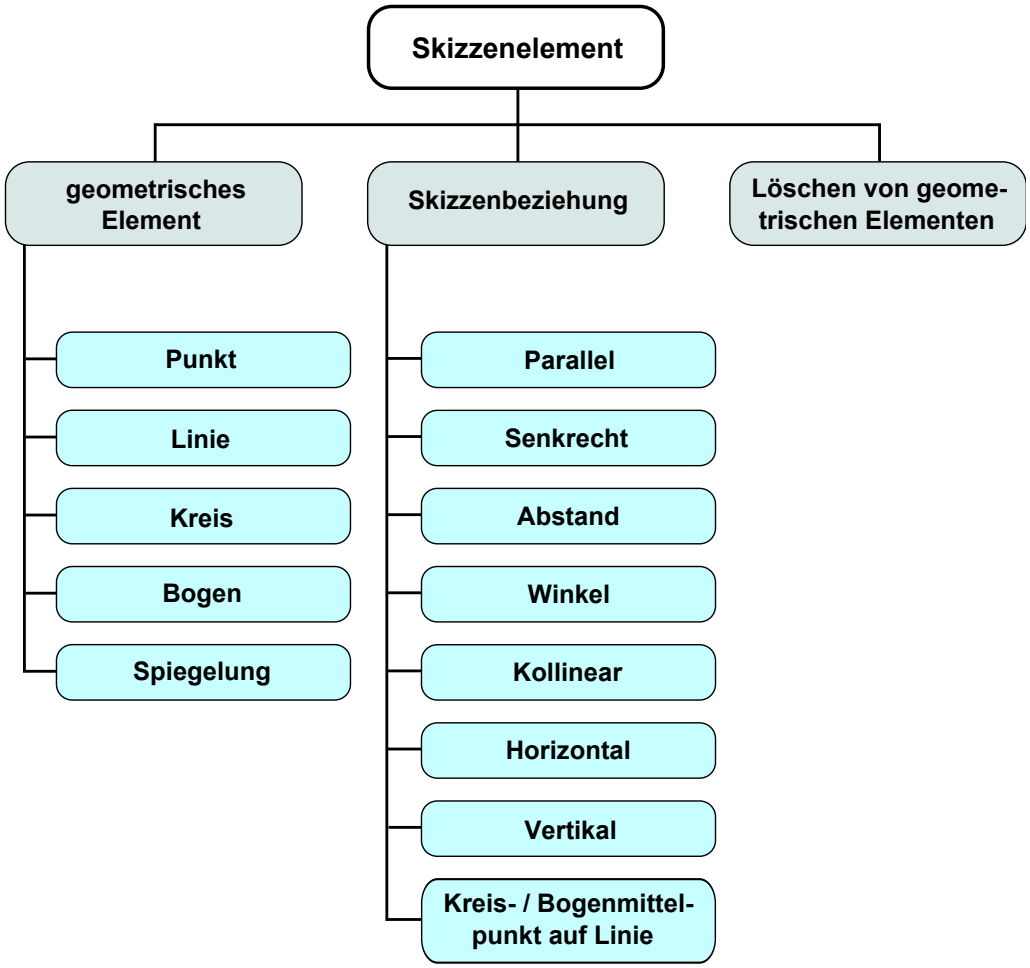

Abbildung 3.2: Einteilung der Skizzenelemente

  - Ein **Kreis** wird durch einen referenzierten Punkt, dem Mittelpunkt, und einem Radius angegeben. Auch hier gibt es die Optionen einen Hilfskreis zu erzeugen und den Kreis im CAD-System nicht darzustellen.

  - Ein **Bogen** wird entweder aus einem Mittelpunkt mit Radius oder aus drei Punkten erzeugt. Die Angaben Hilfsbogen und den Bogen nicht im CAD-System anzuzeigen sind optional.
    ▷ Ein *Bogen aus einem (referenzierten) Mittelpunkt mit Radius* weist dazu einen Start- und einen Endwinkel auf.
    ▷ Ein *Bogen aus drei Punkten* wird aus drei referenzierten Punkten erstellt: einem Startpunkt, einem Punkt auf dem Bogen und einem Endpunkt.

  - Eine **Spiegelung** beinhaltet eine Spiegelachse, sowie mindestens ein Element, das gespiegelt werden soll, und sie gibt die gespiegelten Elemente zurück. Die Spiegelachse ist entweder eine Grundebene oder eine bestehende Achse, Ebene oder Linie.

- Bei den Skizzenbeziehungen gibt es folgende Möglichkeiten: Parallel, Senkrecht, Abstand, Winkel, Kollinear, Horizontal, Vertikal und Kreis- / Bogenmittelpunkt auf eine Linie.

  - **Parallel** oder **Senkrecht** gesetzt werden zwei bestehende Linien.

  - Zu einem **Abstand** oder **Winkel** gehören zwei referenzierte Linien und der gewünschte Abstand bzw. Winkel.

  - **Kollinear** beinhaltet entweder zwei Kreise bzw. Bögen oder zwei bestehende Achsen bzw. Linien oder Grundachsen.

  - **Horizontal** oder **Vertikal** platziert wird eine referenzierte Linie.

  - **Kreis- / Bogenmittelpunkt auf eine Linie** enthält einen bestehenden Kreis oder Bogen und eine bestehende Achse bzw. Linie oder eine Grundachse.

- Die letzte Möglichkeit eines Skizzenelements ist das **Löschen von geometrischen Elementen**, wobei mindestens ein Element angegeben werden muss.

An jeder Stelle, an der ein Wert eingegeben werden muss, ist die Auswahl zwischen einer ganzen Zahl, Dezimalzahl, Variable, einem definierten Parameter, der Zahl π oder einem Klammerausdruck zu treffen. Daher stehen alle Angaben inklusive eines optionalen negativen Vorzeichens in einer Ebene. Ein Klammerausdruck ist hierbei entweder ein absoluter Betrag, eine Wurzel-, Sinus-, Kosinus- bzw. Tangensberechnung oder eine einfache Klammer („(...)“). Damit die korrekte Reihenfolge bei Punkt- und Strichrechnung gewährleistet wird, besteht eine Punktrechnung aus einem linken und einem rechten Ausdruck und einem *Mal-* oder *Durch*-Zeichen. Eine Strichrechnung enthält ein *Plus-* oder *Minus*-Zeichen und links und rechts jeweils eine Punktrechnung. Dieses Ergebnis ist entweder abgeschlossen oder wird wiederum in einem Klammerausdruck verwendet. Ferner kann eine Punkt- und Strichrechnung mehrmals hintereinander auftreten.

3.3 Vergleich mit Fortran und ISO 13584-31

Im Vergleich zu Fortran und der ISO 13584-31 ergeben sich einige Unterschiede, wovon die meisten in Tabelle 3.1 dargestellt sind. Ein Teil der Unterschiede wird im Folgenden erklärt.

- In den Geometriedateien sind die Ansichten für 2D- und 3D-Darstellungen enthalten. Da die meisten der geläufigen CAD-Systeme 3D-Modellierung unterstützen und fast niemand mehr in 2D konstruiert, enthält die neue Sprache nur 3D-Konstruktionselemente.
- Einige Methoden der ISO 13584-31 sind unverständlich. Das heißt, dass für das Nachvollziehen des Inhalts einer Geometriedatei die Bedeutung und Anwendung der Methode in der Norm nachgeschlagen werden muss.
- Umständlich ist in den Geometriedateien, dass die Zuweisungen der Variablen immer vor deren Verwendung in den Methoden erfolgt. Das stört, weil manchmal eine ganze Quelltextseite oder mehr zwischen der Zuweisung und der Verwendung liegt. Im Neutral-Format werden in den jeweiligen Funktionen Zahlen direkt verwendet. Und Variablen werden anders genutzt: sie dienen hier zur Berechnung von Werten, die nicht übergeben werden.

- Bedingt durch den Aufbau und die Definition der ISO 13584-31 enthalten die Geometriedateien Schleifen sowie Gruppen, welche die Nachvollziehbarkeit bzw. die Verständlichkeit erschweren. Das betrifft auch weitere Methoden, die bei einem anderen Aufbau der Sprache dann nicht mehr erforderlich sind.

Tabelle 3.1: Vergleich zwischen Fortran mit der ISO 13584-31 und dem Neutral-Format

Fortran und ISO 13584-31	**Neutral-Format**
2D- und 3D-Darstellung	nur 3D-Darstellung
Reihenfolge der Übergabeparameter vorgegeben; Prüfung auf positiven Wert	Reihenfolge ist beliebig; keine Prüfung
keine Leerzeichen möglich	Verwendung und Anzahl der Leerzeichen und Tabs bis auf die Syntax nicht vorgeschrieben
Winkelangaben teilweise in Radiant	alle Winkelangaben in Grad
Länge der Methoden auf 31 Zeichen begrenzt; (auch dadurch) Funktion z. T. unverständlich	keine Begrenzung der Länge; Funktion eindeutig verständlich
Verwendung von Gruppen und Schleifen (*for*)	Wegfall nicht notwendiger Methoden der ISO; neue Funktionen
Geometrie liegt im Raum (keine Skizzen vorhanden)	Geometrie liegt nur in Skizzen
Punkte mit 3 Koordinaten (x,y,z)	Punkte mit 2 Koordinaten (x,y)
Berechnung aller notwendigen Koordinaten	Verwendung von Hilfselementen
keine Beziehungen	Skizzenelemente mit Beziehungen
Fase in Kontur enthalten	Fase als Feature
nur Boolesche Operationen	Features beinhalten Boolesche Operationen
keine Anreicherungen definiert	Anreicherungen bei Erweiterung der Sprache möglich
Zuweisungen der Variablen vor deren Verwendung in Methoden	Verwendung direkt bei jeder Funktion
Auflistung aller verwendeten Variablen zu Beginn der Routine	Ort der Auflistung der Variablen und Parameter nicht vorgegeben
Vergleiche in Fortran-Syntax („.LT.“, „.EQ.“, ...)	gewohnte Schreibweise (<, <=, !=, ...)
Dezimalzahlen enden mit „D0“ und müssen Kommastelle enthalten	Zahlen können mit / ohne Kommastelle sein

4 Umsetzung

Für die Implementierung der neuen Sprache gibt es grundsätzlich zwei verschiedene Möglichkeiten.
Eine Variante ist die Verwendung einer Programmiersprache und eventuell einer Programmierumgebung, welche die Effizienz erheblich steigert. Weiterhin würde mit der Wahl einer Programmiersprache, mit der die API direkt angesprochen werden kann, keine Zwischenstufe benötigt werden. Ein Nachteil ist, dass der Quelltext bei jeder Änderung neu kompiliert werden muss, was mehr Aufwand bedeutet, da für die Prüfung ein zusätzlicher Schritt - die Kompilierung - notwendig ist. Die Alternative ist die Verwendung einer Skriptsprache. Dabei ist zu beachten, dass unter Umständen eine Verbindung zwischen der Skriptsprache und der API-Programmiersprache erstellt werden muss.
Die andere Möglichkeit ist die Erstellung einer eigenen Sprache, die mit der Programmierung eines Interpreters einhergeht. Für die Umsetzung bietet sich eine Auszeichnungssprache wie XML an. Für das Erstellen von Modellen gibt es viele verfügbare Hilfsmittel (Custom UI Editor, Visual Studio, Editix, Morphon uvm.).

Der größte Vorteil einer eigenen Sprache ist die Vorgabe aller erlaubten Anweisungen, womit ein Großteil der Fehler verhindert werden kann. Das führte zu der Entscheidung, eine eigene Sprache zu entwickeln und zur Suche nach dafür geeigneten Werkzeugen. Ein Werkzeug, welches einen Teil der Entwicklung abnimmt, unterstützt und beschleunigt dabei die Umsetzung.

4.1 Java und Eclipse

Für das Einlesen und Interpretieren einer nf-Datei und das Ausführen der entsprechenden Funktionen der API, muss eine Programmiersprache verwendet werden.

Dabei stehen je nach CAD-System meist mehrere zur Auswahl. Tabelle 4.1 zeigt einen Überblick über die möglichen Programmiersprachen der APIs der von ARC Solutions am Häufigsten genutzten CAD-Systeme *NX* und *Pro/E*. Die Verwendung einer Sprache, die von beiden CAD-System-APIs unterstützt wird, bedeutet am wenigsten Aufwand. Somit bleibt eine Auswahl zwischen C++, Visual Basic und Java, wobei letztere die im Unternehmen am Häufigsten eingesetzte Sprache ist. Die Benutzung von Java hat zwei Vorteile. Zum einen ist es plattformunabhängig und zum anderen wurde REMARC damit programmiert, was eine Anbindung an dieses Programm vereinfachen würde. Somit wurde sich für Java entschieden.
Für die effiziente Programmierung in Java ist es sinnvoll, eine Programmierumgebung zu verwenden. Als Beispiele seien hier Eclipse, NetBeans, JBuilder, JCreator und Java-Editor genannt. Die Entscheidung fiel auf Eclipse, weil die Implementierung von REMARC darauf basiert und dadurch eine spätere Eingliederung des Projekts wenig Anpassung bedeutet. Sehr vorteilhaft ist die ständige Weiterentwicklung von Eclipse, was auch den umfangreichen Bereich der Teilprojekte beinhaltet.

Tabelle 4.1: Überblick über die API-Programmiersprachen der CAD-Systeme Pro/E und NX

Pro/E		Java	C++		VB	
NX	C#	Java	C++	C++.NET	VB	VB.NET

4.2 Eclipse Modeling Framework

Eines der vielen Teilprojekte ist das Eclipse Modeling Framework (EMF). EMF ist ein Framework für die automatisierte Generierung von Quelltext auf Basis von Modellen[4]. Diese Modelle werden mit Ecore gebildet - eine Metasprache zur Definition objektorientierter Datenstrukturen. [Sch] Quelle und Ziel eines solchen Ecore-Modells kann ein UML-Klassendiagramm, eine XML-Schemadefinition oder ein Gerüst von Java-Interfaces und -Klassen sein, was Abbildung 4.1 verdeutlicht.

[4] Genau genommen ist EMF ein Framework zur Entwicklung von Werkzeugen und Applikationen, die auf einem strukturierten Datenmodell basieren, mit Quelltexterzeugung. Das EMF-Ecore-Modell ist eine Meta-Modell, mit dem eigene domänenspezifische Sprachen (DSLs) definiert werden können. Der Begriff DSL wird meist in der Informatik verwendet.

[GME] Es kann allerdings auch mit Hilfe des „Ecore Model Editors“ in einer Baumstruktur selbst erstellt werden. In dem Ecore-Modell werden charakteristische Merkmale einer objektorientierten Programmiersprache, wie Klassen, Methoden, Attribute, Interfaces, Vererbung und Beziehungen, abgebildet.
Für die Generierung von Java-Quelltext wird aus dem Ecore-Modell ein Generator-Modell (*genmodel*) erzeugt, was Konfigurationsinformationen beinhaltet. Aus dem Generator-Modell wird zuerst das EMF.Model-Plug-in generiert, was u. a. den Java-Quelltext für alle Modellelemente und eine Fabrikklasse zur Erzeugung von Exemplaren der Modellobjekte (Factory) beinhaltet. Weiterhin kann ein EMF.Editor-Plug-in generiert werden, was auf das EMF.Model-Plug-in aufbaut. Es umfasst u. a. den Modelleditor und die Registrierung der Dateiendung. Mit dem Modelleditor kann nun ein eigenes konkretes Objekt erstellt werden, was später mit Hilfe des Java-Quelltextes ausgelesen wird.

[Vog], [SBPM09]

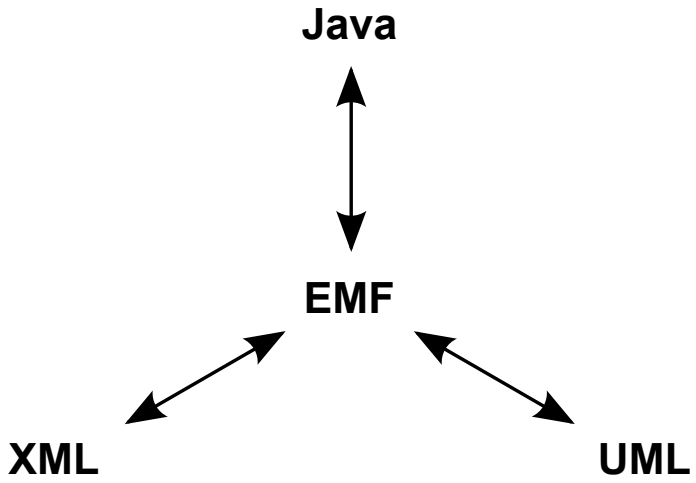

Abbildung 4.1: EMF vereinigt die Technologien Java, XML und UML

EMF ist ein Werkzeug, mit dem die neue Sprache umgesetzt werden kann. Mit Hilfe des Modelleditors können syntaktisch korrekte Objekte erstellt werden. Und es können nur definierte Attribute zugewiesen werden. Allerdings können optionale Attribute nicht erkannt werden - sie sind zusammen mit den Pflicht-Attributen zu sehen. Der Modelleditor ist einfach aufgebaut und die Elemente werden übersichtlich angezeigt, wie Abbildung 4.2 zeigt.
Unklar ist, wie Berechnungen sinnvoll dargestellt werden können. Einerseits können

sie in der Art des gesamten Modells aufgebaut werden, was bei einer umfangreichen Berechnung zu einer stark verzweigten und unübersichtlichen Struktur führt. Andererseits kann die Eingabe einer Berechnung in einem Attribut erfolgen, wobei dann keine Kontrolle auf Richtigkeit möglich ist.
Durch die Vielseitigkeit von EMF und die daraus resultierende Komplexität sinkt die einfache Handhabung und führt zu uneffizientem Arbeiten, was sich in vielen Bereichen bei der Benutzung bemerkbar macht. Auch entstehen dadurch Fehler, was u. U. im Löschen und Neuanlegen des Projektes (wobei das Ecore-Modell natürlich übernommen wird) endet. Zum Beispiel müssen vor dem Löschen einer Klasse alle Abhängigkeiten aufgehoben werden und danach alle Einträge im Java-Quelltext von Hand entfernt werden.
EMF ist zwar für die Implementierung geeignet, dennoch führten die vielen Schwierigkeiten zur Suche nach einer Erweiterung.

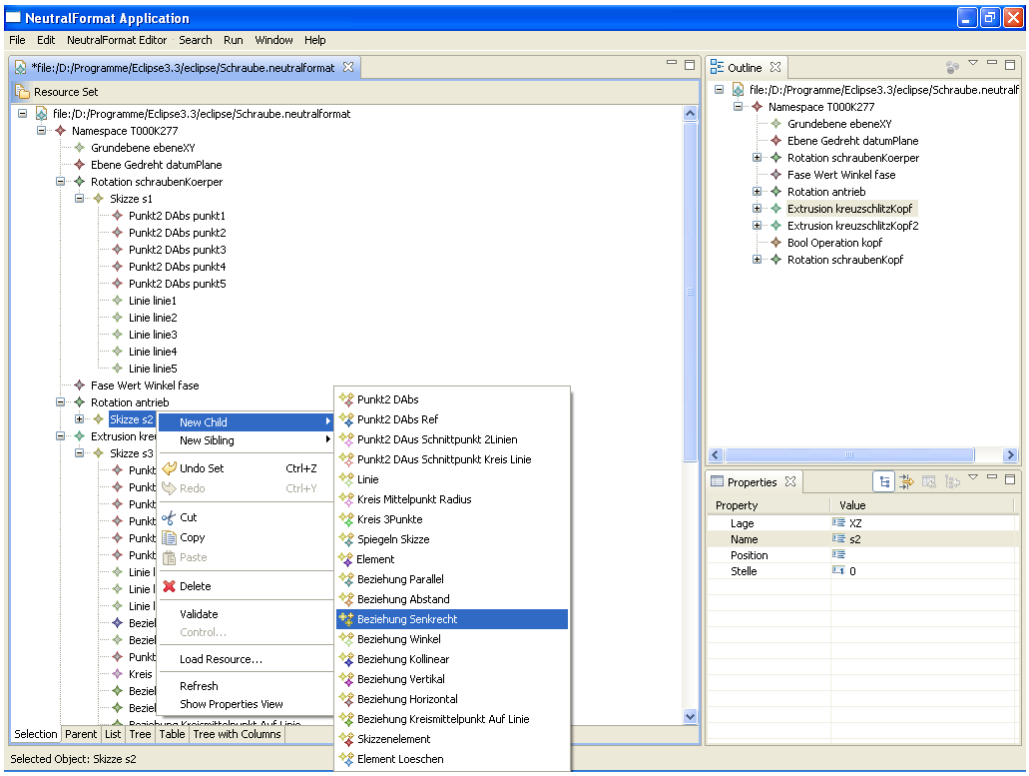

Abbildung 4.2: EMF-Modelleditor, in dem ein Objekt bearbeitet wird. Über das Kontextmenü können neue Elemente bzw. Features hinzugefügt werden.

4.3 Xtext

Xtext ist ein weiteres Teilprojekt und Framework für Eclipse, eng mit EMF integriert und speziell dafür geschaffen worden, um textuelle DSLs zu entwickeln. Die Beschreibung der Sprache wird hier *Grammatik* genannt. Die Grammatik hat, ähnlich wie bei EMF, Merkmale einer objektorientierten Programmiersprache (Attribute, Ableitung, Beziehungen, usw.). Anhand dieser Grammatik erzeugt der Generator von Xtext alle zur Anwendung der DSL benötigten Werkzeuge: ein Ecore-Modell (mit dem der Java-Quelltext erzeugt wird), Parser, Linker, Validator, einen Texteditor mit Syntaxhervorhebung, Auto-Vervollständigung und Navigation, uvm. Ein Teil davon wird durch das Werkzeug ANTLR erstellt, das im nächsten Kapitel erläutert wird. Die Bereitstellung der Werkzeuge bedeutet, dass sich allein auf die Grammatik konzentriert werden braucht.
Eine praktische Einführung mit ausführlichem Beispiel ist unter [KZ09a], [KZ09b] und [KZ10] zu finden.

[Ecl]

Ein großer Vorteil ist die einfache Handhabung des Frameworks, was aus der Tatsache hervorgeht, dass „Xtext von vornherein auf Erweiterbarkeit hin gebaut [ist], sodass sich Anpassungen fast überall sehr einfach und elegant vornehmen lassen“ [JAX].
Darüber hinaus gibt es weitere komfortable Vorteile:

- Die nf-Dateien werden zwar nicht in XML-Form gespeichert, sondern als reine Textdateien. Das stellt jedoch aufgrund des Parsers, der die Dateien einliest, kein Problem dar. Doch aufgrund der engen Integration mit EMF kann jede nf-Datei auch in XML-Form umgewandelt und gespeichert werden.

- Durch die Verwendung einer Grammatik können die Berechnungen einfach dargestellt werden.

- Beim Erstellen der nf-Dateien kontrollieren Parser und Validator die Eingaben und zeigen Fehler sofort an. Dabei ist es egal, ob es sich um die Reihenfolge bzw. Schreibweise der Attribute und Schlüsselwörter oder um anzugeben-

de Werte (Berechnungen, Verweise) handelt. Zum Teil werden Fehler sogar verständlich erklärt.

- Die Auto-Vervollständigung im Texteditor gibt an jeder Stelle eine Auswahlmöglichkeit von gültigen Schlüsselwörtern an, wie in Abbildung 4.3 dargestellt ist.
- Bei der automatischen Generierung des Quelltextes ist kein manueller Eingriff nötig.

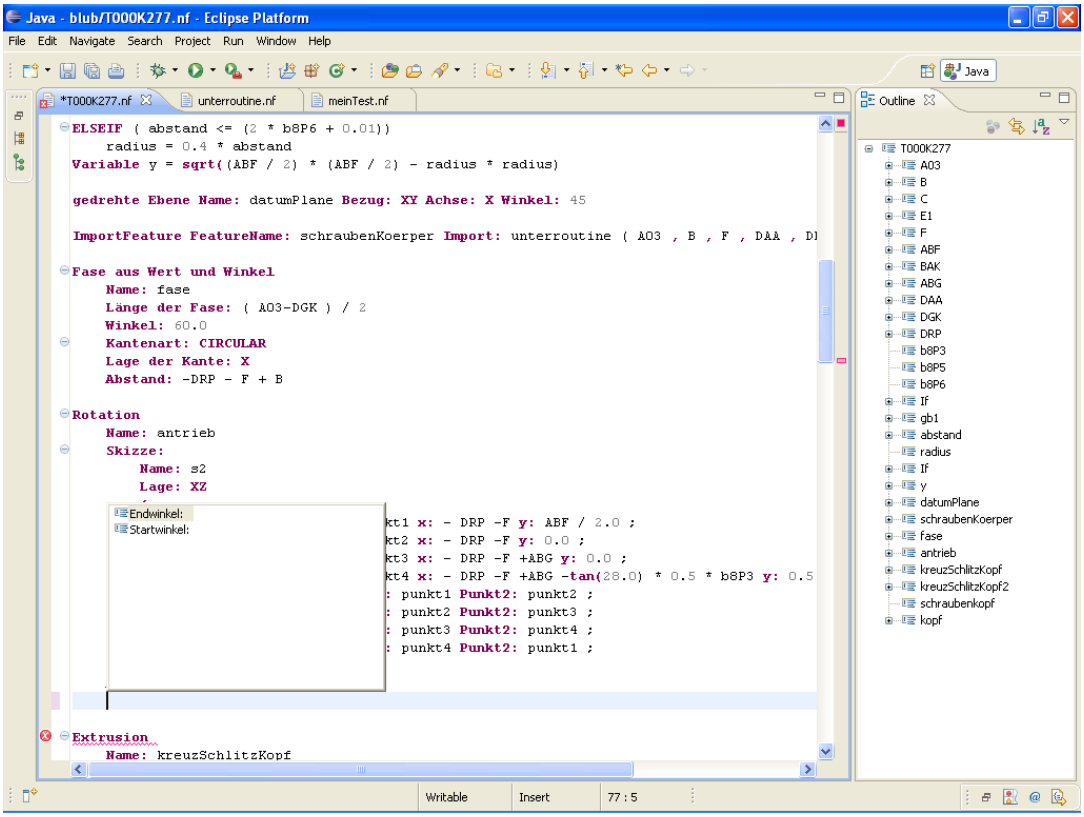

Abbildung 4.3: Texteditor von Xtext zur Erstellung von Objekten. Lila dargestellt sind Schlüsselwörter des Neutral-Formates. Mit Hilfe der Auto-Vervollständigung werden an jeder Stelle eine Liste von gültigen Schlüsselwörtern angezeigt (unten links zu sehen).

Mit Xtext können die im Kapitel 3.1 genannten Anforderungen umgesetzt werden. Weiterhin werden die im vorigen Kapitel aufgezeigten Schwierigkeiten durch die indirekte Benutzung von EMF umgangen. Letztendlich wurde die neue Sprache mit Xtext erfolgreich umgesetzt.

Die Benutzung von Xtext umschließt jedoch folgende Einschränkungen:

- Das Ecore-Modell kann nicht vollständig in einer Xtext-Grammatik abgebildet werden: Klassen, von denen abgeleitet wird, können keine Attribute enthalten. Das kann gelöst werden und verursacht lediglich eine wenig aufwändigere Erstellung der Grammatik.
- Xtext ist erst seit vier Jahren in Entwicklung und seit zwei Jahren in Eclipse als Projekt eingebunden. Kleine Schwächen zeigen sich an verschiedenen Stellen und betreffen lediglich die Oberfläche und nicht die Funktionalität, sodass sie in Kauf genommen werden können. Zum Beispiel werden nach der Berichtigung von Fehlern in der Xtext-Grammatik diese solange angezeigt, bis die Datei geschlossen und neu geöffnet wird.

4.4 ANTLR

Mit dem Werkzeug ANTLR (ANother Tool for Language Recognition) können Texte eingelesen und interpretiert werden. Für diese Aufgabe muss eine Grammatik definiert werden, aus der automatisch ein Lexer, Parser und Baum-Parser erzeugt wird. Diese Grammatik wird mit der von ANTLR bereitgestellten Sprache angelegt und ist hier dieselbe, die mit Xtext erstellt wurde. Ein Lexer (auch Scanner genannt) liest die Eingabedaten (in diesem Fall eine nf-Datei) ein und zerlegt sie in einzelne Tokens - Wörter, die der Parser versteht. Unterschieden wird dabei zwischen Schlüsselwörtern, Variablen und Operatoren. Als Nächstes liest der Parser diese Tokens ein, überprüft sie auf Richtigkeit und erstellt daraus eine hierarchische Struktur. ANTLR liefert auch entsprechenden Quelltext, mit dem diese Struktur ausgelesen werden kann. Der Quelltext kann in verschiedenen Sprachen erstellt werden, wobei für die Anwendung in Xtext Java verwendet wird.
Für tieferes Verständnis siehe [Hof].

Es ist erkennbar, dass ANTLR die Technik ist, die bei der Erstellung der Xtext-Grammatik, beim Einlesen der nf-Dateien und bei der Kontrolle im Xtext-Editor verwendet wird.

5 Werkzeuge zur Anwendung

5.1 Initialisierung der Übergabewerte

Vor dem Aufruf des Hauptprogramms ist ein Zwischenschritt, in dem die Übergabewerte festgelegt werden, notwendig. Dabei sind folgende Punkte enthalten:

- Initialisierung der Übergabeparameter entsprechend der gewünschten Norm.
- Definition der Pfade (z. B. für die nf-Datei und den Ausgabepfad).
- Festlegen von Optionen (Einstellungen).

Dieser Zwischenschritt ist lediglich für das Testen der Werkzeuge gedacht. Die Geometrieerstellung über das Neutral-Format soll zukünftig fest in REMARC verankert werden. Dort können die Parameter und die Pfade einfach abgefragt werden.

5.2 Steuerung der Geometrieerzeugung

Das Hauptprogramm bekommt die Parameter, Pfade und Optionen übergeben und hat folgende Grundaufgaben:

1. Erzeugen einer Session - Die Session ist die Grundlage zum Arbeiten mit einem CAD-System.
2. Öffnen eines Templates - Ein Template ist eine native und oft firmen- bzw. anwendungsspezifische Vorlage, die generische Information beinhaltet. Die Benutzung eines Templates ist jedoch nicht zwingend notwendig.
3. Erzeugung der Geometrie
4. Speicherung des Teils

Für die Erzeugung der Geometrie wird zuerst eine nf-Datei eingelesen. Dabei wird über die enthaltenen Features iteriert und die entsprechende Funktion zur Geometrieerstellung aufgerufen. Analog dazu wird bei einer Skizze mit den Skizzenelementen verfahren.

Es gibt vier Maps, die jeweils die übergebenen Parameter, die Variablen, die Optionen und in einer Skizze auftretenden Punkte beinhalten. Letzteres ist sinnvoll, da sonst bei jedem Verweis auf einen Punkt dieser neu berechnet werden müsste. Ist ein Skizzenelement ein Punkt, so werden die Koordinaten berechnet und der Punkt in der Map gespeichert. Bei einer Abfrage des Punktes wird dieser nun nicht neu berechnet, sondern wird aus der Map abgerufen. Anhand eines Beispiels wird die Zweckmäßigkeit klar: Zum Beispiel soll eine Linie aus zwei Punkten erstellt werden. Punkt eins ist ein Schnittpunkt aus zwei Linien, wobei sich die Linien auf relative Punkte beziehen. Punkt zwei ist ein relativer Punkt, der sich wiederum auf einen anderen relativen Punkt bezieht. Nun ist es einfacher, die Koordinaten der betreffenden Punkte aus der Map abzufragen, als sie über mehrfache Aufrufe (die erst bei absoluten Punkten enden) zu berechnen.

Weiterhin enthält das Hauptprogramm Funktionen für Schnittpunktberechnungen zwischen zwei Elementen. Diese sind für die Verwendung des Komplexbeispiels selbst erstellt worden und nicht allgemein verwendbar. Zwar gibt es diese Routinen in der *remarc.dll*[5], dort sind sie jedoch nur als C-Quelltext vorhanden und eine Umwandlung in Java-Quelltext wäre aufwändiger gewesen als die Neuprogrammierung[6]. Bei der Erweiterung des Neutral-Formates muss geklärt werden, ob die Funktionen aus der *remarc.dll* übernommen werden (im Sinne einer Umwandlung von C- in Java-Quelltext), ob die *remarc.dll* (mit Hilfe von JNI[7]) an das Hauptprogramm angebunden wird, oder ob sämtliche benötigte Funktionen neu geschrieben werden.

[5] eine Bibliothek mit den Methoden der ISO 13584-31 und verschiedenen mathematischen Berechnungen; siehe Seite 11

[6] Die Umwandlung der genannten Funktionen in Java-Quelltext wurde versucht. Aufgrund der allgemeinen Verwendungsmöglichkeit dieser Funktionen hätte sehr viel Quelltext umgeschrieben werden müssen. Es war abzusehen, dass eine Neuprogrammierung deutlich angebrachter ist.

[7] Java Native Interface (JNI) ist eine API, um von Java aus auf Bibliotheken anderer Programmiersprachen zuzugreifen und umgekehrt. Siehe dazu: http://java.sun.com/docs/books/jni/ (JNI auf der Seite von Sun)

5.3 Generator zur Geometrieerzeugung

Der Generator enthält die Funktionen für die Erzeugung der Geometrie, die jeweils von dem Hauptprogramm aufgerufen werden. In den Funktionen wird auf die NXOpen Java API zugegriffen.

Um Teile in anderen CAD-Systemen erzeugen zu können, wurde ein Interface abgeleitet. Es enthält alle Funktionssignaturen, die von dem Hauptprogramm aufgerufen werden. Soll z. B. Pro/E in den Teileerstellungsprozess eingebunden werden, so werden die Funktionssignaturen des Interfaces übernommen und mit der API von Pro/E verknüpft.

5.4 Optionen

Die Optionen sind eine Sammlung von möglichen Einstellungen, die durch eine Aufzählung (Enumeration) dargestellt werden. Welche der Optionen verwendet werden, wird bei der Initialisierung der Übergabewerte festgelegt. Ausgewertet werden sie in dem Generator.
Derzeit werden nur zwei der Optionen verwendet: für Fasen und für einen Debugmodus. Ist die Option „Fasen“ gesetzt, so werden alle in der nf-Datei enthaltenen Fasen auch erzeugt. Andernfalls werden sie weggelassen. Der Debugmodus ist für den Entwickler bzw. Konstrukteur gedacht. Dabei wird bei einem Fehler in der Erzeugung eines Features nicht abgebrochen, sondern dieser wird ignoriert und die Abarbeitung fortgesetzt. Das kann zweifellos zu weiteren Fehlern führen. Für die Anwendung zur Teileerstellung wird die Option „Debugmodus“ nicht übergeben, weil ein fehlerhaftes Teil nicht verwendet werden kann.
Für die Zukunft sind weitere Optionen gedacht, beispielsweise für das Löschen der Konstruktionshistorie[8] und für die Auswahl zwischen nativem Gewinde und Gewindekerben.

[8] Die Konstruktionshistorie in einem CAD-System zeigt die Reihenfolge und Abhängigkeit aller verwendeten Features / Konstruktionsschritte.

6 Komplexbeispiel

Für den Beweis der Funktionalität und erfolgreicher Umsetzung der neuen Sprache soll als Beispiel eine Senkschraube mit Kreuzschlitz nach DIN EN ISO 7046-1 dienen. Diese Schraube wurde ausgesucht, weil in den zugehörigen Geometriedateien die meisten Elemente der neuen Sprache enthalten sind.
Zuerst soll anhand zwei konkreter Features der Unterschied zwischen den Geometriedateien und den nf-Dateien gezeigt werden.

In Quelltext 6.1 ist der Inhalt einer Geometriedatei, die als Unterroutine genutzt wird, dargestellt. Zuerst werden die Variablen, danach die Punkte und Linien festgelegt und zum Schluss wird die Rotation durchgeführt. Unwichtige Zeilen wurden der Übersichtlichkeit halber weggelassen. Die komplette Geometriedatei befindet sich im Anhang B.

```
1  FUNCTIOND80021(P1,P2,P3,P4,P5,P6,P7,P8,P9)
2   ...
3  DXX1=0.5D0*(P5-P1)/DTAN(P7*PI/180.0D0)
4   ...
5  DXX2=0.5D0*P1/DTAN(P7*PI/180.0D0)
6  DYY1=0.5D0*(P1-P6)
7  X=0.0D0          Y=0.0D0          Z=0.0D0
8  IPN(1)=PNT_CARTESIAN_ABSOLUTE(X,Y,Z,KFIX)
9  DX=-DXX2         DY=0.0D0         DZ=0.0D0
10 IPN(2)=PNT_CARTESIAN_RELATIVE(IPN(1),DX,DY,DZ,KFIX)
11 DX=-P4           DY=0.0D0         DZ=0.0D0
12 IPN(3)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
13 DX=0.0D0         DY=0.5D0*P5      DZ=0.0D0
14 IPN(4)=PNT_CARTESIAN_RELATIVE(IPN(3),DX,DY,DZ,KFIX)
```

```
15  DX=-DXX1          DY=0.5D0*P5          DZ=0.0D0
16  IPN(5)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
17  DX=0.0D0          DY=0.5D0*P1          DZ=0.0D0
18  IPN(6)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
19  DX=P2-P3          DY=0.0D0             DZ=0.0D0
20  IPN(7)=PNT_CARTESIAN_RELATIVE(IPN(6),DX,DY,DZ,KFIX)
21  DX=0.0D0          DY=-DYY1             DZ=0.0D0
22  IPN(8)=PNT_CARTESIAN_RELATIVE(IPN(7),DX,DY,DZ,KFIX)
23  DX=P8             DY=0.0D0             DZ=0.0D0
24  IPN(9)=PNT_CARTESIAN_RELATIVE(IPN(7),DX,DY,DZ,KFIX)
25  DX=P2             DY=0.0D0             DZ=0.0D0
26  IPN(10)=PNT_CARTESIAN_RELATIVE(IPN(6),DX,DY,DZ,KFIX)
27  DX=P2             DY=0.0D0             DZ=0.0D0
28  IPN(11)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
29  IGR1=CREATE_GRP()
30  ILIN(1)=LIN_2_PNT(IPN(11),IPN(3),KFIX)
31  ILIN(2)=LIN_2_PNT(IPN(3),IPN(4),KFIX)
32  ILIN(3)=LIN_2_PNT(IPN(4),IPN(5),KFIX)
33  ILIN(4)=LIN_2_PNT(IPN(5),IPN(6),KFIX)
34  IF(P2.GT.P3)THEN
35  ILIN(5)=LIN_2_PNT(IPN(6),IPN(7),KFIX)
36  ENDIF
37  ILIN(6)=LIN_2_PNT(IPN(7),IPN(8),KFIX)
38  ILIN(7)=LIN_2_PNT(IPN(8),IPN(9),KFIX)
39  ILIN(8)=LIN_2_PNT(IPN(9),IPN(10),KFIX)
40  ILIN(9)=LIN_2_PNT(IPN(10),IPN(11),KFIX)
41  ...
42  CALLCLOSE_GRP()
43  N=1
44  ICTR1=CTR_GEN(N,IGR1,KFIX)
45  N=0
46  IAPS1=APS_GEN(ICTR1,N,NULL,KFIX)
47  X=1.0D0           Y=0.0D0              Z=0.0D0
48  IDIR1=DIR_COMPONENT(X,Y,Z,KFIX)
```

```
49  IACHS1=A1P_GEN(IPN(1),IDIR1,KFIX)
50  ANG1=360.0D0
51  D80021=SLD_REVOLUTION(IAPS1,ANG1,IACHS1,KFIX)
```

Quelltext 6.1: Geometriedatei zur Erstellung einer Rotation

In der Xtext-Grammatik ist eine Rotation gemäß Kapitel 3.2 aufgebaut (siehe Quelltext 6.2). Der deutlichste Unterschied ist, dass hier die Rotation als Erstes angegeben und danach alles Zugehörige festgelegt wird. Vor jeder Zuweisung eines Attributs steht eine Beschriftung, um die Zugehörigkeit der Eingabe zu verdeutlichen.
Zur weiteren Betrachtung ist die gesamte Xtext-Grammatik im Anhang C dargestellt.

```
1  Rotation : 'Rotation ' 'Name: ' name=ID skizze=Skizze
2    'Achse: ' (bezug=[Achse] | typ=Achstyp) ('Startwinkel: '
3    startWinkel=Ausdruck)? 'Endwinkel: ' endWinkel=Ausdruck
4    ('Booltyp: ' booltyp=BoolTyp 'Grundkörper: '
5    grundkoerper=[VolumenFeature])? ;
6  Skizze : 'Skizze: ' 'Name: ' name=ID 'Lage: ' lage=Lage '{ '
7    (elemente+=Skizzenelement)* '} ' ;
```

Quelltext 6.2: Xtext-Grammatik von Rotation und Skizze

In dem Neutral-Format wurde die oben angegebene Geometriedatei auch als Unterroutine definiert, wie in Quelltext 6.3 abgebildet ist. Die Übergabeparameter sind hier extra aufgelistet, während sie in der Geometriedatei im Dateikopf stehen. Die Angaben, die mit einer Raute gekennzeichnet sind, dienen als Hilfestellung.
Auffällig ist die geringe Anzahl der Punkte und Linien, was damit zu begründen ist, dass die nf-Datei nicht umgewandelt, sondern neu aufgebaut wurde. Dabei wurden überflüssige Punkte ausgelassen. Damit keine Fehler bei der Geometrieerstellung passieren, muss bei einer eventuellen, späteren Umwandlung der Geometriedatei der Aufbau komplett übernommen werden, da die Unterroutine eventuell auch für an-

dere Normen genutzt wird.
Variablen wurden weggelassen, da sie an dieser Stelle nicht nötig sind und die Werte bzw. Berechnungen direkt bei den Koordinaten angegeben werden.

```
1  Parameter d1 #1#gewindedurchmesser
2  Parameter d2 #2#nennlaenge
3  Parameter d5 #5#kopfhoehe
4  Parameter d9 #9#senkwinkel
5  Parameter d11 #11#referenzpunktverschiebung
6
7  Rotation
8    Name: schraubenKoerper
9    Skizze:
10     Name: s1
11     Lage: XZ
12     {
13       Absoluter Punkt Name: punkt1 x: -d11 y: d1 / 2 ;
14       Absoluter Punkt Name: punkt2 x: -d11-d5
15          y: d1 / 2.0 + tan(0.5 * d9) * d5 ;
16       Absoluter Punkt Name: punkt3 x: -d11-d5 y: 0.0 ;
17       Absoluter Punkt Name: punkt4 x: -d11-d5+d2 y: 0.0 ;
18       Absoluter Punkt Name: punkt5 x: -d11-d5+d2 y: d1 / 2 ;
19       Linie Name: linie1 Punkt1: punkt1 Punkt2: punkt2 ;
20       Linie Name: linie2 Punkt1: punkt2 Punkt2: punkt3 ;
21       Linie Name: linie3 Punkt1: punkt3 Punkt2: punkt4 ;
22       Linie Name: linie4 Punkt1: punkt4 Punkt2: punkt5 ;
23       Linie Name: linie5 Punkt1: punkt5 Punkt2: punkt1 ;
24     }
25   Achse: X
26   Endwinkel: 360.0
```

Quelltext 6.3: Eine Neutral-Format-Datei zur Beschreibung einer Rotation

Wird der Generator mit dieser Routine ausgeführt, so entsteht der Grundkörper der Schraube, wie es in Abbildung 6.1 dargestellt ist.

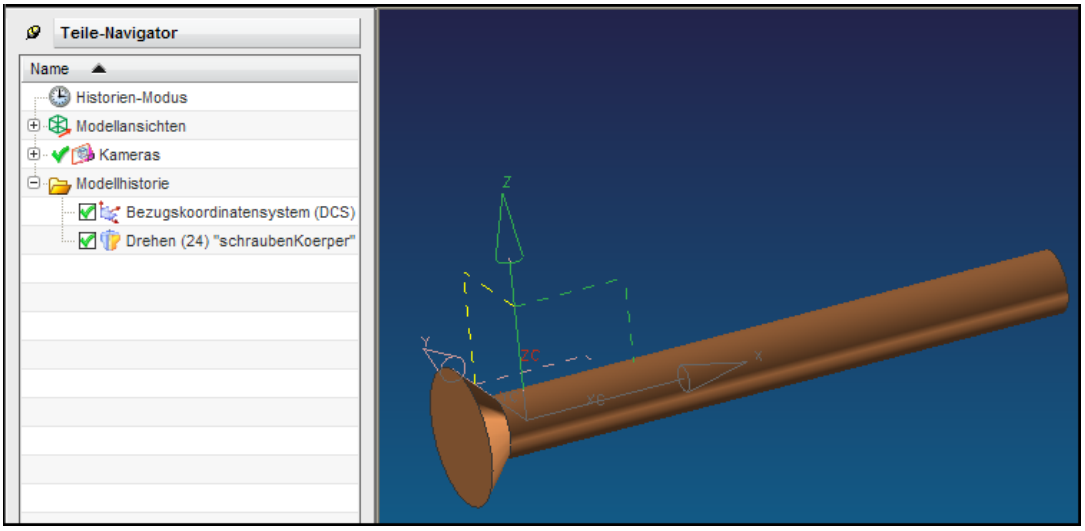

Abbildung 6.1: Der Grundkörper einer Schraube, der aus dem Quelltext 6.3 generiert wird. Links in der Modellhistorie ist der Name des Features („schraubenKoerper“) zu sehen.

Das zweite Beispiel ist die Angabe einer Fase.

In der Geometriedatei vom vorigen Beispiel (Quelltext 6.1) wurde nach der Definition der Linien in Zeile 41 Quelltext weggelassen. In diesem Bereich wird die Fase erzeugt (siehe Quelltext 6.4). Da die ISO 13584-31 kein Fasen-Feature kennt, wird hier im 2D-Raum gearbeitet. Die Methode *LIN_CHAMFER_2_LIN* bewirkt eine Verkürzung der angeführten Linien (*ILIN(8)* und *ILIN(9)*) an deren Schnittpunkt um die angegebenen Längen (*P9* und *DYY1*). Durch die spätere Rotation sieht diese Verkürzung im 3D-Raum wie eine Fase aus.

```
1  IF(P9.GT.EPS)THEN
2  ILIN(10)=LIN_CHAMFER_2_LIN(P9,DYY1,ILIN(8),ILIN(9),KFIX)
3  ENDIF
```

Quelltext 6.4: Auszug aus einer Geometriedatei. Die ISO 13584-31-Methode *LIN_CHAMFER_2_LIN* definiert die Erstellung einer Fase.

In der Xtext-Grammatik (siehe Quelltext 6.5) wird eine Fase entweder durch zwei Werte oder durch einen Wert und einen Winkel definiert (gemäß Kapitel 3.2). Zwar

wird in der Geometriedatei die Fase aus zwei Werten gebildet, doch wurde in der nf-Datei die Fase aus Wert und Winkel verwendet, damit die zweite Länge nicht berechnet werden muss.

```
1  Fase_Wert_Winkel : 'Fase aus Wert und Winkel ' 'Name: '
2    name=ID 'Länge der Fase: ' laenge=Ausdruck 'Winkel: '
3    winkel=Ausdruck 'Kantenart: ' kantenTyp=KantenTyp
4    ('Alternative: ' stelle=INT)? ;
```

Quelltext 6.5: Xtext-Grammatik für die Definition einer Fase aus einem Wert und einem Winkel

In dem Neutral-Format wird die Fase als Feature verwendet und ist nicht in der Skizzengeometrie zu finden. Das Feature wird in der Hauptroutine beschrieben, kann sich aber auch genauso in der Unterroutine befinden. Wie in Quelltext 6.6 zu sehen ist, handelt es sich um eine kreisrunde Fase. Die Kante wird im angegebenen Abstand[9] vom Nullpunkt in x-Richtung gesucht.
Der Vorteil der Fasen-Darstellung in der Skizze ist, dass auf die zwei Linien referenziert werden kann, wohingegen die Kante im 3D-Raum durch die Rotation erst entsteht. Bei dieser einfachen Geometrie ist es leicht, die Position der Kante herauszufinden bzw. zu berechnen. Bei komplizierter Geometrie oder bei sehr großen Bauteilen kann es durchaus passieren, dass das Bauteil erst einmal ohne Fase erzeugt und der Kantenabstand mit Hilfe des CAD-Systems ermittelt werden muss.

```
1  Fase aus Wert und Winkel
2    Name: fase
3    Länge der Fase: ( A03–DGK ) / 2
4    Winkel: 60.0
5    Kantenart: CIRCULAR
6    Lage der Kante: X
7    Abstand: –DRP – F + B
```

Quelltext 6.6: Neutral-Format-Quellcode zur Beschreibung einer Fase

[9] *DRP*, *F* und *B* sind Parameter

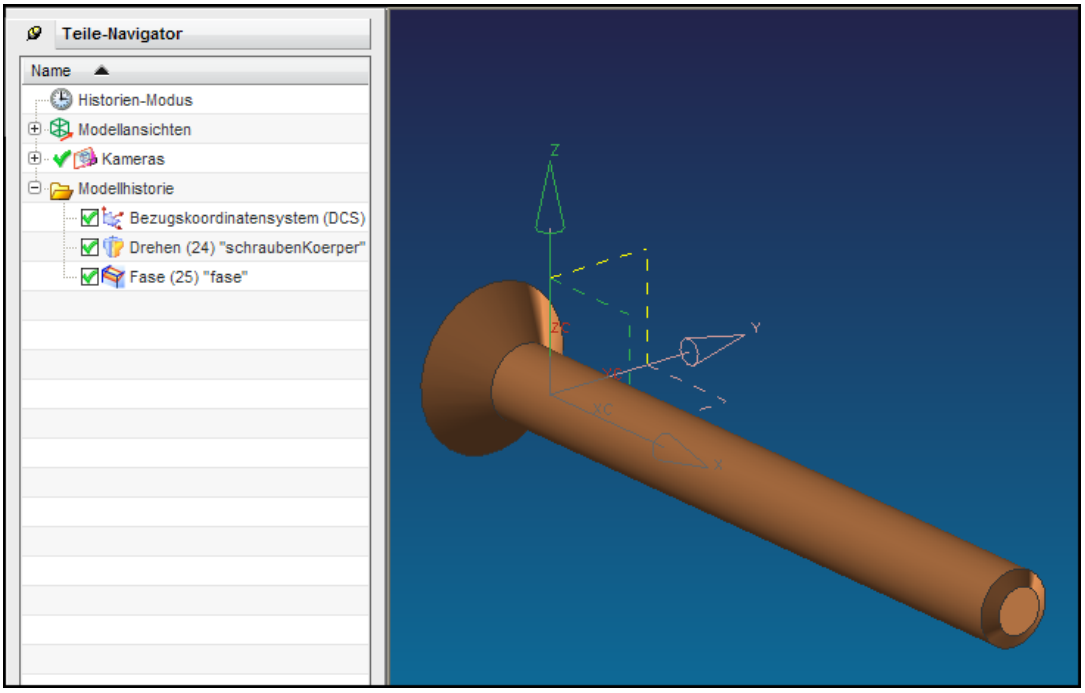

Abbildung 6.2: Der Grundkörper einer Schraube mit Fase, der aus den Quelltexten 6.3 und 6.6 generiert wird. Die Fase ist links in der Modellhistorie und rechts am Teil zu sehen.

Wird die Rotation zusammen mit der Fase umgesetzt, so entsteht das in Abbildung 6.2 gezeigte Teil.

Darüber hinaus soll auch der Unterschied zwischen den auf verschiedenen Wegen erstellten Schrauben gezeigt werden. In Abbildung 6.3 ist die durch die Geometriedateien und in Abbildung 6.4 die mit Hilfe der nf-Dateien erzeugte Schraube dargestellt. Im Vergleich zur ursprünglichen Schraube zeigen sich folgende Unterschiede:

▷ Im Historienbaum ist zu erkennen, dass alle Features benannt sind.

▷ Der zweite Eintrag ist eine gedrehte Ebene, die für die Erstellung des Schraubenkopfes benötigt wird.

▷ Es gibt einige Features mehr. Dies kommt daher, dass die Boolesche Operation *Schnittmenge* durch einen Block und zwei *Subtrahieren*-Operationen ersetzt wurden.

▷ Der Schraubenkopf ist um 45 Grad gedreht, was mit der abweichenden Konstruktion zu begründen ist. In der Praxis hat das keine Bedeutung.

▷ Die Farbe ist anders, da im Neutral-Format bisher keine Farben definiert sind und daher das CAD-System die Standardfarbe verwendet.

- In Abbildung 6.3 ist außerdem die Kerbe zu sehen, die das Gewinde andeutet.
- Nicht zu sehen ist der Unterschied bei den Attributen und der Layer-Belegung.

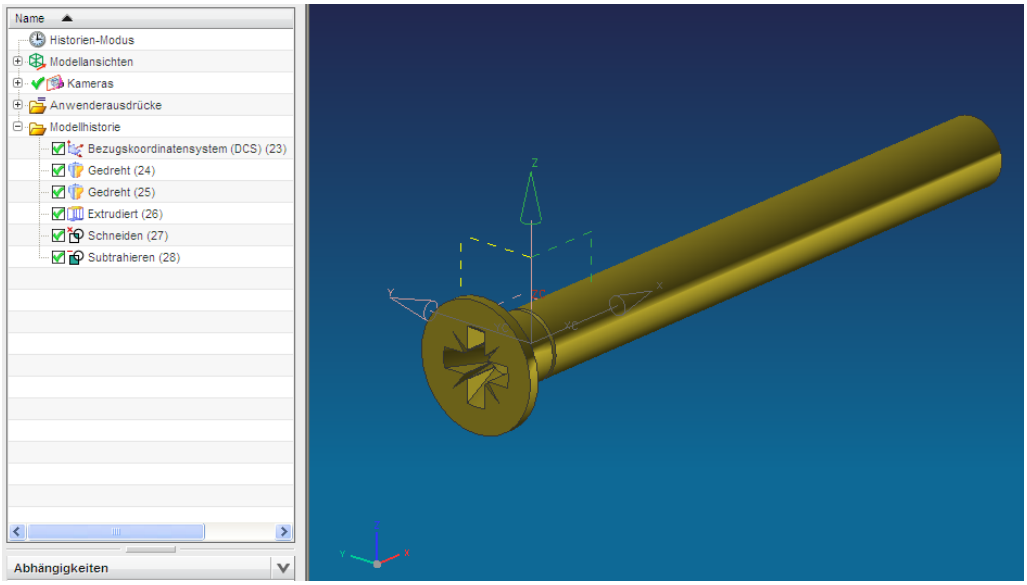

Abbildung 6.3: Eine Schraube nach DIN EN ISO 7046-1, die aus den Geometriedateien erzeugt wird.

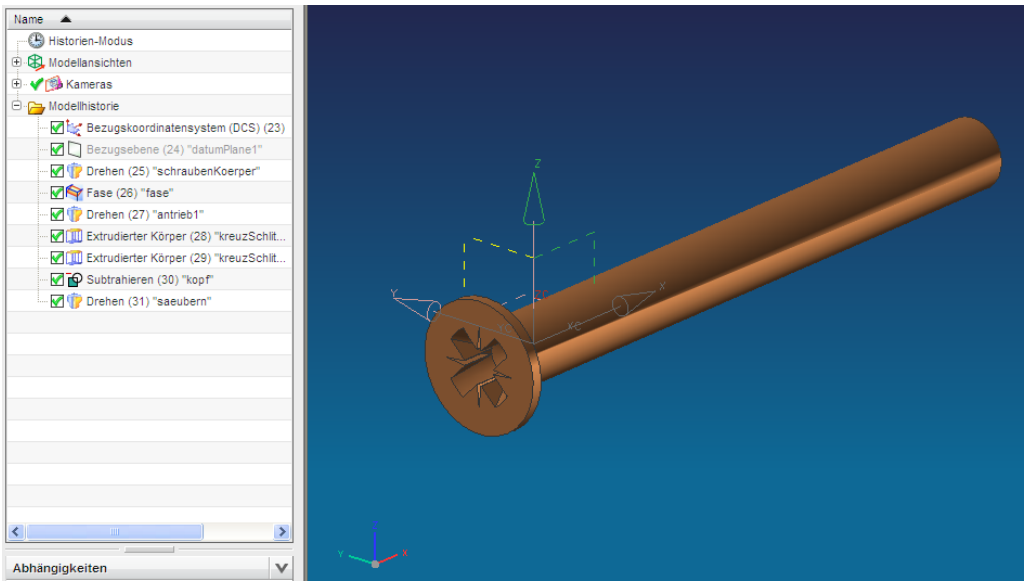

Abbildung 6.4: Eine Schraube nach DIN EN ISO 7046-1, die aus den nf-Dateien erzeugt wird.

7 Zusammenfassung und Ausblick

Anliegen der Arbeit war es, eine neue Sprache zur Beschreibung von CAD-Teilen zu entwickeln und die Eignung mit Hilfe eines Beispiels und eines prototypischen Generators nachzuweisen. Die Bestätigung konnte erfolgreich erbracht werden, auch wenn einige Bereiche noch nicht vollständig implementiert wurden.

Für die Umsetzung der Ansprüche an die Sprache wurde eine Technologie gewählt, die nach jetzigem Erkenntnisstand als die am Besten geeignete erscheint. Die verwirklichten Anforderungen an die Sprache konnten mit nur wenig Schwierigkeiten umgesetzt werden. Mit Hilfe des Editors ist die Bearbeitung von nf-Dateien eine große Erleichterung, auch wenn hinsichtlich der Benutzerfreundlichkeit noch ein paar Verbesserungen erforderlich sind. Zum Beispiel würden Kommentare zu den angezeigten Vorschlägen (bei Drücken von Steuerung und Leertaste) die Unterscheidung von optionalen Attributen ermöglichen und das Überprüfen anhand der Xtext-Definition ersetzen.

Von der erfolgreichen Implementierung abgesehen gibt es derzeit drei größere Probleme:

1. In der ISO 13584-31 gibt es eine Methode, die aus zwei Körpern die Schnittmenge bildet, welche aber nur wenige CAD-Systeme (wie *CATIA V4* oder *NX*) verarbeiten können. Diese Methode muss nun so abgebildet werden, dass sie von allen CAD-Systemen verarbeitet werden kann. Die einzige gefundene Möglichkeit ist, einen Block zu erstellen und den ersten Körper davon abzuziehen. Danach wird der Block vom zweiten Körper abgezogen. Damit keine Fehler entstehen, muss der Block für diese Operationen allerdings groß genug sein, um beide Körper zu überdecken. Nachteil ist, dass im Historienbaum anstelle eines Eintrages nun drei verzeichnet sind.

2. Für die native Darstellung von Gewinde wurde bisher keine Funktion in der NXOpen Java API gefunden.

3. Ist eine Skizze direkt auf einer Fläche und nicht auf einer Ebene platziert, so befindet sich der Ursprung des Skizzenkoordinatensystems (SCS) immer in der Mitte der Fläche. Das bedeutet eine Verschiebung der Skizzengeometrie um den Ursprung des SCS und führt zu Konflikten, da die Koordinaten der Skizzengeometrie in der nf-Datei vom Weltkoordinatensystem ausgehen und es nur schwer vorhersehbar ist, wo der Ursprung des SCS liegt.

An der Bewältigung der Schwierigkeiten wurde gearbeitet und daher z. B. für den ersten Punkt ein akzeptables Ergebnis geschaffen, aber keine endgültige Lösung gefunden.

Die Sprache

Die Sprache ist zwar in sich geschlossen und kann durchaus schon verwendet werden, allerdings ist sie beschränkt auf die implementierten Elemente. Damit ist sie nicht sehr leistungsfähig in dem Sinne, dass sie weder als sinnvolle Alternative zur Modellierung, noch als Ablösung der Geometriedateien und der ISO 13584-31 dienen kann. Daher sind folgende Punkte die nächsten Schritte zur Erweiterung und Verbesserung der Sprache:

1. Die nicht umgesetzten Anforderungen müssen noch geleistet werden.

2. vollständiger Test aller Sprachelemente und deren Kombinationen

3. Die Sprache ist anhand eines weiteren CAD-Systems zu prüfen.

Abgesehen von der vollständigen Umsetzung der Anforderungen ist die Einbindung der folgenden Features bzw. die Erweiterung der bestehenden sinnvoll:

- Ebenen können auch eine Fläche als Bezug haben
- Extrusion entlang eines beliebigen Vektors
- Rotation um einen beliebigen Vektor
- Achse aus Punkt und Vektor; Achse aus Schnittpunkt von zwei Ebenen
- Ebene aus Punkt und Vektor

Schon jetzt ist die Sprache sehr umfangreich und komplex und macht eine Dokumentation unentbehrlich, damit jeder Anwender sie verstehen und nutzen kann.

Generator und Hauptprogramm

Für eine vollständige Fertigstellung des Generators und Hauptprogramms nach derzeitigem Stand ist noch Folgendes zu tun:

- Hinzufügen der Beziehungen *Horizontal*, *Vertikal* und *Senkrecht*
- Ergänzung der (Element)Spiegelung um weitere Elemente
- Implementierung eines Kreises aus drei Punkten (die Kernaufgabe liegt hier, aus den drei Punkten den Mittelpunkt zu berechnen)
- Verallgemeinerung der Schnittpunktberechnungen und Abbruch bei einem Fehler (z. B. kein Fund eines Schnittpunktes)

Ablösung der Geometriedateien

Bei der Umwandlung der Geometriedateien in das Neutral-Format müssen die nachstehenden Aspekte beachtet werden:

1. Das Neutral-Format muss vollständig sein, d. h. der Satz aller Methoden der ISO 13584-31 muss auf die neue Sprache abbildbar sein. Dies ist gegeben, wenn die noch nicht umgesetzten Anforderungen erfüllt sind.
2. Die Geometriedateien müssen eingelesen und umgewandelt werden. Für diese Aufgabe kann (wiederum) ANTLR verwendet werden. In [Grä09] wurden Merkmaldateien, die auch auf Fortran beruhen, in eine andere Struktur überführt. Auf diese Grundlage kann aufgebaut und die Grammatik für Geometriedateien, die jedoch erst erstellt werden muss, hinzugefügt werden. Den Abschluss bildet ein Programm, das nun jede Geometriedatei einliest und in das Neutral-Format konvertiert.

zukünftiges Umfeld

Wie in Kapitel 5.1 schon erwähnt wurde, soll das System (Hauptprogramm und Generator) für die zukünftige Verwendung in REMARC als weiterer Teilegenerator eingebunden werden. Hierfür ist eine Veränderung des Hauptprogramms bedeutend.

Derzeit wird zu Beginn eine Session erzeugt und lediglich ein einzelnes Bauteil erstellt. Für die Erstellung von vielen Teilen ist es jedoch äußerst uneffizient, für jedes Teil eine neue Session zu erzeugen. Bei der Umprogrammierung muss erreicht werden, dass nur eine einzige Session für die Erzeugung einer beliebigen Anzahl von Teilen verwendet wird.

Ein weiterer Ausbau ist die Teileerzeugung für andere CAD-Systeme, da die Sprache bisher allein mit NX getestet wurde. Das Interface, was aus dem Generator abgeleitet wurde, ist genau für diesen Zweck erstellt worden (siehe Kapitel 5.3). Dieser Schritt geht einher mit dem Aspekt, die Sprache anhand eines weiteren CAD-Systems zu prüfen.

Bisher wurde ausschließlich über eine Konvertierungsrichtung gesprochen, das heißt, wie aus einer nf-Datei eine native Datei wird. Interessant ist nun der umgekehrte Fall, bei dem ein natives Format in das Neutral-Format umgewandelt wird. Dabei sind zwei Wege denkbar. Erstens, eine native Datei wird ins Neutral-Format überführt und danach parametrisiert. Oder zweitens, das Teil wird im CAD-System mit Parametern versehen und danach konvertiert. Der erste Vorschlag wurde anhand eines einfachen Teils in *NX* getestet und stellte keine Probleme dar. Die Parametrisierung sollte auch ohne Schwierigkeiten durchzuführen sein. Dessen ungeachtet steigt der Aufwand und die Fehleranfälligkeit mit erhöhter Bauteilkomplexität. Der zweite Vorschlag muss getestet werden und hängt lediglich davon ab, ob Parameter aus dem Teil ausgelesen werden können.

Anhang

A Auflistung der Anreicherungen

- Features
  - Bezüge
    - Koordinatensystem
    - Achse / Mittellinie
    - Ebene
  - Geometrie
    - Fase
    - Gewinde
    - ...
  - Routing
    - Port
    - Anker
- Metadaten
  - Farbe
    - Einzelteil
    - Baugruppe
    - Fläche
    - Achse / Mittellinie
  - Werkstoff (Material)
  - Attribut (Eigenschaft)
  - Layer (Kategorien)
    - Solid
    - Gewinde
    - ...
  - Reference Sets (Kategorien)
  - Einheit (mm / ")
  - Toleranzen
  - Bemaßung
- Berechnungen

B vollständige Geometriedatei

```
1   FUNCTIOND80021(P1,P2,P3,P4,P5,P6,P7,P8,P9)
2   INTEGERD80021
3   DOUBLEPRECISIONP1
4   DOUBLEPRECISIONP2
5   DOUBLEPRECISIONP3
6   DOUBLEPRECISIONP4
7   DOUBLEPRECISIONP5
8   DOUBLEPRECISIONP6
9   DOUBLEPRECISIONP7
10  DOUBLEPRECISIONP8
11  DOUBLEPRECISIONP9
12  INTEGERPNT_CARTESIAN_ABSOLUTE
13  INTEGERPNT_CARTESIAN_RELATIVE
14  INTEGERLIN_2_PNT
15  INTEGERCREATE_GRP
16  INTEGERLIN_CHAMFER_2_LIN
17  INTEGERCTR_GEN
18  INTEGERAPS_GEN
19  INTEGERDIR_COMPONENT
20  INTEGERA1P_GEN
21  INTEGERSLD_REVOLUTION
22  INTEGERKFIX,N,NULL,IPN(11),ILIN(10),IGR1,ICTR1,IAPS1,
        IDIR1,IACHS1
23  DOUBLEPRECISIONEPS,PI,DXX1,DXX2,DYY1,X,Y,Z,DX,DY,DZ,ANG1
24  EPS=0.00001D0
```

```
25      PI=3.14159265358979323846D0
26      D80021=0
27      IF(P1.LE.EPS)GOTO999
28      IF(P2.LT.P3)GOTO999
29      IF(P3.LE.EPS)GOTO999
30      IF(P5.LE.P1)GOTO999
31      IF(P6.LE.EPS)GOTO999
32      IF(P6.GE.P1)GOTO999
33      IF(P7.LE.EPS)GOTO999
34      IF(P7.GE.90.0D0)GOTO999
35      IF(P8.LE.EPS)GOTO999
36      IF(P8.GE.P3-P9)GOTO999
37      IF(P9.GE.P3-P8)GOTO999
38      DXX1=0.5D0*(P5-P1)/DTAN(P7*PI/180.0D0)
39      IF((P4-DXX1).LE.EPS)GOTO999
40      KFIX=0
41      NULL=0
42      DXX2=0.5D0*P1/DTAN(P7*PI/180.0D0)
43      DYY1=0.5D0*(P1-P6)
44      X=0.0D0
45      Y=0.0D0
46      Z=0.0D0
47      IPN(1)=PNT_CARTESIAN_ABSOLUTE(X,Y,Z,KFIX)
48      DX=-DXX2
49      DY=0.0D0
50      DZ=0.0D0
51      IPN(2)=PNT_CARTESIAN_RELATIVE(IPN(1),DX,DY,DZ,KFIX)
52      DX=-P4
53      DY=0.0D0
54      DZ=0.0D0
55      IPN(3)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
56      DX=0.0D0
57      DY=0.5D0*P5
58      DZ=0.0D0
```

```
59      IPN(4)=PNT_CARTESIAN_RELATIVE(IPN(3),DX,DY,DZ,KFIX)
60      DX=-DXX1
61      DY=0.5D0*P5
62      DZ=0.0D0
63      IPN(5)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
64      DX=0.0D0
65      DY=0.5D0*P1
66      DZ=0.0D0
67      IPN(6)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
68      DX=P2-P3
69      DY=0.0D0
70      DZ=0.0D0
71      IPN(7)=PNT_CARTESIAN_RELATIVE(IPN(6),DX,DY,DZ,KFIX)
72      DX=0.0D0
73      DY=-DYY1
74      DZ=0.0D0
75      IPN(8)=PNT_CARTESIAN_RELATIVE(IPN(7),DX,DY,DZ,KFIX)
76      DX=P8
77      DY=0.0D0
78      DZ=0.0D0
79      IPN(9)=PNT_CARTESIAN_RELATIVE(IPN(7),DX,DY,DZ,KFIX)
80      DX=P2
81      DY=0.0D0
82      DZ=0.0D0
83      IPN(10)=PNT_CARTESIAN_RELATIVE(IPN(6),DX,DY,DZ,KFIX)
84      DX=P2
85      DY=0.0D0
86      DZ=0.0D0
87      IPN(11)=PNT_CARTESIAN_RELATIVE(IPN(2),DX,DY,DZ,KFIX)
88      IGR1=CREATE_GRP()
89      ILIN(1)=LIN_2_PNT(IPN(11),IPN(3),KFIX)
90      ILIN(2)=LIN_2_PNT(IPN(3),IPN(4),KFIX)
91      ILIN(3)=LIN_2_PNT(IPN(4),IPN(5),KFIX)
92      ILIN(4)=LIN_2_PNT(IPN(5),IPN(6),KFIX)
```

```
93      IF(P2.GT.P3)THEN
94      ILIN(5)=LIN_2_PNT(IPN(6),IPN(7),KFIX)
95      ENDIF
96      ILIN(6)=LIN_2_PNT(IPN(7),IPN(8),KFIX)
97      ILIN(7)=LIN_2_PNT(IPN(8),IPN(9),KFIX)
98      ILIN(8)=LIN_2_PNT(IPN(9),IPN(10),KFIX)
99      ILIN(9)=LIN_2_PNT(IPN(10),IPN(11),KFIX)
100     IF(P9.GT.EPS)THEN
101     ILIN(10)=LIN_CHAMFER_2_LIN(P9,DYY1,ILIN(8),ILIN(9),KFIX)
102     ENDIF
103     CALLCLOSE_GRP()
104     N=1
105     ICTR1=CTR_GEN(N,IGR1,KFIX)
106     N=0
107     IAPS1=APS_GEN(ICTR1,N,NULL,KFIX)
108     X=1.0D0
109     Y=0.0D0
110     Z=0.0D0
111     IDIR1=DIR_COMPONENT(X,Y,Z,KFIX)
112     IACHS1=A1P_GEN(IPN(1),IDIR1,KFIX)
113     ANG1=360.0D0
114     D80021=SLD_REVOLUTION(IAPS1,ANG1,IACHS1,KFIX)
115 999 RETURN
116     END
```

C vollständige Xtext-Grammatik des Neutral-Formats

```
1  grammar de.arcsolutions.remarc.NeutralFormat hidden (WS,
       ML_COMMENT, SL_COMMENT)
2  import "http://www.eclipse.org/emf/2002/Ecore" as ecore
3  generate neutralFormat "http://www.arcsolutions.de/remarc/
       NeutralFormat"
4
5  Modell : 'Teil-Name: ' name=ID (elemente += Feature*) ;
6
7  Feature: VolumenFeature | Fase | Ebene | Achse |
       ParameterDef | Variable | If | Gewinde ;
8  VolumenFeature : Extrusion | Rotation | Block_2_Punkte |
       BoolOperation | Spiegeln | ImportFeature ;
9  Extrusion : 'Extrusion ' 'Name: ' name=ID skizze=Skizze '
       Achse: ' (bezug=[Achse] | typ=Achstyp)
10   ('Startwert: ' startWert=Ausdruck)? 'Endwert: ' endWert=
       Ausdruck
11   ('Booltyp: ' booltyp=BoolTyp 'Grundkörper: ' grundkoerper
       =[VolumenFeature])? ;
12 Rotation : 'Rotation ' 'Name: ' name=ID skizze=Skizze '
       Achse: ' (bezug=[Achse] | typ=Achstyp)
13   ('Startwinkel: ' startWinkel=Ausdruck)? 'Endwinkel: '
       endWinkel=Ausdruck
14   ('Booltyp: ' booltyp=BoolTyp 'Grundkörper: ' grundkoerper
       =[VolumenFeature])? ;
15 Skizze : 'Skizze: ' 'Name: ' name=ID 'Lage: ' lage=Lage '{
```

```
    ' (elemente+=Skizzenelement)* '} ' ;
16  Lage : Lage_Ebene | Lage_Grundebene ;
17  Lage_Ebene : ebene=[Ebene] ;
18  Lage_Grundebene : typ=Ebenentyp ('Position der Fläche: '
      position=Ausdruck ('Bezugs-Feature: ' stelle=[
      VolumenFeature])? )? ;
19  BoolOperation : 'BoolOperation ' 'Name: ' name=ID '
      Grundkörper: ' grundkoerper=[VolumenFeature]
20    'Werkzeug: ' werkzeug=[VolumenFeature] 'Booltyp: '
        booltyp=BoolTyp ;
21  Block_2_Punkte : 'Block aus 2 diagonalen Punkten ' 'Name: '
       name=ID
22    'Punkt1: ' punkt1=Punkt3D 'Punkt2: ' punkt2=Punkt3D ('
        Booltyp: ' booltyp=BoolTyp
23    'Grundkörper: ' grundkoerper=[VolumenFeature])? ;
24  Punkt3D : 'x: ' x=Ausdruck 'y: ' y=Ausdruck 'z: ' z=
      Ausdruck ;
25  Spiegeln : 'Spiegeln ' 'Name: ' name=ID 'Spiegelebene: ' (
      bezug=[Ebene] | typ=Ebenentyp)
26    '{ ' (features+=[VolumenFeature])+ '} ' ;
27  ImportFeature : 'ImportFeature ' 'FeatureName: ' name=ID '
      Import: ' importName=ID
28    '(' parameter=Ausdruck ( ', ' parameter_r+=Ausdruck )* ')
        ' ;
29  Fase : Fase_2_Werte | Fase_Wert_Winkel ;
30  Fase_Wert_Winkel : 'Fase aus Wert und Winkel ' 'Name: '
      name=ID 'Länge der Fase: ' laenge=Ausdruck
31    'Winkel: ' winkel=Ausdruck 'Kantenart: ' kantenTyp=
        KantenTyp
32    ('Alternative: ' stelle=INT)? ;
33  Fase_2_Werte : 'Fase aus 2 Werten ' 'Name: ' name=ID '
      Länge1 der Fase: ' laenge1=Ausdruck
34    'Länge2 der Fase: ' laenge2=Ausdruck 'Kantenart: '
        kantenTyp=KantenTyp
35    ('Alternative: ' stelle=INT)? ;
36  KantenTyp : Kante_Circular | Kante_Linear ;
```

```
37 Kante_Circular : 'CIRCULAR ' 'Lage der Kante: ' koordinate=
      Koordinate
38   'Abstand: ' position=Ausdruck ;
39 Kante_Linear : 'LINEAR ' 'Lage1 der Kante: ' koordinate1=
      Koordinate 'Abstand: ' position1=Ausdruck
40   'Lage2 der Kante: ' koordinate2=Koordinate 'Abstand: '
       position2=Ausdruck ;
41 Ebene : Ebene_Parallel | Ebene_Gedreht | Ebene_Grundebene ;
42 Ebene_Parallel : 'parallele Ebene ' 'Name: ' name=ID 'Bezug
      : ' (bezug=[Ebene] | typ=Ebenentyp)
43   'Abstand: ' abstand=Ausdruck ;
44 Ebene_Gedreht : 'gedrehte Ebene ' 'Name: ' name=ID 'Bezug:
      ' (bezugsEbene=[Ebene] | ebenenTyp=Ebenentyp)
45   'Achse: ' (bezugsAchse=[Achse] | achsTyp=Achstyp) 'Winkel
       : ' winkel=Ausdruck ;
46 Ebene_Grundebene : 'Grundebene ' 'Name: ' name=ID 'Lage: '
      typ=Ebenentyp ;
47 Achse : Achse_Grundachse ;
48 Achse_Grundachse : 'Grundachse ' 'Name: ' name=ID 'Lage: '
      typ=Achstyp ;
49 ParameterDef : 'Parameter ' name=ID (parameter=Parameter)?
      ;
50 Parameter : '#' int=INT ('#' id=ID) ? ;
51 Variable : 'Variable ' name=ID ('=' berechnung=Ausdruck)? ;
52 If : 'IF ' bedingung1=Bedingung ( 'ELSEIF ' bedingung2+=
      Bedingung )* ( 'ELSE ' bedingung3=BedingungAnweisung )?
      ;
53 Bedingung : kopf=BedingungKopf anweisungen=
      BedingungAnweisung ;
54 BedingungKopf : '(' ausdruck1=Ausdruck zeichen=( '<' | '>'
      | '== ' | '<=' | '>=' | '!=' ) ausdruck2=Ausdruck ')' ;
55 BedingungAnweisung : (variable+=[Variable] '=' ausdruck+=
      Ausdruck)+ ;
56 Gewinde : 'Gewinde ' 'Anfang: ' anfang=Ausdruck 'Ende: '
      ende=Ausdruck 'Feature: ' feature=[VolumenFeature] ;
57 enum BoolTyp : SUBTRAHIEREN | VEREINIGEN | SCHNITTMENGE ;
```

```
58  enum Koordinate : X | Y | Z ;
59  enum Ebenentyp : XY | XZ | YZ ;
60  enum Achstyp : X | Y | Z ;
61
62
63  Skizzenelement : Element | Beziehung | Option ;
64  Element : Linie | Punkt2D | Kreis | Bogen | Spiegeln_Skizze
        ;
65  Linie : 'Linie ' 'Name: ' name=ID 'Punkt1: ' punkt1=[
       Punkt2D | FQN]
66    'Punkt2: ' punkt2=[Punkt2D | FQN] (referenz?='-Hilfslinie
        - ')? (imCADnichtZuSehen?='imCADnichtZuSehen ')? ';' ;
67  Punkt2D : Punkt2D_Abs | Punkt2D_Aus_Schnittpunkt_2_Linien |
       Punkt2D_Aus_Schnittpunkt_KreisBogen_Linie | Punkt2D_Rel
        ;
68  Punkt2D_Abs : 'Absoluter Punkt ' 'Name: ' name=ID 'x: ' x=
       Ausdruck 'y: ' y=Ausdruck
69    (imCADzuSehen?='imCADzuSehen ')? ';' ;
70  Punkt2D_Rel : 'Relativer Punkt ' 'Name: ' name=ID ( 'PunktX
       : ' punktX=[Punkt2D | FQN] )? 'x: ' x=AusdruckRel
71    ( 'PunktY: ' punktY=[Punkt2D | FQN] )? 'y: ' y=
        AusdruckRel (imCADzuSehen?='imCADzuSehen ')? ';' ;
72  Punkt2D_Aus_Schnittpunkt_2_Linien : 'Punkt aus Schnittpunkt
       2er Linien ' 'Name: ' name=ID 'Linie1: '
73    linie1=[Linie] 'Linie2: ' linie2=[Linie] (imCADzuSehen?='
        imCADzuSehen ')? ';' ;
74  Punkt2D_Aus_Schnittpunkt_KreisBogen_Linie : 'Punkt aus
       Schnittpunkt Kreis / Bogen und Linie '
75    'Name: ' name=ID 'Kreis / Bogen: ' kreis=[KreisBogen] '
        Linie: ' linie=[Linie]
76    (imCADzuSehen?='imCADzuSehen' )? ';' ;
77  Kreis : Kreis_Mittelpunkt_Radius ;
78  Kreis_Mittelpunkt_Radius : 'Kreis aus Mittelpunkt und
       Radius ' 'Name: ' name=ID
79    'Zentrumspunkt: ' zentrum=[Punkt2D | FQN] 'Radius: '
        radius=Ausdruck (referenz?='-Hilfskreis- ')?
```

```
80    (imCADnichtZuSehen?='imCADnichtZuSehen ')? ';' ;
81  Bogen : Bogen_Mittelpunkt_Radius | Bogen_3_Punkte ;
82  Bogen_Mittelpunkt_Radius : 'Bogen aus Mittelpunkt und
       Radius ' 'Name: ' name=ID 'Zentrumspunkt: '
83    zentrum=[Punkt2D | FQN] 'Radius: ' radius=Ausdruck '
         Startwinkel: ' startwinkel=Ausdruck
84    'Endwinkel: ' endwinkel=Ausdruck (referenz?='-Hilfsbogen-
         ')? (imCADnichtZuSehen?='imCADnichtZuSehen ')? ';' ;
85  Bogen_3_Punkte : 'Bogen aus 3 Punkten ' 'Name: ' name=ID '
       Anfangspunkt: ' anfangspunkt=[Punkt2D | FQN]
86    'Endpunkt: ' endpunkt=[Punkt2D | FQN] 'PunktAufKurve: '
         punktAufKurve=[Punkt2D | FQN]
87    (referenz?='-Hilfsbogen- ')? (imCADnichtZuSehen?='
         imCADnichtZuSehen ')? ';' ;
88  Spiegeln_Skizze returns ElementSpiegelung : 'Spiegeln ' '
       Name: ' name=ID 'Spiegelachse: '
89    (spiegel=[Spiegelebene] | ebene=Ebenentyp ) '{ ' (
         elemente+=[Element])+ '} ' ;
90  ElementSpiegelung : (liste+=Element)+ ;
91  Beziehung: Beziehung_Parallel | Beziehung_Abstand |
       Beziehung_Senkrecht | Beziehung_Winkel |
92    Beziehung_Kollinear | Beziehung_Horizontal |
         Beziehung_Vertikal |
93    Beziehung_KreisBogenMittelpunkt_Auf_Linie;
94  Beziehung_Parallel : 'Beziehung: Parallel ' 'Linie1: '
       linie1=[Linie] 'Linie2: ' linie2=[Linie] ;
95  Beziehung_Abstand : 'Beziehung: Abstand ' 'Linie1: ' linie1
       =[Linie] 'Linie2: ' linie2=[Linie] 'Abstand: ' abstand=
       Ausdruck ;
96  Beziehung_Senkrecht : 'Beziehung: Senkrecht ' 'Linie1: '
       linie1=[Linie] 'Linie2: ' linie2=[Linie] ;
97  Beziehung_Winkel : 'Beziehung: Winkel ' 'Linie1: ' linie1=[
       Linie] 'Linie2: ' linie2=[Linie] 'Winkel: ' winkel=
       Ausdruck ;
98  Beziehung_Kollinear : 'Beziehung: Kollinear ' elemente=
       KollinearElement ;
```

```
99  KollinearElement : Kollinear_KreisBogen | Kollinear_Achse
       Linie ;
100 Kollinear_KreisBogen : 'Kreis / Bogen 1: ' kreis1=[
       KreisBogen] 'Kreis / Bogen 2: ' kreis2=[KreisBogen] ;
101 Kollinear_AchseLinie : 'Element1: ' (achseLinie1=[Achse
       Linie] | typ1=Achstyp)
102   'Element2: ' (achseLinie2=[AchseLinie] | typ2=Achstyp) ;
103 Beziehung_Vertikal : 'Beziehung: Vertikal ' 'Linie: ' linie
       =[Linie] ;
104 Beziehung_Horizontal : 'Beziehung: Horizontal ' 'Linie: '
       linie=[Linie] ;
105 Beziehung_KreisBogenMittelpunkt_Auf_Linie : 'Beziehung:
       Kreis- / Bogen-Mittelpunkt auf Linie '
106   'Kreis / Bogen: ' kreis=[KreisBogen] 'Linie: ' (
        achseLinie=[AchseLinie] | typ=Achstyp) ;
107 Option : Element_Loeschen ;
108 Element_Loeschen : 'Element(e) löschen: ' '{ ' (elemente+=[
       Element])+ '} ' ;
109 KreisBogen : Kreis | Bogen ;
110 AchseLinie : Achse | Linie ;
111 Spiegelebene : Ebene | Achse | Linie ;
112
113
114
115
116
117 Klammer : ( wurzel='sqrt(' | abs='abs(' | sin='sin(' | cos
       ='cos(' | tan='tan(' | klammer='(' ) ausdruck=Ausdruck
       ')' ;
118 Wert : ParameterDef | Variable ;
119 Factor : (vorzeichen='-')? (zahl=Zahl | wert=[Wert] |
       klammer=Klammer | pi='PI') ;
120 Term : factor_l=Factor ( zeichen+=( '*' | '/' ) factor_r+=
       Factor )* ;
121 Ausdruck : term_l=Term ( zeichen+=( '+' | '-' ) term_r+=
       Term )* ;
```

```
122 AusdruckRel : zeichen=( '+' | '-' | '*' | '/' ) '('
       ausdruck=Ausdruck ')' ;
123 FQN : ID ('.' ID)* ;
124 Zahl : INT ('.' INT )? ;
125
126 terminal INT returns ecore :: EInt : ( '0'..'9' )+ ;
127 terminal ID : ( 'a'..'z' | 'A'..'Z' | '_' ) ( 'a'..'z' | 'A
       '..'Z' | '_' | '0'..'9' )* ;
128 terminal STRING : '"' ( '\\' ( 'b' | 't' | 'n' | 'f' | 'r'
       | '"' | "'" | '\\' )  | ! ( '\\' | '"' ) ) * '"' |
129   "'" ( '\\' ( 'b' | 't' | 'n' | 'f' | 'r' | '"' | "'" |
         '\\' ) | ! ( '\\' | "'" ) ) * "'" ;
130 terminal ML_COMMENT : '/*' -> '*/' ;
131 terminal SL_COMMENT : '//' ! ( '\n' | '\r' ) * ('\r' ? '\n'
       ) ? ;
132 terminal WS : ( ' ' | '\t' | '\r' | '\n' ) + ;
133 terminal ANY_OTHER : . ;
```

Literaturverzeichnis

[Bug96] Bugow, Rainer: *Die Bereitstellung von Teilebibliotheken im rechnerunterstützten Konstruktionsprozeß.* 1. Auflage. Beuth Verlag GmbH, 1996 (DIN Deutsches Institut für Normung: DIN-Normungskunde)

[din90] *CAD-Normteiledatei nach DIN (DIN-Fachbericht 14).* 3. Auflage 1990

[Ecl] Eclipse Foundation: *Xtext User Guide.* `http://www.eclipse.org/Xtext/documentation/0_7_2/xtext.html`, Abruf: 03.05.2010

[GME] Gerhardt, Frank ; Moroff, Dieter ; Eberle, Stephan: *Das Eclipse Modeling Framework (EMF).* `http://www.java-forum-stuttgart.de/jfs/2006/folien/D2_Gerhardt_Moroff_Eberle.pdf`, Abruf: 15.04.2010

[Grä09] Gräfe, Konrad (Hrsg.): *Konzeption und prototypische Implementierung eines Programms zur Interpretierung von Merkmaldateien nach DIN V 4001 und Speicherung der Daten in einem leistungsfähigeren Format.* - 2009. - 61 S. Mittweida : Hochschule Mittweida (FH), Fakultät Mathematik/Physik/Informatik, Bachelorarbeit 2009

[Hof] Hoffmann, Christian: *ANTLR - Eine Einführung.* `http://www.c-hoffmann.de/WinPL/antlr/`, Abruf: 03.05.2010

[Inn] Innovations Report: *DIN Deutsches Institut für Normung.* `http://www.innovations-report.de/html/profile/profil-1198.html`, Abruf: 04.05.2010

[JAX] JAXenter: *Galileo-Interview: Textuelle DSLs entwickeln mit Xtext.* `http://www.jaxenter.de/koehnlein/`, Abruf: 28.04.2010

[Kun02] Kunze, Harald: *Ein Beitrag zur Gewinnung von Funktionen für mechanische Produkte aus 3D-CAD-Gestaltmodelldaten.* Aachen. Shaker Verlag, 2002

[KZ09a] Köhnlein, Jan ; Zarnekow, Sebastian: Domänenspezifische Sprachen mit Xtext: Teil 1: Einführung in Xtext. In: *Eclipse Magazin* 05 (2009), S. 67–71

[KZ09b] Köhnlein, Jan ; Zarnekow, Sebastian: Domänenspezifische Sprachen mit Xtext: Teil 2: Erweiterung und Ausführung von DSLs. In: *Eclipse Magazin* 06 (2009), S. 68–74

[KZ10] Köhnlein, Jan ; Zarnekow, Sebastian: Domänenspezifische Sprachen mit Xtext: Teil 3: Interpreter und fortgeschrittene Möglichkeiten in der Grammatik. In: *Eclipse Magazin* 01 (2010), S. 50–56

[nor12] *Norm ISO 13584-31 (Teilebibliothek) Teil 31.* Ausgabe 1999-12

[pli] *PLIB Home Page.* `http://www.plib.ensma.fr/`, Abruf: 03.05.2010

[SBPM09] Steinberg, Dave ; Budinsky, Frank ; Paternostro, Marcelo ; Merks, Ed: *EMF Eclipse Modeling Framework.* 3. Auflage - Boston. Addison-Wesley, 2009

[Sch] Schmidt, Maik: *Einführung in das Eclipse Modeling Framework (EMF).* `http://pi.informatik.uni-siegen.de/Mitarbeiter/mschmidt/lehre/SS09/ST2/emf.pdf`, Abruf: 15.04.2010

[Sen94] Sendler, Ulrich: *3D-CAD: Die Produktivität der neuen Systemgeneration.* Berlin Heidelberg. Springer-Verlag, 1994

[Sti09] Stitz, Manuel: *Entwicklung eines Verfahrens zur Speicherung, Verarbeitung und Übernahme von CAD-System-neutraler Geometrie und dazugehöriger Anreicherungen.* 2009. – (unveröffentlicht)

[Vog] Vogel, Lars: *Eclipse Modeling Framework (EMF) - Tutorial.* `http://www.vogella.de/articles/EclipseEMF/article.html`, Abruf: 03.05.2010

Selbständigkeitserklärung

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

Mittweida, den 3. August 2010