Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Computergestützte Modellierung und Simulation
Prof. Dr.-Ing. André Borrmann

Repräsentation und Detaillierung parametrischer Skizzen mit Hilfe von Graphersetzungssystemen

Simon Vilgertshofer

Masterthesis
für den Master of Science Studiengang Bauingenieurwesen

Autor:	Simon Vilgertshofer
Matrikelnummer:	[redacted]
Betreuer:	Prof. Dr.-Ing. André Borrmann

Ausgabedatum:	01. April 2014
Abgabedatum:	28. September 2014

Abstract

In the scope of planning and realizing large infrastructural projects it may be reasonable to create product models as multi-scale models in various levels of detail (LODs). Implementing this idea requires research to compose a representation of such a model where consistency is preserved, across the different LODs.

Previous research work in this area has shown that manually creating those models is a very complex, time consuming and error-prone task. Therefore research concerning fundamental analysis for the automation of correspondent detailing processes is necessary.

This thesis investigates the basic problem relevant for this approach. More precisely it analyzes how two-dimensional parametric geometric models (sketches) can be represented by graphs and how those sketches may be detailed through the use of graph transformation to alter their graph-based representations. The theoretical framework of this research comprises the theory of graphs, graph transformation and parametric modeling.

In the first place it is necessary to develop a general approach to represent sketches by the use of graphs and to survey the kind of limitations applying to such an approach. It is discussed how geometric elements and corresponding parametric constraints of a sketch can be depicted by the nodes and edges of a graph and their attributes.

To define the elements of the representing graph, a so called graph metamodel has to be developed. It describes the different types of nodes and edges, and the attributes they possess. Based on this metamodel a set of graph rewrite rules is predefined. Those rewrite rules represent steps in the detailing process of a sketch. By applying a rewrite rule to a graph-based representation of a sketch, it is possible to be altered in a way that the resulting graph depicts a detailed version of the particular sketch.

The application of the graph transformation is performed by the graph transformation tool GrGen.NET. It allows automatic graph rewrite operations and can export the graph data.

A developed software tool creates the possibility to interpret the graph data and to display the represented sketch in a parametric CAD design software (Autodesk Inventor).

To demonstrate the functional capability of the presented theories a prototypic implementation is presented. The developed functionalities are illustrated through the stepwise refinement of a sketch representing a shield tunnel section.

Zusammenfassung

Im Bereich der Infrastrukturplanung ist es sinnvoll, Produktmodelle in verschiedenen sogenannten Levels of Detail (LODs) also als mehrskaliges Modell darzustellen. Um dies zu ermöglichen, sind Überlegungen zum Aufbau von konsistenzwahrenden Repräsentationen dieser Modelle in allen LODs notwendig.
Die vorangegangene Forschungsarbeit in diesem Bereich hat gezeigt, dass das manuelle Erstellen konsistenzwahrender mehrskaliger Modelle äußerst komplex und fehleranfällig ist. Daher sollen in dieser Arbeit erste Untersuchungen zur Automatisierung der entsprechenden Detaillierungsvorgänge durchgeführt werden.
Diese Arbeit beschäftigt sich mit der für diesen Ansatz grundlegenden Fragestellung, wie zweidimensionale parametrische geometrische Modelle (Skizzen) durch Graphen und die Detaillierung dieser Modelle durch entsprechende Graphersetzungsregeln dargestellt werden können. Als Grundlage dient dabei die Theorie der Graphersetzungssysteme und der parametrischen Modellierung.
Es wird dafür zuerst erarbeitet, wie sich eine Skizze durch einen Graphen prinzipiell repräsentieren lässt und welche Einschränkungen dabei beachtet werden müssen. Dazu wird beschrieben, wie die geometrischen Elemente und die zugehörigen parametrischen Zwangsbedingungen der Skizze im Graph durch Knoten und Kanten sowie deren Attribute beschrieben werden können. Dies geschieht durch die Definition eines sogenannten Metamodells, dass die Menge der Knoten und Kanten aus den sich der tatsächliche Graph zusammensetzt, beschreibt. Auf Basis dieses Metamodells werden Graphersetzungsregeln vordefiniert, durch die sich der Graph aufbauen bzw. verändern lässt. Durch die Anwendung solcher Graphersetzungsregeln wird die Detaillierung einer Skizze in einzelnen Schritten in der graphbasierten Repräsentation durchgeführt.
Das Metamodell und die Graphersetzungsregeln werden anschließend mit der Graphersetzungssoftware GrGen.NET modelliert. So sollen die Graphersetzungsregeln automatisiert durchgeführt und der Graph in digitaler Form ausgegeben werden können. Durch die Entwicklung eines Software-Tools ist es möglich, die durch den erstellten Graphen repräsentierte Skizze in einer parametrischen Modellierungssoftware (Autodesk Inventor) anzeigen zu können.
Die Überlegungen werden durch die Erstellung eines Prototyps überprüft, der die Skizze eines Tunnelquerschnitts in einzelnen Schritten aufbaut bzw. detailliert.

Inhaltsverzeichnis

1 Einführung **1**
1.1 Problemstellung und Motivation 1
1.2 Ziel und erwartete Ergebnisse der Arbeit 2
1.3 Aufbau der Arbeit 4

2 Konzept **6**
2.1 Begriffserklärung 6
2.2 Der graphbasierte Ansatz 7
2.3 Problemstellungen 9
2.3.1 Repräsentation parametrischer Skizzen durch einen Graphen 9
2.3.2 Detaillierung der Skizze durch Transformation des Graphen 12
2.3.3 Darstellung in einem parametrischen Modellierer 13

3 Graphen **15**
3.1 Graphentheorie und Graphersetzungssyteme 15
3.1.1 Aufbau eines Graphen 16
3.1.2 Graphersetzungssysteme 19
3.2 Software zur Graphersetzung GrGen.Net 21
3.2.1 Überblick 21
3.2.2 Funktionsumfang 22
3.2.3 Aufbau eines Metamodells in GrGen.Net 23
3.2.4 Aufbau von Graphersetzungsregeln in GrGen.NET 24
3.2.5 Datenformate und -export 26

4 Parametrik **28**
4.1 Parametrische Modellierung 28
4.1.1 Dimensionale Zwangsbedingungen 29
4.1.2 Geometrische Zwangsbedingungen 29
4.2 Autodesk Inventor 30
4.2.1 Überblick 30
4.2.2 Parametrische Modellierung in Autodesk Inventor 30

4.2.3 Funktionen der API von Autodesk Inventor . . . 32

5 Beschreibung der methodischen Vorgehensweise 34
5.1 Überblick . . . 35
5.2 Analyse der Skizze . . . 37
5.3 Definition des Metamodells . . . 38
5.4 Graphersetzungsregeln . . . 39
5.5 Erstellung und Export des Graphen . . . 39
5.6 Einlesen des Graphen in Autodesk Inventor . . . 40
5.7 Überprüfung der Skizze . . . 41
5.8 Fazit . . . 41

6 Entwicklung des Graphersetzungssystems 42
6.1 Grundsätzliche Feststellungen . . . 43
6.1.1 Eigenschaften des Graphen . . . 43
6.1.2 Verwendung von Knoten und Kanten . . . 44
6.1.3 Andockstellen . . . 45
6.1.4 Temporäre Koordinaten . . . 45
6.1.5 Wiederverwendbarkeit von Graphersetzungsregeln . . . 47
6.1.6 Resultat in korrekter Repräsentation . . . 47
6.2 Konkrete Definition des Metamodells . . . 47
6.2.1 Geometrische Elemente . . . 48
6.2.2 Parametrische Zwangsbedingungen . . . 50
6.3 Definition von Graphersetzungsregeln . . . 54
6.3.1 Aufgabe der Graphersetzungsregeln . . . 54
6.3.2 Konzeption der Graphersetzungsregeln . . . 55
6.3.3 Beispielhafter Aufbau einer Graphersetzungsregel . . . 56
6.4 Vorgehen zur Erstellung der Skizze in Autodesk Inventor . . . 58
6.5 Zusammenfassung . . . 61

7 Prototypische Umsetzung 62
7.1 Programmierung des Graphersetzungssystems in GrGen.NET . . . 63
7.1.1 Metamodell . . . 63
7.1.2 Graphersetzungsregeln . . . 66
7.1.3 Ausführung der Graphersetzungsoperationen . . . 69
7.2 Entwicklung des G2I-Tools . . . 70
7.2.1 Funktionsumfang . . . 70
7.2.2 Programmiersprache und Bibliotheken . . . 71
7.2.3 Datenimport und Klassenstruktur . . . 71
7.2.4 Zugriff auf Autodesk Inventor . . . 72

7.2.5 Erstellung der geometrischen Elemente und der dimensionalen Zwangsbedingungen in Autodesk Inventor 72

8 Fazit und Ausblick **74**

8.1 Zusammenfassung der Ergebnisse 74

8.2 Ausblick 75

A Compact Disc **77**

Abbildungsverzeichnis

2.1 Repräsentation einer stufenweise aufgebauten Skizze durch einen Graph . . . 9
2.2 Durch parametrische Zwangsbedingungen repräsentierte Skizze dargestellt in Autodesk Inventor . . . 10
2.3 Veranschaulichung der Mehrdeutigkeit eines Systems parametrischer Zwangsbedingungen . . . 11
2.4 Aufbau bzw. Detaillierung einer Skizze . . . 12
2.5 Ablauf der Verarbeitung der Daten des Graphen . . . 14

3.1 Ein gerichteter Graph A und sein Subgraph S . . . 17
3.2 Grundlegendes Prinzip der Graphersetzung entsprechend dem SPO (Blomer *et al.*, 2013) . . . 20
3.3 Systemkomponenten von GrGen.NET (Blomer *et al.*, 2013)(Kroll, 2007) . . . 22
3.4 Visualisierung des durch eine Graph eXchange Language (GXL)-Datei beschriebenen Graphen . . . 27

4.1 Durch parametrische Zwangsbedingungen repräsentierte Skizze dargestellt in Autodesk Inventor . . . 31
4.2 Symbole der geometrischen Zwangbedingungen in Autodesk Inventor . . . 32
4.3 Ausschnitt aus dem *Object Model* der API von Autodesk Inventor 2014, entnommen aus dem Software Development Kit von Autodesk Inventor . . . 33

5.1 Iteratives Vorgehen zur Konzeption von Graphersetzungssystem und Visualisierung des Graphen . . . 36
5.2 Darstellung der Geometrie einer Skizze und ihrer parametrischen Zwangsbedingungen in Autodesk Inventor . . . 38
5.3 Visualisierung eines Graphen in yComp . . . 40

6.1 Form des Graphen bei alternativer Verwendung von Knoten und Kanten . . . 44
6.2 Hierarchische Darstellung der im Metamodell definierten Knoten . . . 53
6.3 Hierarchische Darstellung der im Metamodell definierten Kanten . . . 54
6.4 Vorgehen zur Definition einer Graphersetzungsregel . . . 55
6.5 Detaillierungsschritt . . . 56

6.6 Ersetzungsregel, durch die eine horizontale Linie, deren Endpunkte koinzident mit einem bereits bestehenden Kreisbogen sind, in eine Skizze eingefügt wird 57
6.7 Anwendung der in Abbildung 6.6 dargestellten Ersetzungsregel auf den Graphen 57
6.8 Unterschiedliche Ergebnisse der graphbasierten Repräsentation eines Dreiecks 60

7.1 Detaillierungsvorgang der Skizze, der in diesem Kapitel als Graphersetzungssystem umgesetzt wird 62
7.2 Ablaufschema des G2I-Tools 71

Abkürzungsverzeichnis

2D	Zweidimensional
3D	Dreidimensional
BIM	Building Information Modeling
XML	Extensible Markup Language
CAD	Computer Aided Design
SPO	Single Pushout Approach
LOD	Level of Detail
API	Application Programming Interface
GCS	Geometric Constraint Solver
GUI	grafische Benutzeroberfläche
GXL	Graph eXchange Language

Kapitel 1

Einführung

1.1 Problemstellung und Motivation

Umfangreiche Infrastrukturprojekte stellen die Gesellschaft, die Planer und die Bauindustrie seit jeher vor große Herausforderungen.

Das Aufeinandertreffen einer Vielzahl von Beteiligten unterschiedlicher Fachrichtungen, die nötige Integration der Projekte in bereits bestehende Systeme und die Akzeptanz durch die betroffene Bevölkerung spiegeln dabei nur einen Teil aller Problematiken, die Großprojekte mit sich bringen, wieder.
Denn im Gegensatz zu anderen Bauprojekten erstrecken sich die in der Planung zu erfassenden Bereiche, beispielsweise bei Bahntrassen oder Tunnels, über große Distanzen, die für eine effektive Planung sinnvoll abgebildet werden müssen. Hier ist es nötig, neben einer Übersicht über das gesamte Projekt, auch kleinste Details so zu modellieren und abzubilden, dass sie für den jeweiligen Nutzer gut lesbar sind.
Konkret bedeutet dies, dass die Darstellung Maßstäbe abbilden muss, die sich vom Kilometerbereich bis in den Zentimeterbereich erstrecken (Borrmann *et al.*, 2012) - dies kann bei der Erstellung von Plänen und Modellen mit dem Prinzip der sogenannten mehrskaligen Modellierung umgesetzt werden. Dabei liegt ein Modell in mehreren sogenannten Levels of Detail vor, die je nach Betrachtungsmaßstab zur Anzeige genutzt werden.

Durch die Idee und Entwicklung des Building Information Modeling (BIM) werden interdisziplinäre Planungsprozesse mit vielen Beteiligten immer besser unterstützt und die Bewältigung der oben genannten Herausforderungen zumindest teilweise vorangetrieben. Allerdings unterstützen die Modellierungstools, die im BIM-Bereich angewendet werden, die genannte mehrskalige Modellierung mit verschiedenen Level of Detail (LOD) bisher nur rudimentär oder überhaupt nicht, obwohl das zugrundeliegende Konzept im Bereich der Geoinformati-

onssysteme bereits etabliert ist und beispielsweise im Rahmen von CityGML angewandt wird (Kolbe, 2009).

Um dieses Prinzip mit den Methoden des BIM zusammenzuführen und dort umzusetzen sind Überlegungen, wie Geometrien in einem Produktmodell mehrskalig und dabei konsistent dargestellt werden können, unumgänglich. Eine generelle Methode zur Kombination von semantischen und geometrischen Aspekten wurde in (Borrmann *et al.*, 2012) bereits beschrieben und anhand der Entwicklung eines mehrskaligen Produktmodells für Tunnel beispielhaft ausgeführt. Dabei wurde eine formale Definition unterschiedlicher LOD und die Abbildung von Beziehungen zwischen geometrischen Elementen, die in verschiedenen LOD liegen, untersucht (Borrmann *et al.*, 2014) (Borrmann *et al.*, 2014).

Einen wesentlichen Bestandteil solcher Produktmodelle bildet ein geometrisches Modell, das es ermöglichen soll, das dargestellte Objekt in verschiedenen Detaillierungsstufen abzubilden. Die manuelle Erstellung solcher mehrskaliger Modelle hat sich, wie vorangegangene Forschungsarbeiten gezeigt haben, allerdings als komplex und fehleranfällig erwiesen. Zu untersuchen, wie solche Detaillierungsvorgänge automatisiert durchgeführt werden können, ist deshalb eine wesentliche Motivation für diese Arbeit.
Dafür sind Überlegungen zur Repräsentation des geometrischen Teils eines solchen mehrskaligen Produktmodells anzustellen. Hierfür ist es in einem ersten Schritt notwendig, zu untersuchen, wie ein parametrisches geometrisches Modell abgebildet und in einzelnen Schritten detailliert werden kann.

Der Ansatz dieser Thesis zur Automatisierung von Detaillierungsvorgängen besteht dabei darin, die Grundlagen der Graphentheorie mit dem Konzept der parametrischen Modellierung zu kombinieren.
Das Ziel dieser Arbeit, das im folgenden Abschnitt ausführlich formuliert wird, ist dementsprechend die Untersuchung von Möglichkeiten der graphbasierten Repräsentation parametrischer geometrischer Modelle und die Veränderung und Detaillierung dieser Modelle durch die Transformation des Graphen bzw. die Anwendung von Graphersetzungsregeln.

1.2 Ziel und erwartete Ergebnisse der Arbeit

In dieser Arbeit soll untersucht werden, wie sich ein zweidimensionales parametrisches geometrisches Modell, eine sogenannte Skizze, durch einen Graphen darstellen und durch die Transformation des Graphen erweitern oder verändern lässt. Damit soll perspektivisch eine Automatisierung von Detaillierungsvorgängen ermöglicht werden.

Die Grundlagen für diesen Ansatz bilden dabei die formale Graphentheorie und die dadurch gegebenen Möglichkeiten zur Graphentransformation sowie das Konzept der parametrischen

Modellierung. Dazu sollen diese beiden Themenbereiche beschrieben und die für die Problematik dieser Arbeit relevanten Inhalte dargestellt werden.

Der Schwerpunkt der Arbeit liegt darin, auf Basis dieser Grundlagen zu untersuchen, ob und wie die Anwendung von Graphen zur Repräsentation des parametrischen geometrischen Modells realisierbar ist. Dazu müssen die einzelnen Bestandteile eines solchen Modells herausgearbeitet und beschrieben werden. Anschließend ist festzulegen, welche Informationen in der graphbasierten Repräsentation enthalten sein müssen, um ein Modell bzw. seine einzelnen Komponenten eindeutig zu definieren.

Dabei soll das Modell soweit möglich vollständig parametrisiert sein und der repräsentierende Graph dementsprechend nur ein Minimum an fixen Koordinaten enthalten.
Weiterhin soll das repräsentierte Modell automatisiert in einer parametrischen CAD-Software dargestellt werden können. Dazu muss die Möglichkeit bestehen, die einzelnen Bestandteile des Graphen durch Zugriff auf die API der CAD-Software automatisiert zeichnen zu lassen.

Hieraus ergeben sich die folgenden Anforderungen an die graphbasierte Repräsentation:

- Der Graph muss die geometrischen Elemente der Skizze abbilden können.
- Der Graph muss die verschiedenen parametrischen Zwangsbedingungen, die die Beziehungen der geometrischen Elemente der Skizze zueinander definieren, beinhalten.
- Die so generierten Elemente des Graphen müssen alle nötigen Informationen beinhalten, um die repräsentierte Skizze in einer parametrischen CAD-Software darstellen zu können.
- Die Skizze soll dabei so weit wie möglich parametrisiert sein, um eine größtmögliche Flexibilität hinsichtlich der nachträglichen Veränderung von Parametern zu gewährleisten.

Es ist davon auszugehen, dass sich durch diese Vorgaben Einschränkungen und Problematiken bezüglich der graphbasierten Repräsentation ergeben. Diese sollen im Rahmen dieser Thesis herausgearbeitet und die dadurch entstehenden Konsequenzen für die praktische Nutzung der graphbasierten Repräsentation dargelegt werden.

Ein Vorteil, der sich durch diesen Ansatz der graphbasierten Repräsentation ergibt, ist die Veränderbarkeit des repräsentierenden Graphen durch Graphentransformation. Dazu sollen sogenannte Graphersetzungsregeln definiert werden, durch die es möglich ist, die graphbasierte Repräsentation einer Skizze automatisiert zu verändern und dadurch beispielsweise die Skizze zu detaillieren.
Dabei muss beachtet werden, dass durch die Anwendung einer Regel das repräsentierte Modell generell weiterhin eindeutig definiert bleibt und dass der veränderte Graph auch genau die gewünschte Veränderung in der Skizze hervorruft.

Dazu sollen zuerst die generellen Anforderungen, die sich an solche Ersetzungsregeln stellen, herausgearbeitet und auf dieser Basis beispielhafte Regeln erstellt werden. Prinzipiell ist dabei davon auszugehen, dass Graphersetzungsregeln für jedes Detaillierungs- oder Veränderungsszenario individuell zu formulieren sind.

Zur Überprüfung der Funktionalität der graphbasierten Repräsentation und der zugehörigen Ersetzungsregeln sollen diese als prototypisches Beispiel in der Graphersetzungssoftware GrGen.NET programmiert werden. Die thematische Grundlage für dieses Beispiel bildet dabei der vereinfachte Querschnitt eines Tunnelbauwerks.
Die Daten des durch GrGen.NET erstellten Graphen sollen anschließend exportiert werden, um aus ihnen die repräsentierte Skizze automatisiert in einer parametrischen CAD-Software darstellen zu können. Als parametrische CAD-Software soll dabei das Programm Inventor der Firma Autodesk zum Einsatz kommen.
Dazu müssen einerseits die durch GrGen.NET exportierten Daten auf ihre Struktur hin untersucht werden. Andererseits muss eine Möglichkeit zur Konvertierung dieser Daten in ein für das Application Programming Interface (API) der CAD-Software interpretierbares Format erarbeitet werden.

1.3 Aufbau der Arbeit

Zur Orientierung des Lesers wird der Aufbau der Arbeit im Folgenden kurz umrissen:

Kapitel 2 beschreibt den grundlegenden Ansatz dieser Arbeit. Auf dieser Basis werden dann die konkreten Problemstellungen, deren Lösung Ziel dieser Arbeit ist, formuliert.

Kapitel 3 und 4 geben einen Überblick über die theoretischen Grundlagen und Softwareprodukte, die in dieser Arbeit genutzt werden. Dabei wird zuerst auf die Themengebiete Graphentheorie und Graphersetzung eingegangen und die Funktionsweisen der Graphersetzungssoftware GrGen.NET beschrieben.
Analog werden anschließend die Prinzipien der parametrischen Modellierung und die praktische Anwendung in der parametrischen CAD-Software Autodesk Inventor umrissen. Dabei wird auf die Durchführung von parametrischen Modellierungsoperationen über die graphische Benutzeroberfläche der Software eingegangen.

Basierend auf den in den vorhergehenden Kapiteln dargelegten Problemstellungen und theoretischen Grundlagen wird in Kapitel 5 das methodische Vorgehen, welches zur Erstellung des Graphersetzungssystems genutzt wurde, beschrieben.

Anschließend wird in Kapitel 6 das erarbeitete Graphersetzungssystem zur graphbasierten Darstellung und Detaillierung von parametrischen Skizzen vorgestellt.
Hier werden zuerst die Bestandteile von Skizzen, die durch den Graphen repräsentiert werden

sollen, untersucht. Auf dieser Basis wird anschließend das Graphersetzungssystem in Form von Metamodell und Graphersetzungsregeln konzeptioniert. Dabei wird auch detailliert auf die Vorgehensweise und die Entscheidungen, die zu einer funktionsfähigen Lösung geführt haben, eingegangen.

In Kapitel 7 wird aufgeführt, wie das zuvor erarbeitete Konzept in GrGen.NET praktisch umgesetzt werden kann, um eine automatisierte Erstellung der graphbasierten Repräsentation einer Skizze zu ermöglichen. Dazu wird beschrieben, wie das in Kapitel 2 definierte Metamodell und die zugehörigen Graphersetzungsregeln in GrGen.NET implementiert werden können.
Anschließend wird auf die Entwicklung des Softwaretools G2I für Import und Visualisierung der durch den Graphen repräsentierten Daten in Autodesk Inventor eingegangen.

Zum Abschluss werden die wesentlichen Erkenntnisse dieser Arbeit zusammengefasst und aufgezeigt, welche Möglichkeiten sich hinsichtlich zukünftiger Forschungsarbeiten ergeben.

Kapitel 2

Konzept

2.1 Begriffserklärung

Zum besseren Verständnis des in diesem Kapitel vorgestellten generellen Konzepts bzw. der Problemstellungen, die sich durch dieses Konzept ergeben, ist es nötig, einige Begriffe einleitend zu definieren, obwohl dadurch den Kapiteln 3 und 4 vorgegriffen wird.
An dieser Stelle wird dabei nur ein kurzer Überblick gegeben - die ausführlichen Beschreibungen sind in den entsprechenden Kapiteln detailliert aufgeführt.

Skizze

Unter dem Begriff Skizze (in englischsprachigen Veröffentlichungen *Sketch*) wird in dieser Arbeit ein zweidimensionales parametrisches geometrisches Modell verstanden. Dies entspricht auch der Verwendung des Begriffs in Softwareprodukten zur parametrischen Modellierung wie Autodesk Inventor oder Siemens NX.
Im Gegensatz zu einer Zeichnung ist eine Skizze nicht durch die absoluten Positionen von geometrischen Elementen, die durch Koordinaten vorgegeben werden, definiert. Stattdessen sind die geometrischen Elemente durch sogenannte Zwangsbedingungen (engl. *Constraints*) miteinander verknüpft, um ihre Lage zueinander festzulegen.
Eine solche Skizze kann beispielsweise eine Zeichnung in einem CAD-System sein, die noch keine präzisen Informationen zu tatsächlichen Dimensionen beinhaltet. Stattdessen definiert sie nur eine grobe Anordnung von geometrischen Elementen wie Punkten, Linien und Kreisen. Erweitert wird diese Zeichnung dabei durch die genannten Zwangsbedingungen.
Den Prozess der Erstellung einer solchen Skizze bezeichnet man als parametrische Modellierung.

Graphersetzungssystem

Ein Graphersetzungssystem wird zur formalen Beschreibung der Veränderung von Graphen genutzt. Es setzt sich in erster Linie aus einer Menge von Graphersetzungsregeln zusammen, die dazu dienen, einen Teil eines Graphen durch einen neuen Graphen zu ersetzen.
Neben den Graphersetzungsregeln ist für den Aufbau eines Graphersetzungssystems meist auch ein sogenanntes Metamodell nötig. In diesem Metamodell werden die verschiedenen Arten von Knoten und Kanten, aus denen sich ein Graph zusammensetzen kann, definiert.
Um den Vorgang der Veränderung des Graphen durch eine Graphersetzungsoperation zu beschreiben, werden die Formulierungen „Tranformation des Graphen“ oder „Graphersetzung“ genutzt.

Graphersetzungssoftware

Die beschriebenen Graphersetzungssysteme werden oft mit Computerprogrammen implementiert, um eine automatische Anwendung der Graphersetzungsregeln auf einen Graph zu ermöglichen. In der Literatur werden auch diese Programme teilweise als Graphersetzungssystem bezeichnet, was zu unklaren Formulierungen führen kann.
In dieser Arbeit werden deshalb Programme, mit denen Graphersetzungssysteme modelliert und Transformationen von Graphen durchgeführt werden können, als Graphersetzungssoftware bezeichnet.

2.2 Der graphbasierte Ansatz

Der wesentliche Ansatz dieser Thesis besteht darin, ein zweidimensionales parametrisches geometrisches Modell durch einen Graphen zu repräsentieren.
Bevor weiter auf die konkreten Problemstellungen, die sich durch diesen Ansatz ergeben, eingegangen wird, soll zuerst die Motivation für diesen Ansatz erläutert werden.

Erkennt man die Parallelen zwischen den Möglichkeiten, die eine graphbasierte Repräsentation bietet und den Anforderungen, die parametrische geometrische Modelle an eine Methode zur Repräsentation stellen, so liegt es nahe, solche Skizzen durch einen Graphen zu repräsentieren.
Auch die in parametrischen Modellierungssystemen integrierten Geometric Constraint Solver (GCS) verwenden teilweise einen graphbasierten Ansatz, um die Zwangsbedingungen einer Skizze zu repräsentierten (Fudos & Hoffmann, 1997)(Fudos, 1995)(Jubierre, 2009).
Im Kontext der mathematischen Graphentheorie ist ein Graph in erster Linie nur eine netzartige Struktur abstrakter Art, die einerseits eine Menge von Objekten und andererseits die Zusammenhänge zwischen diesen Objekten abbilden kann. Zusätzlich können Graphen aber anhand formaler Regeln konsequent verändert werden und so Veränderungen der Objekte

und Zusammenhänge, die sie abbilden, darstellen.
Geometrische Modelle können als Strukturen verstanden werden, die sich aus einer Menge von einzelnen geometrischen Grundelementen (Punkte, Linien, Kreise, etc.) zusammensetzen und durch die Lage dieser Grundelemente ein tatsächliches Modell bilden. Die Lage dieser Grundelemente zueinander kann prinzipiell durch sogenannte parametrische Zwangsbedingungen genau und eindeutig definiert werden (Diese Methode wird bereits in der parametrischen CAD-Modellierung erfolgreich umgesetzt.). Ein geometrisches Modell kann also auch als eine Menge von Objekten, die in verschiedenartigen Beziehungen zueinander stehen, verstanden werden.
Insofern ist anzunehmen, dass die graphbasierte Repräsentation eines geometrischen Modells, dessen Topographie durch parametrische Zwangsbedingungen definiert wird, prinzipiell möglich ist.
Der tatsächliche Vorteil besteht nun darin, dass der Graph durch Graphersetzungsregeln automatisiert veränderbar ist. So könnten Arbeitsschritte, die eine Skizze verfeinern, durch Graphersetzungsregeln formal definiert und ausgeführt werden, ohne dass die eigentliche Skizze manuell bearbeitet werden muss. Für den Anwender ist es also nur nötig, die auszuführende Regeln auszuwählen - die tatsächliche Anwendung der Regeln, und die damit einhergehende Veränderung der Skizze, erfolgt automatisiert.

Bei der Umsetzung dieses Ansatzes liegt die erste wesentliche Entscheidung darin, festzulegen welche Elemente der Skizze im Graphen durch Knoten und welche Elemente durch Kanten repräsentiert werden. Da die Zwangsbedingungen die Beziehungen der geometrischen Elemente untereinander definieren, erscheint es sinnvoll, diese durch Kanten zu modellieren. Dementsprechend sollen die geometrischen Grundelemente mit den Knoten des Graphs repräsentiert werden. Diese grundlegende Entscheidung wird in Kapitel 6 detailliert begründet.

Zur Veranschaulichung des beschriebene Ansatzes ist in Abbildung 2.1 die beispielhafte Detaillierung einer Skizze und des zugehörigen (stark vereinfachten) Graphen dargestellt. Dabei liegt der Fokus nicht darin, schon eine tatsächlich funktionsfähige Repräsentation abzubilden. Vielmehr soll diese Abbildung die prinzipielle Idee der Methode visualisieren, um damit die folgenden Abschnitte einzuleiten. Insofern wurde auf die Benennung der Kanten und die Darstellung von parallelen Kanten und Schleifen verzichtet.
Auch im weiteren Verlauf der Arbeit wird die Entwicklung des Graphersetzungssystem anhand dieses Beispiels erläutert, wobei die Darstellung des Graphen dabei immer weiter verfeinert wird.

Auf Basis der erfolgreichen Umsetzung dieses Ansatzes ist es dann denkbar, weiter zu untersuchen, inwiefern sich die graphbasierte Repräsentation auch für die eingangs erwähnten Modelle eignet, die Geometrien in mehreren LOD abbilden, deren Objekte parametrisch voneinander abhängig sind und die bei Veränderungen LOD-übergreifend konsistent bleiben. Diese Überlegungen sind allerdings nicht Gegenstand dieser Arbeit.

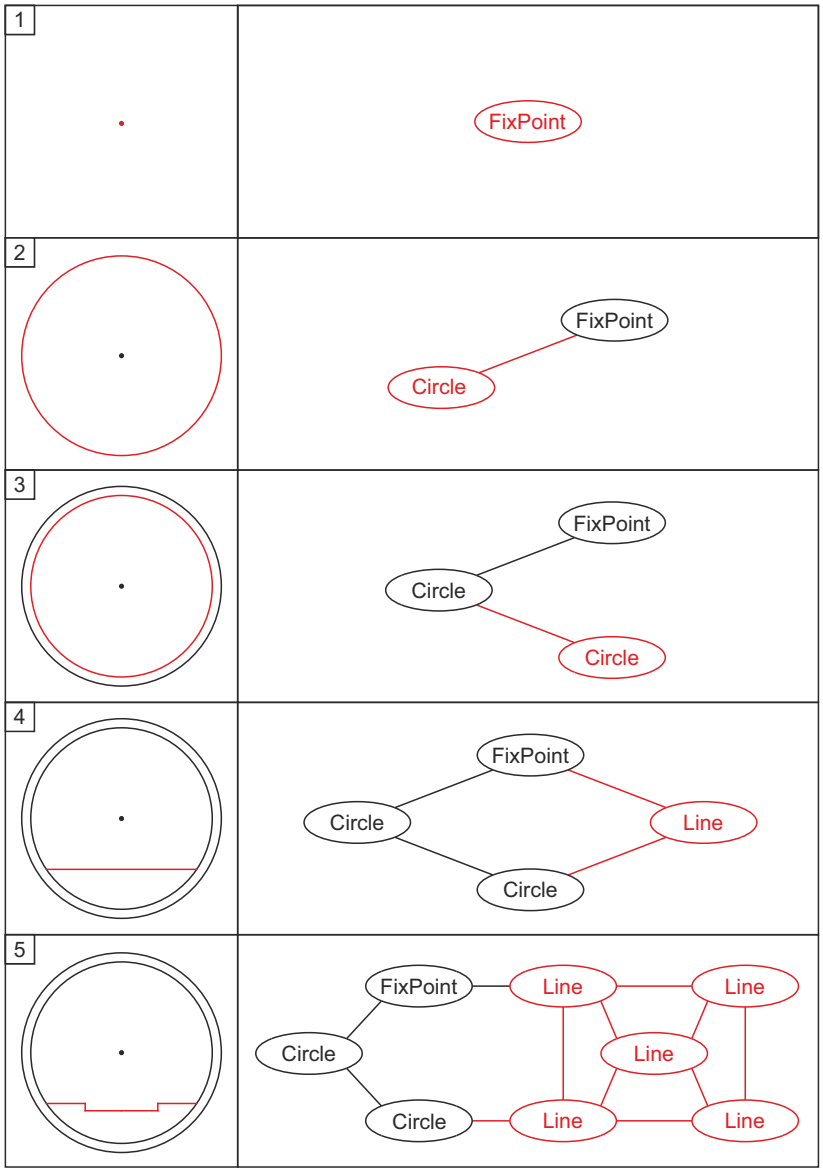

Abbildung 2.1: Repräsentation einer stufenweise aufgebauten Skizze durch einen Graph

2.3 Problemstellungen

Aus dem im vorigen Abschnitt beschriebenen Ansatz werden im Folgenden drei wesentliche Problemstellungen abgeleitet.

2.3.1 Repräsentation parametrischer Skizzen durch einen Graphen

Eine Skizze wurde im vorangegangenen Abschnitt als zweidimensionales parametrisches geometrisches Modell definiert.
Die graphbasierte Repräsentation einer solchen Skizze muss demnach sowohl das geometri-

sche Modell, als auch dessen parametrische Eigenschaften abbilden können. Daraus ergeben sich die im Folgenden beschriebenen grundlegenden Anforderungen.

Ein *geometrisches Modell* ist als eine mathematische Struktur zu verstehen, die sich aus Elementen wie Punkten, Geraden, Kreisen und deren räumlicher Lage zueinander zusammensetzt und dadurch eine bestimmte Anordnung dieser Elemente im Raum definieren kann.

Ein *parametrisches Modell* hat die Eigenschaft, dass die Lage der geometrischen Elemente zueinander durch funktionelle Abhängigkeiten definiert wird. Diese können im Nachhinein veränderbar sein. Dadurch ist es beispielsweise möglich, Abmessungen einzelner Elemente zu ändern, ohne dass die eigentliche Gestalt des Modells verlorengeht. Stattdessen werden andere Elemente des Modells durch die Parametrisierung automatisch entsprechend verändert. Dies bedingt allerdings, dass weitgehend auf absolute Koordinaten, die die Lage der geometrischen Elemente definieren, verzichtet werden sollte, um eine möglichst vollständige Parametrisierung zu ermöglichen.
In der folgenden Abbildung ist eine Skizze dargestellt, die vollständig durch parametrische Zwangsbedingungen definiert ist.

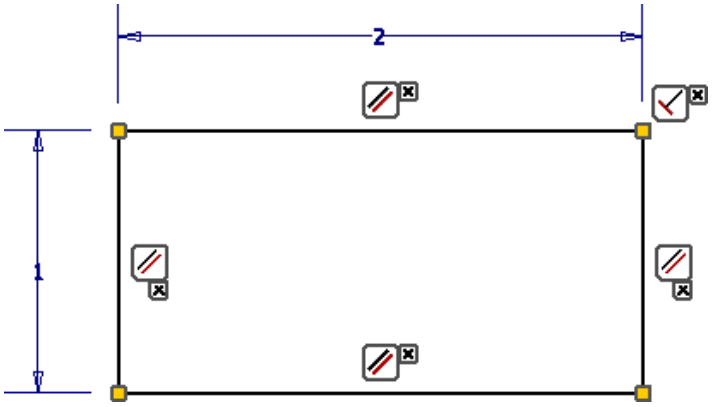

Abbildung 2.2: Durch parametrische Zwangsbedingungen repräsentierte Skizze dargestellt in Autodesk Inventor

Es müssen also im Graph einerseits die geometrischen Elemente und andererseits deren parametrische Beziehungen, also ihre Lage zueinander, repräsentiert sein.
Die Daten des Graphen, der eine Skizze anhand eines Systems aus geometrischen Elementen und parametrischen Zwangsbedingungen repräsentiert, sollten dabei eindeutig sein. Es ist also wünschenswert, dass ein bestimmter Graph nur genau eine mögliche Skizze darstellt und bei der Interpretation des Graphen keine unterschiedlichen oder falschen Ergebnisse entstehen können. Anzumerken ist dabei, dass es andererseits durchaus denkbar ist, dass ein und dasselbe geometrische Modell durch verschiedene Graphen, die nicht isomorph sind, repräsentiert wird.

Es ist abzusehen, dass die Erfüllung dieser Anforderungen an eine eindeutige Repräsentation spätestens beim Import des Graphen in eine parametrische CAD-Software problematisch

wird. Da die Anordnung der einzelnen geometrischen Elemente nicht über Koordinaten, sondern über parametrische Bedingungen definiert werden sollen, muss die CAD-Software die Skizze anhand der vorhandenen Elemente und Zwangsbedingungen aufbauen. Dies geschieht durch einen in der CAD-Software implementierten GCS (Fudos, 1995)(Jubierre, 2009). Ein solcher GCS interpretiert die Menge der parametrischen Zwangsbedingungen und generiert daraus eine Lösung, die alle Anforderungen, die sich durch die Zwangsbedingungen ergeben, erfüllt. Durch fehlende Informationen oder Mehrdeutigkeiten der Zwangsbedingungen sind allerdings in vielen Fällen verschiedene Lösungen möglich.
Wird beispielsweise ein Punkt P koinzident mit dem Kreisbogen eines Kreises mit Radius r gesetzt und zusätzlich ein vertikaler Abstand $d \in \{x \mid 0 < x < r\}$ zwischen Kreismittelpunkt M und P definiert, ergeben sich vier verschiedene Lösungsmöglichkeiten. Diese sind zur Veranschaulichung in folgender Abbildung 2.3 abgebildet:

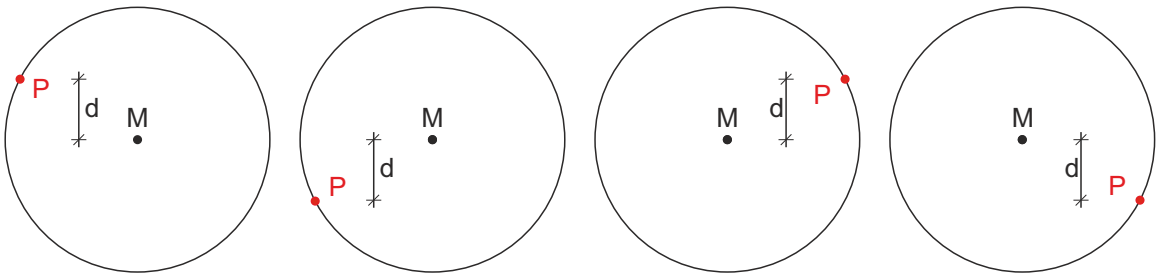

Abbildung 2.3: Veranschaulichung der Mehrdeutigkeit eines Systems parametrischer Zwangsbedingungen

Dementsprechend sollte die graphbasierte Repräsentation so viele Informationen wie möglich beinhalten, die das tatsächliche Aussehen der Skizze, also der Anordnung der geometrischen Elemente, festlegen. Dies ist allerdings teilweise nur durch Koordinaten, die die relative Position von Elementen zueinander beschreiben und nicht ausschließlich durch parametrische Bedingungen umsetzbar.
Dementsprechend steht diese Anforderung an die Eindeutigkeit im Gegensatz zum oben beschriebenen Anspruch nach einer möglichst vollständigen Parametrisierung.

Unter diesen Voraussetzungen werden folgende Anforderungen an den Graphen gestellt:

- Der Graph muss geometrische Elemente unterschiedlicher Art repräsentieren können.
- Der Graph muss verschiedene parametrische Zwangsbedingungen beinhalten können.
- Die repräsentierte Skizze soll nach Möglichkeit vollständig parametrisiert sein und nur die nötigsten absoluten Koordinaten enthalten.
- Bei einer Interpretation des Graphen durch einen GCS soll nur eine Lösung existieren.

Es ist hier sofort ersichtlich, dass sich aus den beiden letzten Punkten ein Konflikt hinsichtlich des Aufbaus des Graphen ergeben wird. Ein weiterer Anspruch an die graphbasierte

Repräsentation ist es dementsprechend, einen Kompromiss zu finden, der diesen Konflikt möglichst sinnvoll löst.

2.3.2 Detaillierung der Skizze durch Transformation des Graphen

Prinzipiell ist die Detaillierung bzw. der Aufbau einer Skizze als ein Vorgang zu verstehen, bei dem aus einer zunächst primitiven Abbildung durch das Einfügen oder Ersetzen von Elementen eine immer komplexere Darstellung entsteht. Dieser Vorgang erfolgt in einzelnen Schritten.
In der folgenden Abbildung 2.4 ist der Aufbau einer vereinfachten Skizze eines Tunnelquerschnittes beispielhaft dargestellt. Dabei sind nur die geometrischen Elemente ohne parametrische Bedingungen visualisiert.

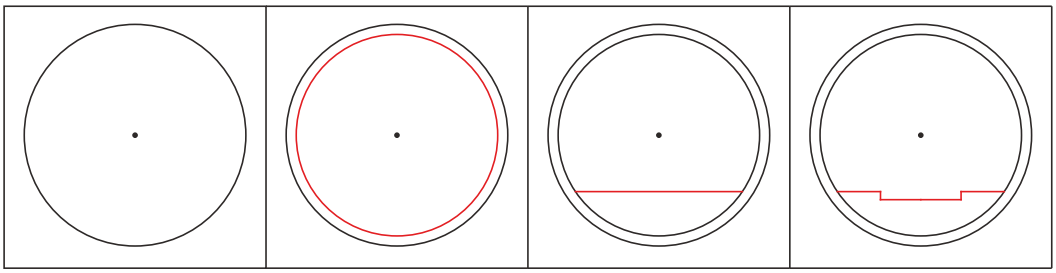

Abbildung 2.4: Aufbau bzw. Detaillierung einer Skizze

Da die Skizze durch die Interpretation des zugrunde liegenden Graphen erzeugt werden soll, muss sich dieser Vorgang auch im repräsentierenden Graphen widerspiegeln. Dementsprechend wird dieser mit der Durchführung jedes Schrittes entsprechend komplexer und umfangreicher.

Hierbei muss zuerst untersucht werden, welche Methoden der Graphersetzung für solche Operationen sinnvoll sind und wie diese definiert werden können, um die gewünschten Ergebnisse im resultierenden Graphen und in der daraus resultierenden Skizze zu erzielen.

Auf Basis dieser Methoden zur Graphersetzung muss ein Graphersetzungssystem aufgebaut werden, in dem alle gewünschten Detaillierungsschritte durch Graphersetzungsregeln vordefiniert sind.
Die nötigen Graphersetzungsregeln richten sich dabei jeweils nach der konkreten Skizze, die erstellt werden soll und es ist davon auszugehen, dass sie für jedes Szenario individuell erstellt werden müssen.
Für eine automatisierte Ausführung solcher Regeln müssen diese als Teil eines Graphersetzungssystems in einer Graphersetzungssoftware implementiert werden.

Bevor allerdings konkrete Graphersetzungsregeln definiert werden können, müssen allgemeine Ansprüche an solche Regeln herausgearbeitet werden. Nur so kann davon ausgegangen werden, dass die Resultate, die sich aus der Anwendung der Regeln ergeben, konform zu den im vorigen Abschnitt definierten Ansprüchen an die graphbasierte Repräsentation sind.
Erst dann ist es möglich, konkrete Graphersetzungsregeln zu definieren, um beispielsweise den in Abbildung 2.4 dargestellten Detaillierungsvorgang zu repräsentieren.

Zusammengesfasst ergeben sich die folgenden grundlegenden Fragestellungen:

- Welche Methoden der Graphersetzung sind sinnvoll, um die Detaillierung von Skizzen zu repräsentieren?
- Welchen Ansprüchen müssen die Graphersetzungsregeln genügen, um eine eindeutige Skizze zu erzeugen?
- Welche Informationen müssen die Graphersetzungsregeln beinhalten, um einen konkreten Detaillierungsvorgang abzubilden?

Abschließend ist festzuhalten, dass sich die Problemstellungen zu Repräsentation und Veränderung, die in diesem und dem vorigen Abschnitt ausgearbeitet wurden, gegenseitig bedingen. Dementsprechend muss bei der Lösungsfindung immer überprüft werden, ob Ergebnisse konform zu den Ansprüchen aus beiden Problemstellungen sind. Dabei ist allerdings vorerst davon auszugehen, dass sich die Definition der Regeln den Ansprüchen, die sich an die generelle graphbasierte Repräsentation stellen, unterordnen muss.

2.3.3 Darstellung in einem parametrischen Modellierer

In der Einleitung und in Abschnitt 2.3.1 wurde bereits beschrieben, dass es möglich sein soll, die durch den Graph beschriebene Skizze in einem parametrischen CAD-System tatsächlich zu erzeugen, um damit weitere Konstruktionsarbeiten zu ermöglichen. Es ist beispielsweise denkbar die Skizze zu nutzen, um durch Extrusionen dreidimensionale Modelle zu erzeugen. Innerhalb des CAD-Systems kann die erzeugte Skizze anhand von schon im Graphen definierten Parametern in ihren Abmessungen verändert werden, wie es auch bei einer manuell erstellten Skizze der Fall wäre.
Das Erstellen der Skizze soll dabei automatisch und nicht per Eingabe über die grafische Benutzeroberfläche (GUI) erfolgen. Dementsprechend muss auf die API der CAD-Software zugegriffen werden, um eine automatisierte Verarbeitung der graphbasierten Daten zu ermöglichen.

Die Daten des durch die Graphersetzungssoftware GrGen.NET automatisiert erzeugten Graphen müssen demnach zuerst aus der entsprechenden Software exportiert werden. An-

schließend sind sie hinsichtlich ihrer Struktur zu analysieren, um anschließend eine automatisierte Weiterverarbeitung durch die API zu ermöglichen.
Dazu ist es notwendig, ein externes Programm zu entwickeln, das diesen Vorgang durchführt.

Der generelle Ablauf der Verarbeitung der Daten des Graphen und die Funktionsweise des Programms ist im Folgenden schematisch und in Abbildung 2.5 graphisch dargestellt.

- Export der Daten des erstellten Graphen aus der Graphersetzungssoftware GrGen.NET in einem von diesem Programm unterstützen Datenformat.
- Einlesen der exportierten Daten in das zu erstellenden Programm.
- Umwandlung der Daten innerhalb des Programms in eine für die weitere Verarbeitung sinnvolle Struktur.
- Zugriff des Programms auf eine laufende Instanz der CAD-Software durch die entsprechende API.
- Aufruf geeigneter Methoden der API, die die geometrischen und parametrischen Elemente als Skizze aufbauen.
- Anzeigen der fertiggestellten Skizze in der CAD-Software.

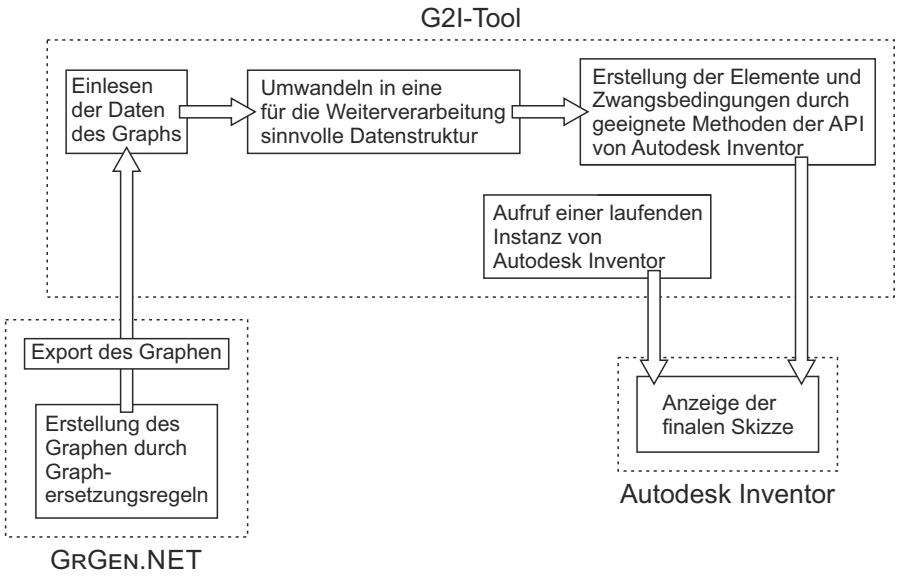

Abbildung 2.5: Ablauf der Verarbeitung der Daten des Graphen

Bei dieser Vorgehensweise muss berücksichtigt werden, welche Funktionen die API überhaupt unterstützt und wie eine Menge von geometrischen Elementen und parametrischen Bedingungen durch den GCS der Software interpretiert werden. Hierdurch haben sich im Laufe der Konzeption der Arbeit weitere Anforderungen an die Informationen, die der Graph enthalten muss, ergeben.

Kapitel 3

Graphen

Die in dieser Thesis verfolgten Ziele machen es erforderlich, Ansätze und Methoden aus verschiedenen wissenschaftlichen Bereichen zu untersuchen. Die beiden folgenden Kapitel 3 und 4 erörtern die genutzten theoretischen Grundlagen und ordnen sie hinsichtlich ihrer Relevanz für die Lösung der in Abschnitt 2.3 beschriebenen Problemstellungen ein.

Im folgenden Kapitel wird dabei zuerst auf die mathematischen und formalen Grundlagen der Theorie von Graphen und Graphersetzungssystemen und anschließend auf den Aufbau und den Funktionsumfang der Graphersetzungssoftware GrGen.NET eingegangen.

3.1 Graphentheorie und Graphersetzungssyteme

In dieser Arbeit sollen Graphen zur Repräsentation von Skizzen beziehungsweise Graphersetzungssysteme zur Veränderung dieser Repräsentationen und damit zur Veränderung der Skizzen genutzt werden.

Die Graphentheorie bildet dabei die formale Grundlage, auf deren Basis diese Repräsentationen erstellt werden. Die Veränderung von Graphen erfolgt mit sogenannten regelbasierten Graphersetzungssystemen. In diesem Abschnitt werden die grundlegenden Aspekte dieser beiden Themengebiete zusammengefasst.
Dabei wird zuerst die Graphentheorie als abstraktes Teilgebiet der Mathematik vorgestellt und darauf aufbauend der Zusammenhang mit konkreteren Anwendungsmöglichkeiten hergestellt. Anschließend wird auf den Aufbau eines Graphen und auf Techniken zur Veränderung bzw. Transformation eines Graphen eingegangen, um so das Konzept von Graphersetzungssystemen zu beschreiben.

3.1.1 Aufbau eines Graphen

Der Begriff Graphentheorie beschreibt ein Teilgebiet der Mathematik, das sich mit der Untersuchung der rein strukturellen Fragen von Netzstrukturen beschäftigt (Tittmann, 2003). Diese Strukturen werden als Graph bezeichnet und stellen eine Menge von Objekten und deren Beziehungen zueinander dar. Die Objekte werden Knoten genannt. Die in der Regel paarweisen Beziehungen zwischen diesen Knoten heißen Kanten.

Diesen soweit noch sehr abstrakten Modellen sind in der rein mathematischen Definition keine Informationen über die genaue Art oder Beschaffenheit der Knoten und Kanten zugeordnet. Es ist allerdings trotz dieses sehr hohen Abstraktionsgrades möglich, diverse Eigenschaften eines Graphen zu untersuchen, die sich nur aus der Anordnung seiner Knoten und Kanten ableiten lassen.
Als Beispiel seien dabei die Suche nach dem kürzesten Weg zwischen zwei Knoten oder die Frage nach der Menge verschiedener Graphen mit der selben Anzahl von Knoten und Kanten genannt.

Je nach Anwendungsgebiet ist es jedoch auch möglich und oft sinnvoll, die Bestandteile eines Graphen weniger abstrakt zu betrachten. So können einerseits verschiedene Klassen von Knoten und Kanten definiert werden, andererseits können diesen Knoten und Kanten unterschiedliche Eigenschaften, sogenannte Attribute, zugeordnet werden. Durch diese Konkretisierung können Szenarien, die sich durch Graphen darstellen lassen, die Realität wesentlich besser abbilden als rein abstrakte Modelle.

Bevor weiter auf konkretere Beispiele zu weniger abstrakten Formen von Graphen eingegangen wird, sind einige rein mathematische Definitionen notwendig. Diese basieren im wesentlichen auf (Tittmann, 2003) und (Diestel, 1996).

Mathematische Definition und Begriffserklärung

Ein Graph $G = (V, E)$ setzt sich aus einer Menge von Knoten (engl. *vertex*) V und einer Menge von Kanten (engl. *edge*) E zusammen. Jeder Kante $e \in E$ von G sind dabei zwei Knoten aus V zugeordnet. Die Anzahl der Knoten wird mit n und die Anzahl der Kanten mit m bezeichnet.

Eine Kante wird mit $e = \{u, v\}$ beschrieben, wobei u und v die Endknoten dieser Kante sind. In einem ungerichteten Graph hat eine Kante keine eindeutige Richtung, es gilt also $e = \{u, v\} = \{v, u\}$. Ist v ein Endknoten der Kante e, ist v inzident zu e. Knoten, die durch eine Kante verbunden sind, heißen adjazent oder benachbart.
Legt man eindeutig fest, dass jede Kante eines Graphen einen Anfangs- und einen Endknoten hat, nennt man diesen Graph gerichtet. In einem gerichteten Graph werden die Kanten auch Bögen genannt und im Gegensatz zu einem gerichteten Graphen mit $e = (u, v)$ bezeichnet,

wobei u der Anfangs- und v der Endknoten ist.
Haben zwei Kanten $a = \{u, v\}$ und $b = \{u, v\}$ die gleichen Endknoten spricht man von Mehrfach- oder Multikanten. Die Kanten a und b werden als parallele Kanten bezeichnet.
Hat eine Kante $a = \{u, u\}$ den selben Knoten als Anfangs- und als Endknoten, wird diese Kante Schlinge oder Schleife genannt.

Ein Graph, der keine Mehrfachkanten und keine Schleifen beinhaltet, heißt einfacher oder schlichter Graph. Multigraphen können im Gegensatz dazu parallele Kanten und Schleifen enthalten.
Eine weitere spezielle Variante von Graphen sind sogenannte Hypergraphen. Diese können Hyperkanten enthalten, die nicht nur zwei sondern beliebig viele Knoten verbinden können.

Einen Graphen $S = (V_S, E_S)$, der Bestandteil eines anderen Graphen $A = (V_A, E_A)$ ist bezeichnet man als Teil- oder Subgraphen. S ist hier ein Subgraph von A, während A der Ober - oder Supergraph von S genannt wird. Formal ausgedrückt ist dies der Fall, wenn V_S eine Teilmenge von V_A und E_S eine Teilmenge von E_A ist, dh. $V_S \subseteq V_A$ und $E_S \subseteq E_A$ gilt.
Anschaulich formuliert bedeutet dies, dass der Subgraph einen Teil (oder alle) der Knoten des Supergraphen und einen Teil (oder auch alle) der Kanten, die diese Knoten im Supergraphen verbinden, besitzt.

Zur Verdeutlichung dieser Zusammenhänge ist in folgender Abbildung ein Graph A und sein Subgraph S visuell dargestellt.

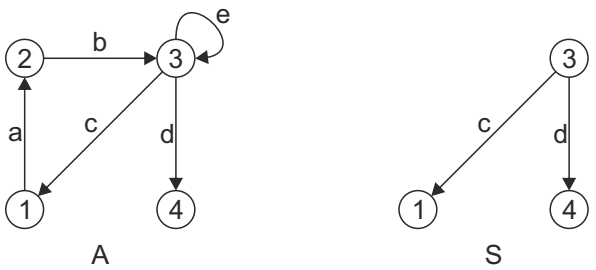

Abbildung 3.1: Ein gerichteter Graph A und sein Subgraph S

Die formale Definition der selben Graphen anhand der schon genutzten Notationen lautet für dieses Beispiel wie folgt:

$$A = (\{1, 2, 3, 4\}, \{a, b, c, d, e\}) \qquad S = (\{1, 3, 4\}, \{c, d\})$$

mit

$$a = (1, 2), \; b = (2, 3), \; c = (3, 1), \; d = (3, 4), \; e = (3, 3)$$

Klassen und Attribute

Soll mit einem Graphen eine reale Netzstruktur dargestellt werden, reichen oft abstrakte Knoten und Kanten nicht mehr aus, um eine eindeutige und sinnvolle Repräsentation zu ermöglichen.
Es ist daher sinnvoll, Knoten bzw. Kanten nicht nur als anonyme Bestandteile eines Graphen zu verstehen, die sich nur durch ihre Position im Graph auszeichnen. Wie bereits zu Beginn des Kapitels erwähnt, ist es daher möglich, den Knoten und Kanten eines Graphen verschiedene Klassen und Attribute zuzuweisen (Kniemeyer, 2008).

Vom Prinzip her werden dabei, ähnlich wie in der objektorientierten Programmierung, unterschiedliche Klassen von Knoten und Kanten definiert, aus deren jeweiligen Instanzen sich der Graph aufbaut.
Neben unterschiedlichen Arten von Knoten und Kanten definieren diese Klassen gegebenenfalls auch sogenannte Attribute, also Eigenschaften, die Knoten oder Kanten einer Klasse gemeinsam haben. Diese Attribute kann man als Variablen verstehen, die zwar in ihrer Art jeder Instanz einer Klasse gemein sind, sich aber hinsichtlich ihres tatsächlichen Wertes unterscheiden können.

Metamodell

Zur Definition der im vorangegangen Abschnitt beschriebenen Klassen und Attribute, über die ein Graph verfügen kann, ist die Erstellung eines sogenannten Metamodells nötig. In ISO 11179 wird ein Metamodell als *„data model that specifies one or more other data models“* (Datenmodell, das ein oder mehrere andere Datenmodelle definiert) beschrieben.
Im Metamodell wird festgelegt, aus welchen Typen von Knoten und Kanten ein auf diesem Metamodell basierender Graph zusammengesetzt sein kann und über welche Attribute seine Knoten und Kanten verfügen.
Ein Graphersetzungssystem benötigt eine solche Definition, die alle Knoten und Kanten mit einschließt, die während des Ersetzungsprozesses genutzt werden können, um angewendet zu werden (Helms, 2013).

Zum besseren Verständnis dieses Konzepts wird auf die Darstellung des im Rahmen dieser Arbeit erstellten Metamodells verwiesen, das in Kapitel 6 in den Abbildungen 6.2 und 6.3 auf Seite 53 dargestellt ist.

3.1.2 Graphersetzungssysteme

Neben der reinen Repräsentation netzartiger Strukturen in Natur und Technik durch Graphen ist es in vielen Fällen auch nötig, Veränderungen an Graphen vorzunehmen. Hierdurch kann die Veränderung solcher Strukturen modelliert bzw. repräsentiert werden.

Um solche Veränderungen formal zu beschreiben nutzt man Graphersetzungssysteme.
Graphersetzungssysteme sind eine Menge von Graphersetzungsregeln. Wird eine solche Regel auf einen Graph angewendet, ist es somit möglich, dem Graphen neue Knoten und Kanten hinzuzufügen oder bereits bestehende Knoten und Kanten zu entfernen. Weiterhin können die Attribute von Knoten und Kanten verändern werden.

Im Folgenden werden die für diese Arbeit relevanten Aspekte von Graphersetzungssystemen und Graphersetzungsregeln beschrieben.
Den Graphen, der verändert wird, nennt man dabei Ausgangs- oder Arbeitsgraph, während das Ergebnis der Graphersetzungsoperation als Ziel- oder als modifizierter Arbeitsgraph bezeichnet wird. Diese Begriffe werden auch in dieser Arbeit verwendet.
In englischsprachigen Veröffentlichungen werden meist die Begriffe *host graph*, *source- state* oder *pre-state* für den Ausgangsgraphen und *result graph*, *target state* oder *post-state* für den Zielgraphen genutzt. (Helms, 2013), (Heckel, 2006), (Hoisl, 2012), (Rozenberg, 1997)

Graphersetzung und Graphersetzungsregeln

Mit Hilfe von Graphersetzungsregeln können Veränderungen von Graphen (die hier als Graphersetzung oder Graphtransformation bezeichnet werden) generalisiert werden (Heckel, 2006). Um solche Regeln zu definieren, muss die zu generalisierende Veränderung anhand von Ausgangs- und Zielzustand des Graphen untersucht werden. Dazu wird einerseits der für die Ersetzung relevante Teil des Graphen im Ausgangszustand (ein Subgraph des Ausgangsgraphen) und andererseits seine Veränderungen im Zielzustand untersucht.
Anschließend kann der Unterschied bzw. die Veränderung zwischen diesen Graphen betrachtet werden.

Eine solche Produktion, die einen Teil des Ausgangsgraphen durch einen neuen Subgraphen ersetzt bzw. diesen Subgraphen an den Ausgangsgraphen anhängt, wird als Graphersetzungsregel bezeichnet.
Eine solche Graphersetzungsregel besteht dabei aus einer linken Seite L (dem *pattern graph* oder Mustergraphen) und einer rechten Seite R (dem *rewrite graph* oder Ersetzungsgraphen).

Der generelle Ablauf einer Graphersetzungsoperation kann nach (Heckel, 2006) vereinfacht in den folgenden drei Schritten beschrieben werden:

1. Der Arbeitsgraph H wird nach dem Vorkommen des Mustergraphen L der Graphersetzungsregel durchsucht.

2. Es werden alle Knoten und Kanten in H, die mit L übereinstimmen und nicht im R enthalten sind (formal $L \backslash R$), gelöscht. Das Ergebnis dieser Operation wird mit H^- bezeichnet.

3. Alle Knoten und Kanten aus R, die nicht in L enthalten sind (formal $R \backslash L$), werden H^- hinzugefügt. Es entsteht der modifizierte Arbeitsgraph H' als Ergebnis der Graphtransformation.

Im Folgenden wird der sogenannte Single Pushout Approach (SPO) weiter betrachtet. Dieser Vorgang ist in der folgenden Abbildung 3.2 schematisch dargestellt.
Weitere Ansätze werden am Endes dieses Abschnitts kurz erwähnt.

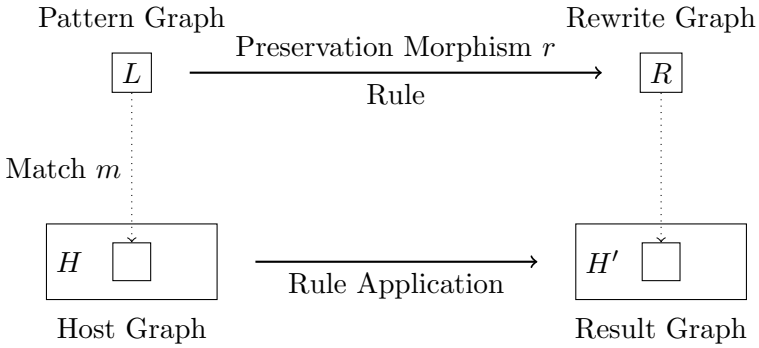

Abbildung 3.2: Grundlegendes Prinzip der Graphersetzung entsprechend dem SPO (Blomer *et al.*, 2013)

In dieser Abbildung ist zusätzlich ein *preservation morphism* r zur Einbettung des Ersetzungsgraphen in den Ausgangsgraphen enthalten. Dieser identifiziert die Elemente des Graphen (Knoten oder Kanten) von L und R, die während des Ersetzungsvorgangs erhalten bleiben sollen.
Auch hier wird im Ausgangsgraphen H nach einem Vorkommen des Mustergraphen G gesucht. Dieses Vorkommen wird als m (*match*) bezeichnet und ist mathematisch gesehen ein Isomorphismus (ein strukturell gleicher Graph) zwischen L und einem Subgraphen von H. Anschließend wird dieses Vorkommen $m(L)$ im Ausgangsgraphen so verändert, dass ein isomorpher Subgraph des Ersetzungsgraphen R entsteht. Bei dieser Veränderung werden Elemente aus L, die in r nicht definiert sind, aus $m(L)$ gelöscht, während Elemente aus R, die nicht in r enthalten sind, H hinzugefügt werden. Alle weiteren Elemente in r werden beibehalten. Dadurch entsteht der Zielgraph H' (Blomer *et al.*, 2013).

Die Graphersetzungssoftware GrGen.NET, die in Abschnitt 3.2 vorgestellt und für die praktische Umsetzung des in dieser Thesis erarbeiteten Konzepts genutzt wird, beruht auf dem SPO. Deshalb werden weitere Ansätze zur Graphersetzung hier nur der Vollständigkeit

halber erwähnt.
Es existieren beispielsweise noch der *Double Pushout Approach* (DPO), der *Node Label Controlled* Mechanismus (NLC) oder dessen erweiterte Version, der sogenannte *edNLC*. Detaillierte Informationen zu den genannten Ansätzen sind in (Rozenberg, 1997) zu finden.

3.2 Software zur Graphersetzung GrGen.Net

Der in dieser Arbeit verfolgte Ansatz zur Erstellung und Veränderung von Graphen soll von einer Software zur Graphersetzung übernommen werden.

Von der Eigenentwicklung einer solchen Software wurde von vornherein abgesehen und stattdessen eine bereits bestehenden Graphersetzungssoftware verwendet.
Unter den frei verfügbaren Softwarelösungen wurde hier GrGen.NET ausgewählt, da dieses alle nötigen Funktionen unterstützt und durch sein großes Spektrum an Funktionen außerdem Möglichkeiten zur Weiterentwicklung des in dieser Arbeit vorgestellten Konzepts bietet.
Im Folgenden Abschnitt wird der Funktionsumfang der Software kurz umrissen und anschließend genauer auf die für diese Arbeit integralen Bestandteile eingegangen.

Im Wesentlichen wurde zur Erstellung dieses Abschnitts die Dokumentation (Blomer *et al.*, 2013) des Programms herangezogen. Da die Funktionsweise von GrGen.NET in dieser Arbeit nur oberflächlich umrissen werden kann, wird auch für weitere Informationen zu GrGen.NET auf diese Dokumentation verwiesen.

3.2.1 Überblick

GrGen.NET ist ein Software Tool zur Graphersetzung, das auf dem Graphersetzungssystem GrGen (Graph Rewrite GENerator) aufbaut, dessen Entwicklung 2003 an der Universität Karlsruhe begonnen wurde (Geiß *et al.*, 2006). Im Rahmen einer Studienarbeit (Kroll, 2007) wurde GrGen auf die objektorientierte .NET-Umgebung portiert und erweitert, um eine größere Verbreitung des Tools zu ermöglichen.
Die wesentliche Aufgabe von GrGen.NET ist es, die Handhabung und Veränderungen graphartiger Datenstrukturen, deren Entitäten in vielfältigen Beziehungen zueinander stehen, zu vereinfachen. Dabei basieren sowohl GrGen wie auch GrGen.NET standardmäßig auf dem Single Pushout Approach (SPO).

In GrGen.NET wird der Prozess der Graphersetzung in vier Schritte aufgeteilt:

- Die Erstellung eines Ausgangsgraphen auf Basis eines Metamodells.
- Die Suche nach einem bestimmten Muster (Subgraph) im Ausgangsgraphen.

- Das Ausführen von Veränderungen an dem im Ausgangsgraphen erkannten Muster.
- Das Auswählen der als nächstes auszuführenden Regel.

Entsprechend dieser Aufteilung ist GrGen.NET in verschiedenen Sprachen aufgebaut.

In Grafik 3.3 sind die einzelnen Komponenten von GrGen.NET und ihre Beziehungen zueinander dargestellt:

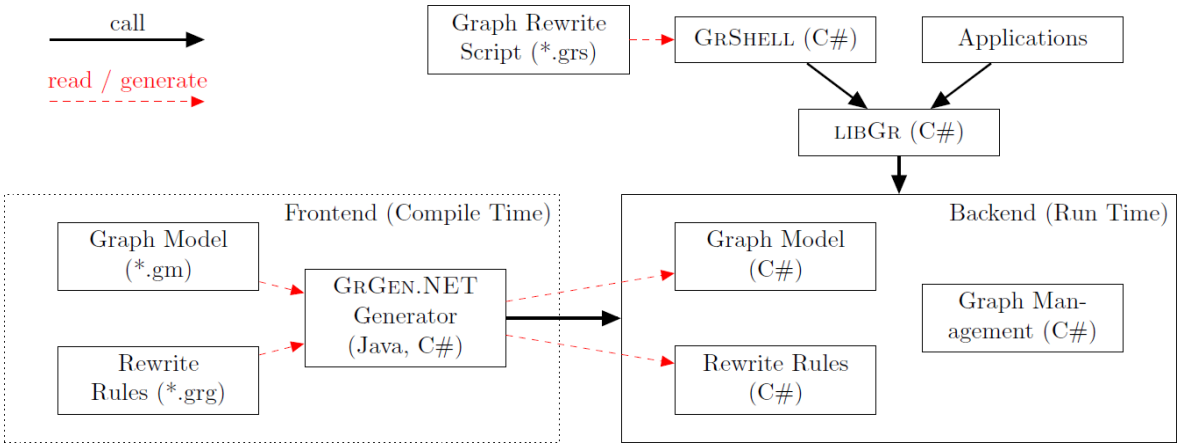

Abbildung 3.3: Systemkomponenten von GrGen.NET (Blomer *et al.*, 2013)(Kroll, 2007)

3.2.2 Funktionsumfang

Der gesamte Funktionsumfang von GrGen.NET überschreitet bei weitem die in dieser Arbeit an ein Graphersetzungssystem gestellten Anforderungen.
Im Folgenden werden die wesentlichen in dieser Arbeit genutzten Funktionen beschrieben. Diese lassen sich am übersichtlichsten bezüglich der Sprachen von GrGen.NET strukturieren. Die einzelnen Sprachen haben unterschiedliche Syntaxen, die jeweils in einem eigenen Dateiformat abgelegt werden. Erstellt werden können diese Dateien in einem beliebigen Texteditor.

Zur Erstellung des Metamodells eines Graphen dient die sogenannte ***graph model (metamodel) language***. Sie erlaubt die Definition verschiedener Klassen und Unterklassen von Knoten und Kanten sowie deren Attributen. Dabei werden sowohl ungerichtete Kanten als auch gerichtete Multigraphen, die mehrfache Kanten der gleichen Klasse zwischen denselben Knoten ermöglichen, unterstützt. Weiterhin ist durch den objektorientierten Aufbau die Vererbung von Attributen durch Klassen an ihre Unterklassen möglich.
Bei der Definition der Attribute können gängige Datentypen wie *Boolean*, *String*, *Integer* oder *Double* genutzt werden.
Das in dieser Sprache definierte Metamodell eines Graphen wird in einer *graph model description file* (*.gm) abgelegt.

Die Graphersetzungsregeln werden in der ***pattern language*** und der ***rewrite language*** formuliert.
Mit Hilfe der *pattern language* wird in erster Linie der Mustergraph einer Graphersetzungsregel anhand eines sogenannten *Graphlets* definiert. Dieses *Graphlet* beschreibt den Subgraphen, nach dem bei Anwendung der Graphersetzungsregel im Ausgangsgraphen der Ersetzungsoperation gesucht wird. Im *Graphlet* können neben der reinen Struktur des Mustergraphen auch die Typen der Knoten und Kanten, aus denen sich dieser Mustergraph zusammensetzt beschrieben werden.
Des Weiteren ist es möglich, Bedingungen an die Werte der Attribute dieser Knoten und Kanten zu stellen, um die Ersetzungsregel nur in bestimmten Fällen ausführen zu lassen.
Zusätzlich kann eine Graphersetzungsregel über Parameter verfügen, die beim Aufrufen der Regel übergeben werden müssen. Auch die Anzahl und Art dieser Parameter werden in der *pattern language* festgelegt.

In der *rewrite language* werden die Ersetzungsoperationen definiert. Entsprechend dem SPO-Ansatz ist es möglich, Elemente des Graphen zu erhalten, zu löschen oder neue Elemente hinzuzufügen. Dabei können auch bereits vorhandene Elemente kopiert oder Kanten andere Anfangs- und Endknoten zugewiesen werden.
Zusätzlich können die Werte der Attribute von bereits bestehenden oder neu erstellten Elementen verändert oder neu berechnet werden. Es ist dabei möglich, die Werte der Parameter der Regel zur Zuweisung der Werte von Attributen zu nutzen. Auch die Werte der Attribute bereits bestehender Elemente des Graphen können dafür genutzt werden. Im Falle von Attributen, die numerische Werte beinhalten, können auch arithmetische Ausdrücke formuliert werden, um diese (neu) zu berechnen.
Die Definition der Graphersetzungsregeln erfolgt in einer *rule set file* (*.grg).

Für die eigentliche Anwendung des Programms, wie beispielsweise die Ausführung von Ersetzungsregeln, die manuelle Definnition eines Ausgangsgraphen oder den Export der Daten eines Graphs, wird die ***rule application control language*** genutzt. Alternativ können die Befehle dieser Sprache auch manuell in die Konsolenanwendung GrShell eingegeben werden, mit der GrGen.NET gesteuert wird.
Eine Auflistung von Befehlen in der entsprechenden Syntax wird in einem *graph rewrite script* (*.grs) abgespeichert. Wird eine solche Datei durch die GrShell aufgerufen, erfolgt eine Stapelverarbeitung der definierten Befehle.

3.2.3 Aufbau eines Metamodells in GrGen.Net

Auch in GrGen.NET ist das Metamodell gemäß seiner Beschreibung in Abschnitt 3.1.1 ein integraler Bestandteil hinsichtlich dem Aufbau des Graphen und der Anwendung von Graphersetzungsregeln.

Wie bereits im vorigen Abschnitt festgehalten, kommt zur Erstellung des Metamodells die in GrGen.NET enthaltene *graph model language* zum Einsatz. Ein in der zugehörigen Syntax definiertes Metamodell wird durch die Dateierweiterung **.gm* gekennzeichnet.

In Folgenden werden die Eigenschaften der *graph model language* beschrieben. Für detaillierte Codebeispiele wird auf die prototypische Umsetzung des in dieser Thesis erarbeiteten Graphersetzungssystems in Kapitel 7 verwiesen.

Das Metamodell wird in GrGen.NET auch nur als *graph model* bezeichnet. Es gliedert sich nach (Blomer *et al.*, 2013) in folgende Schlüsseleigenschaften:

Typen
: Knoten und Kanten werden jeweils bestimmten Typen zugeordnet. Das Prinzip ist dabei mit Klassen in objektorientierten Programmiersprachen vergleichbar. Kantentypen in GrGen.NET können gerichtet und ungerichtet sein.

Attribute
: Wie in Abschnitt 3.1.1 beschrieben, können auch in GrGen.NET Knoten und Kanten Attribute besitzen. Die Attribute, die einem Knoten oder einer Kante zugewiesen werden können, hängen vom jeweiligen Typ des Knotens bzw. der Kante ab. Auch die Attribute selbst sind insofern typisiert, als dass sie jeweils einen bestimmten Datentyp speichern können.

Vererbung
: Die Typen von Knoten und Kanten können durch (mehrfache) Vererbung erstellt werden. Dies vereinfacht die Definition von Attributen, da Subtypen die Attribute ihrer Supertypen erben und diese dementsprechend nicht erneut definiert werden müssen.

Diese Schlüsseleigenschaften beschreiben bereits alle Funktionen der *graph model language*, die in dieser Arbeit genutzt werden. Weitere detaillierte Funktionen und Eigenschaften die GrGen.NET zur Erstellung eines Metamodells bietet, werden an dieser Stelle nicht aufgeführt, können aber, wie bereits zuvor erwähnt, der Dokumentation der Software entnommen werden.

3.2.4 Aufbau von Graphersetzungsregeln in GrGen.NET

Graphersetzungsregeln werden in GrGen.NET standardmäßig entsprechend dem Single Pushout Approach (SPO) ausgeführt.
Dementsprechend wird auch der Aufbau einer Graphersetzungsregel in GrGen.NET im Wesentlichen durch einen Muster- und einen Ersetzungsgraphen bestimmt. Muster- und Ersetzungsgraphen werden dabei, wie in Abschnitt 3.2.2 erwähnt, anhand von *Graphlets* definiert. In der Dokumentation von GrGen.NET wird der Begriff *Graphlet* wie folgt definiert:

> *A graphlet specifies a connected subgraph.* GrGen.NET *provides graphlets as a descriptive notation to define both, patterns to search for as well as the subgraphs that replace or modify matched spots in a host graph. Any graph can be specified piecewise by a set of graphlets.* (Blomer *et al.*, 2013)

Diese *Graphlets* stellen somit den wesentlichen Bestandteil einer Graphersetzungsregel in GrGen.NET dar. Neben der reinen Definition von Muster- und Ersetzungsgraph können den Elementen dieser Graphen auch innerhalb einer Regel Namen zugewiesen werden. Über diese Namen ist es möglich, bestimmte Elemente anzusprechen und so zu verändern, zu löschen oder beizubehalten. Dies ist zum Beispiel nötig, wenn im Ersetzungsgraphen definiert wird, dass eine neu erstellte Kante an einen bereits bestehenden Knoten anschließen soll. Außerdem können nur über die Namen die Werte der Attribute von Elementen des Graphen verändert werden.

Neben den *Graphlets* spielen die Attribute der Knoten und Kanten, die innerhalb einer Regel definierten werden, eine wichtige Rolle.
So kann die Ausführung einer Regel, wie bereits erwähnt, an Bedingungen hinsichtlich der Werte einzelner Attribute geknüpft werden.

Weiterhin können bei der Ausführung einer Graphersetzungsregel durch einen Befehl in der GrShell Parameter (verschiedener Datentypen) an die Regel übergeben werden. So ist es möglich die Werte der Attribute, die durch eine Regel verändert oder neu erstellt werden, beim Aufruf der Regel zu beeinflussen.

Schematisch ist eine Graphersetzungsregel in GrGen.NET (vereinfacht) wie folgt aufgebaut:

- Benennung der Regel.
- Definition von Parametern, die beim Aufruf der Regel übergeben werden müssen (optional).
- Ein oder mehrere *Graphlets*, die den Mustergraphen der Regel beschreiben.
- Festlegen von Voraussetzungen, unter denen die Regel ausgeführt wird (optional).
- Ein oder mehrere *Graphlets*, die den Ersetzungsgraph der Regel beschreiben.
- (Neu)Berechnung und Zuweisung der Werte von Attributen der Elemente des Ersetzungsgraphen (optional).

Es ist in GrGen.NET möglich, diverse weitere Funktionen in eine Graphersetzungsregel einzubinden. Diese werden aber im Rahmen dieser Arbeit nicht genutzt und daher auch nicht weiter ausgeführt.

3.2.5 Datenformate und -export

Neben den bereits beschriebenen Datenformaten **.gm*, **.grg* und **.grs*, durch die ein Graphersetzungssystem in GrGen.NET definiert wird, ist hinsichtlich dieser Thesis vor allem relevant, in welchen Formaten GrGen.NET einen erstellten Graphen exportieren kann.

Für einen Export der Daten eines Graphen in ein externes Datenformat unterstützt GrGen.NET nur die Graph eXchange Language (GXL).
GXL wurde als Standardaustauschformat für graphbasierte Software-Tools entwickelt und basiert auf der Struktur der Extensible Markup Language (XML). Ein wichtiges Merkmal ist dabei, dass nicht nur ein Graph selbst, sondern auch das zugrunde liegende Metamodell in einer *.gxl-Datei gespeichert wird. (Winter *et al.*, 2002)
Weitere Informationen zu GXL sind auf der Website [1] des Projekts zu finden.

Für diese Arbeit ist allerdings nur der Teil einer GXL-Datei relevant, der die Daten des durch GrGen.NET exportierten Graphen enthält. Das zusätzlich in der GXL-Datei abgelegte Metamodell ist hingegen für die weitere Verwendung der Datei obsolet.
Zur Veranschaulichung des Aufbaus einer GXL-Datei wird im Folgenden die Repräsentation zweier Knoten und eine Kante im GXL-Format dargestellt.

Knoten „Knoten1“

```
1 <node id="n54267293">
2   <type xlink:href="#A" />
3   <attr name="Caption">
4     <string>Knoten1</string>
5   </attr>
6 </node>
```

Knoten „Knoten2“

```
1 <node id="n18643596">
2   <type xlink:href="#B" />
3   <attr name="Caption">
4     <string>Knoten2</string>
5   </attr>
6 </node>
```

Es wird dabei in Zeile 1 jeweils das XML-Element *node* geöffnet, um die Definition eines Knotens einzuleiten. Durch das XML-Attribut *id* kann ein Knoten eindeutig indentifiziert werden.
In Zeile 2 wird der Typ des Knotens festgelegt. Dieser ist hier entweder „A“ oder „B“.
Zeile 3 leitet ein Attribut des Knoten durch *attr* ein. Durch das XML-Attribut *name* wird der Name des Attributs (in diesem Fall *Caption*) festgelegt. In der folgenden Zeile wird ein neues XML-Element eingeleitet, dessen Name dem Datentyp des Attributes entspricht.
Zeile 5 und 6 enthalten jeweils die End-Tags der Elemente *attr* und *node*.

[1] http://www.gupro.de/GXL/

Kante „Kante1“

```
1 <edge id="e33574638" from="n54267293" to="n18643596">
2  <type xlink:href="#C" />
3  <attr name="Caption">
4     <string>Kante1</string>
5  </attr>
6  <attr name="Value">
7     <int>5</int>
8  </attr>
9 </edge>
```

Der Aufbau der Repräsentation einer Kante entspricht innerhalb einer GXL-Datei weitgehende dem Aufbau eines Knotens. Wesentlicher Unterschied ist, dass in Zeile 1 das XML-Element *edge* zwei weitere XML-Attribute (*from*, *to*) besitzt, die festlegen, welche Knoten die jeweilige Kante verbindet.
Anschließend werden auch hier zuerst der Typ der Kante und danach ihre Attribute und ihre Werte definiert.

In folgender Abbildung 3.4 ist der aus zwei Knoten und einer Kante bestehende Graph inklusive der Attribute von Knoten und Kanten dargestellt.
Zur Visualisierung wurde das in GrGen.NET integrierte Graphvisualisierungstool yComp[2] genutzt.

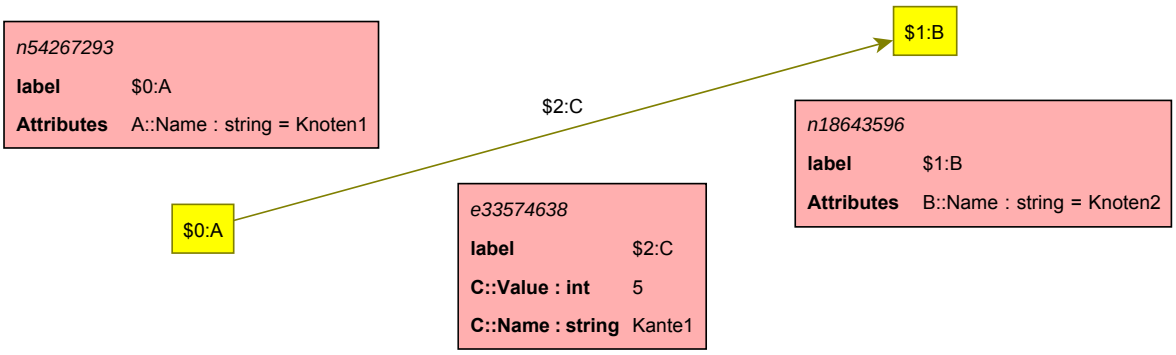

Abbildung 3.4: Visualisierung des durch eine GXL-Datei beschriebenen Graphen

[2] http://pp.ipd.kit.edu/firm/yComp

Kapitel 4

Parametrik

Nachdem im vorhergehenden Kapitel die Grundlagen von Graphentheorie und Graphersetzungssystemen beschrieben wurden, wird im diesem Kapitel auf das Konzept der parametrischen Modellierung eingegangen.

Dabei wird zuerst die parametrische Modellierung im Allgemeinen kurz umrissen und anschließend auf die in dieser Arbeit eingesetzte Software Autodesk Inventor eingegangen.

4.1 Parametrische Modellierung

Das Konzept der parametrischen Modellierung kommt im Bereich des Computer Aided Design (CAD) zum Einsatz und war schon in den ersten CAD-Systemen vorhanden (Sacks *et al.*, 2004).
Es ermöglicht nicht nur einfache Zeichnungen mit festen Abmessungen zu erstellen, sondern stattdessen das Aussehen der Zeichnung mit geometrisch-topologische Abhängigkeiten zwischen den geometrischen Entwurfselementen festzulegen. Eine Zeichnung dieser Art, deren Aussehen über die genannten Abhängigkeiten bzw. Zwangsbedingungen definiert ist, nennt man, wie in Abschnitt 2.1 bereits beschrieben, Skizze oder im Englischen *sketch*.

Eine solche Skizze ist in einem CAD-System eine Zeichnung, die noch keine präzisen Informationen zu tatsächlichen Dimensionen beinhaltet. Stattdessen definiert sie nur eine grobe Anordnung von geometrischen Elementen wie beispielsweise Punkten, Linien und Kreisen. Erweitert wird sie dabei durch die genannten Abhängigkeiten, die als geometrische und dimensionale Zwangsbedingungen bezeichnet werden.

In (Shah, 1995) wird das Vorgehen zur Erstellung einer Skizze wie folgt beschrieben:
In einem ersten Schritt erstellt der Nutzer das generelle Aussehen eines Designs, indem er gängige geometrische Modellierungsoperationen ausführt. Daraus resultiert ein Modell, das

die gewünschten geometrischen Elemente und ihre Verbindungen zueinander aber noch keine Dimensionen beinhaltet.
Als nächstes beschreibt der Nutzer die nötigen Verknüpfungen zwischen den Elementen des Modells mit Hilfe der genannten Zwangsbedingungen. Diese definieren die gewünschten (mathematischen) Beziehungen zwischen den numerischen Variablen der Elemente des Modells.
Anschließend wird durch das Modellierungssystem eine Lösung berechnet, bei der die Anforderungen aller Zwangsbedingungen erfüllt werden.
So kann der Nutzer letztendlich verschiedene Varianten des Modells erstellen, indem er die Werte der Variablen ändert und so eine Neuberechnung des Modells einleitet.

Die geometrischen und dimensionalen Zwangsbedingungen, mit deren Hilfe die Beziehungen zwischen den Elementen einer Skizze festgelegt werden können, werden in den folgenden beiden Abschnitten beschrieben.

4.1.1 Dimensionale Zwangsbedingungen

Dimensionale Zwangsbedingungen werden eingesetzt, um Abstände zwischen Elementen oder die Dimensionen dieser Elemente zu kontrollieren. Es kann sich dabei beispielsweise um horizontale und vertikale Abstände oder um Winkel und Durchmesser handeln. Die Größe dieser Zwangsbedingungen wird durch Parameter ausgedrückt, die entweder durch feste Werte oder durch Variablen definiert werden.
Diese Parameter können dabei voneinander abhängig sein oder durch die Variablen aufeinander verweisen. Diese Definition führt dazu, dass durch die Änderung des Werts eines einzelnen Parameters auch alle davon abhängigen Parameter verändert werden und dementsprechend das Aussehen der kompletten Skizze angepasst wird (Ji *et al.*, 2013).

4.1.2 Geometrische Zwangsbedingungen

Zusätzlich zu den dimensionalen Zwangsbedingungen können geometrische Zwangsbedingungen definiert werden. Sie dienen dazu, die eigentliche Anordnung der geometrischen Elemente, aus denen die Skizze aufgebaut ist, zu definieren. Die für diese Arbeit relevanten geometrischen Zwangsbedingungen werden im Folgenden kurz beschrieben:

- **Konzentrisch**: Zwei Kreise haben denselben Mittelpunkt
- **Gleich**: Zwei Linien haben dieselbe Länge
- **Vertikal**: Eine Linie ist parallel zur vertikalen Achse des Koordinatensystems
- **Horizontal**: Eine Linie ist parallel zur horizontalen Achse des Koordinatensystems
- **Koinzident**: Zwei Elemente treffen sich an einem Punkt.

- **Kollinear**: Zwei Liniensegmente liegen auf der selben Geraden
- **Fest**: Ein Element wird fest an seiner Position im Koordinatensystem verankert
- **Parallel**: Zwei Linien sind parallel zueinander
- **Lotrecht**: Zwei Linien sind rechtwinklig zueinander

4.2 Autodesk Inventor

Die in dieser Arbeit erstellten graphbasierten Repräsentationen geometrischer Modelle sollen in einer CAD-Software angezeigt und verändert werden können (siehe Abschnitt 2.3.3).

Im folgenden Abschnitt wird die dafür genutzte Software Inventor der Firma Autodesk kurz vorgestellt. Insbesondere wird dabei auf die Funktionen zur parametrischen Modellierung eingegangen.

4.2.1 Überblick

Autodesk Inventor ist eine parametrische 3D-CAD-Software, die durch das Unternehmen Autodesk entwickelt wird. Sie wird eingesetzt, um digitale 3D-Prototypen für Design, Visualisierung und Simulation von Produkten zu erstellen. Hauptsächlich wird die Software für mechanische Konstruktionen in den Bereichen Maschinen- oder Werkzeugbau eingesetzt.

Für diese Arbeit ist indes primär die Möglichkeit des Erzeugens von komplett parametrisierten zweidimensionalen Skizzen von Bedeutung.
Durch das beispielhafte Erstellen eines geometrischen Modells kann sehr einfach direkt in der Software visuell überprüft werden, welche dimensionalen und geometrischen Abhängigkeiten nötig sind, um eine vollständige Parametrisierung des Modells zu gewährleisten. Gleichzeitig kann dabei durch das testweise Durchspielen verschiedener Alternativen bei der Modellierung und Parametrisierung festgestellt werden, welche Varianten denkbar sind und dementsprechend auch in einer graphbasierten Repräsentation umgesetzt werden müssen.
Die Notwendigkeit dieses Vorgehens wird in Kapitel 5 detailliert beschrieben.

4.2.2 Parametrische Modellierung in Autodesk Inventor

Für diese Arbeit werden sogenannte 2D-Skizzen im Inventor betrachtet. Diese dienen dazu, in der eigentlich dreidimensionalen Umgebung der Software eine zweidimensionale Skizze auf einer planaren Fläche oder der Arbeitsebene eines Bauteils zu erstellen. Sie können dann genutzt werden, um durch Extrusion dreidimensionale Bauteilelemente zu erstellen und enthalten meist nur Konstruktions- und Referenzgeometrien.

Während des Skizziervorgangs werden geometrische Abhängigkeiten weitgehend automatisch durch das Programm definiert, so dass Ausrichtung und Beziehung zwischen den skizzierten geometrischen Elementen (Punkte, Linien, Kreise, Kurven, etc.) festgelegt sind.
Wird zum Beispiel ein Rechteck gezeichnet, besteht dieses eigentlich aus vier Linien, die erst durch automatisch definierte Abhängigkeiten das tatsächliche Rechteck bilden:

- An den Ecken des Rechtecks sind die Endpunkte der jeweiligen Linien als koinzident definiert (gelbe Quadrate).
- Die jeweils gegenüberliegenden Kanten des Rechtecks sind als parallel definiert.
- An einer der Ecken sind die angrenzenden Kanten als lotrecht definiert.
- Die horizontale und die vertikale Bemaßung sind jeweils dimensionale Zwangsbedingungen, über die die Abmessungen des Rechtecks angepasst werden können.

In der folgenden Abbildung 4.1 ist ein entsprechendes Rechteck inklusive der Icons, die die geometrischen Abhängigkeiten kennzeichnen, als Screenshot abgebildet.

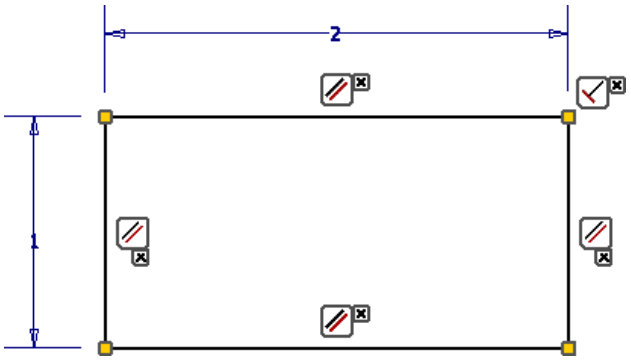

Abbildung 4.1: Durch parametrische Zwangsbedingungen repräsentierte Skizze dargestellt in Autodesk Inventor

In Inventor wird zwischen zwei Arten von geometrischen Abhängigkeiten unterschieden:

- Abhängigkeiten, die eine Ausrichtung zum Koordinatensystem festlegen, sind: Fest, Horizontal und Vertikal.
- Abhängigkeiten, die eine Beziehung zwischen zwei Elementen festlegen, sind: Lotrecht, Parallel, Tangential, Kollinear, Gleich, Koinzident, Stetig, Symmetrisch und Konzentrisch.

Die Symbole, die diesen geometrischen Abhängigkeiten in Autodesk Inventor beschreiben, sind in Abbildung 4.2 dargestellt.

Abbildung 4.2: Symbole der geometrischen Zwangbedingungen in Autodesk Inventor

Dimensionale Zwangsbedingungen werden durch den Inventor weitgehend automatisch eingefügt. Der Nutzer wählt entweder ein geometrisches Element aus, dessen Länge, Durchmesser, etc. durch eine parametrische Bemaßung festgelegt werden soll. Alternativ können zwei Elemente ausgewählt werden. In diesem Fall wird eine parametrische Bemaßung, die den Abstand zwischen diesen beiden Elementen bestimmt, erstellt.

4.2.3 Funktionen der API von Autodesk Inventor

Autodesk bietet für Benutzer, die den Funktionsumfang von Inventor durch selbst erstellte Plug-Ins oder Add-Ins erweitern wollen, eine COM-Programmierschnittstelle (API) an.
Über diese API ist es möglich durch gängige Programmiersprachen wie C++, C# oder Visual Basic auf Autodesk Inventor zuzugreifen und Befehle innerhalb des Programms auszuführen.

Dabei kann entweder ein Add-In erstellt werden, das automatisch beim Start von Inventor geladen wird und auf das über die Benutzeroberfläche zugegriffen werden kann. Alternativ ist es möglich, ein Plug-In zu erstellen. In diesem Fall greift ein externes Programm auf eine laufende Instanz von Autodesk Inventor zu.

Über die API kann eine Vielzahl der Befehle, die auch über die graphische Benutzeroberfläche verfügbar sind, ausgeführt werden. Außerdem können die Eigenschaften von Objekten innerhalb einer Zeichnung verändert werden.
Im Rahmen dieser Arbeit ist es jedoch in erster Linie wichtig, geometrische Elemente und parametrische Zwangsbedingungen in einer 2D-Skizze zu erstellen.
In Abbildung 4.3 sind die Objekte der API dargestellt, über die 2D-Skizzen bearbeitet werden können.
Diese umfassen alle Funktionen, die in dieser Arbeit genutzt werden mussten.

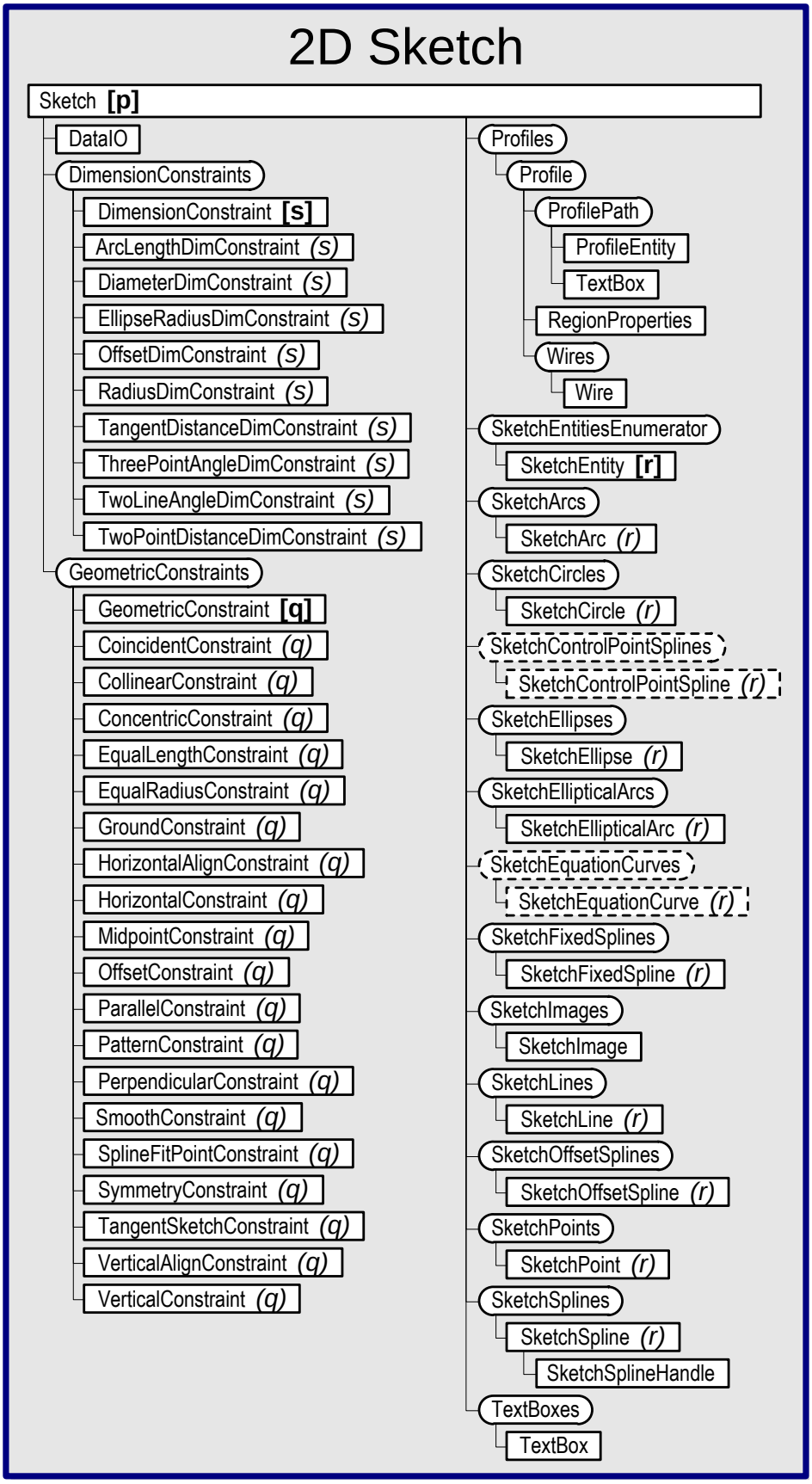

Abbildung 4.3: Ausschnitt aus dem *Object Model* der API von Autodesk Inventor 2014, entnommen aus dem Software Development Kit von Autodesk Inventor

Kapitel 5

Beschreibung der methodischen Vorgehensweise

In Kapitel 2 wurden die wesentlichen Problemstellungen herausgearbeitet, die sich durch den Ansatz der graphbasierten Repräsentation zweidimensionaler parametrischer geometrischer Modelle ergeben.
Weiterhin wurden in den Kapiteln 3 und 4 theoretische Grundlagen, die zur Lösung der beschriebenen Problemstellungen beitragen sollen, aufgeführt.

Im folgenden Kapitel wird nun beschrieben, wie auf Basis dieser Grundlagen bei der Konzeption des Graphersetzungssystems und der Weiterverarbeitung der graphbasierten Daten vorgegangen wurde.
In diesem Teil der Arbeit werden dabei noch keine wesentlichen Ergebnisse dieses Konzeptionsvorgangs aufgeführt. Stattdessen wird nur der eigentliche Vorgang, aus dem die im folgenden Kapitel 6 dargelegten Erkenntnisse resultieren, detailliert beschrieben.

Dazu wird zuerst ein genereller Überblick über den gesamten Vorgang erstellt. Anschließend werden die wesentlichen methodischen Schritte der Reihe nach aufgelistet und erörtert.
Dies soll dazu dienen, diesen Vorgang nachvollziehbar darzustellen und verdeutlichen, inwiefern das finale Konzept des Graphersetzungssystems erst durch eine Reihe von verschiedenen Versuchen und deren Überprüfung entstanden ist.

Es ist hier zu bemerken, dass anfangs mit sehr einfachen Skizzen und Detaillierungsvorgängen, die nur wenige geometrische Elemente und parametrische Zwangsbedingungen aufwiesen, gearbeitet wurde. Im Laufe des Vorgangs wurde die Komplexität diese Skizzen dann immer weiter erhöht, um das erarbeitete Konzept in seiner Allgemeingültigkeit und Flexibilität zu verbessern.

Weiterhin ist an dieser Stelle anzumerken, dass auch die in diesem Kapitel vorgestellte Methodik erst im Laufe der Forschungsarbeiten konzipiert bzw. vom Autor herausgearbeitet wurde. Aufgrund dessen ist nur eine geringe Zahl an Literaturverweisen aufgeführt, da bei der Literaturrecherche keine entsprechenden Quellen gefunden wurden, in denen ein solches Vorgehen beschrieben ist.

5.1 Überblick

Prinzipiell setzt sich ein Graphersetzungssystem aus einem Metamodell zur Definition von Knoten und Kanten und einer Menge von Graphersetzungsregeln, die den Graphen aufbauen bzw. verändern, zusammen.
Diese werden im Weiteren als Kern des Graphersetzungssystems bezeichnet.

Der Kern muss dabei den Anforderungen, die sich an das Graphersetzungssystem stellen, genügen. Diese Anforderungen ergeben sich durch die in Abschnitt 2.3 herausgearbeiteten Zielstellungen.

Dabei betrifft die Problemstellung „Repräsentation parametrischer Skizzen durch einen Graphen“ im Wesentlichen den Aufbau des Metamodells, während sich „Detaillierung der Skizze durch Transformation des Graphen“ weitgehend direkt auf die Definition der Ersetzungsregeln beziehen lässt.
Aus der dritten Problemstellung „Darstellung in einem parametrischen Modellierer“ ergeben sich weiterhin Anforderungen, die das gesamte Graphersetzungssystem betreffen und sowohl bei der Konzeption des Metamodells, als auch bei der Erstellung der Graphersetzungsregeln beachtet werden müssen.

Trotz dieser Beziehungen ist die Konzeption des Graphersetzungssystems in jedem Fall als Iterationsvorgang zu verstehen, bei der nach jedem Entwicklungsschritt überprüft werden muss, ob Konflikte, die zu den aus den Problemstellungen abgeleiteten Anforderungen entstehen, behoben werden müssen.

Dieser Vorgang ist ein wesentlicher Bestandteil der Forschungsarbeiten, die im Rahmen dieser Thesis durchgeführt wurden. Es wird deshalb nicht nur das Endergebnis in Form des funktionsfähigen Graphersetzungssystems beschrieben, sondern auch die Herangehensweise, die zu diesem Endergebnis geführt hat.

In der folgenden Abbildung 5.1 ist dargestellt, in welchen Schritten dieser Vorgang prinzipiell durchgeführt wurde.

An erster Stelle steht dabei die Analyse einer Skizze, beziehungsweise der verschiedenen Detaillierungsstufen dieser Skizze, die repräsentiert werden sollen. Dabei wird untersucht, aus welchen geometrischen und parametrischen Elementen eine Skizze überhaupt bestehen

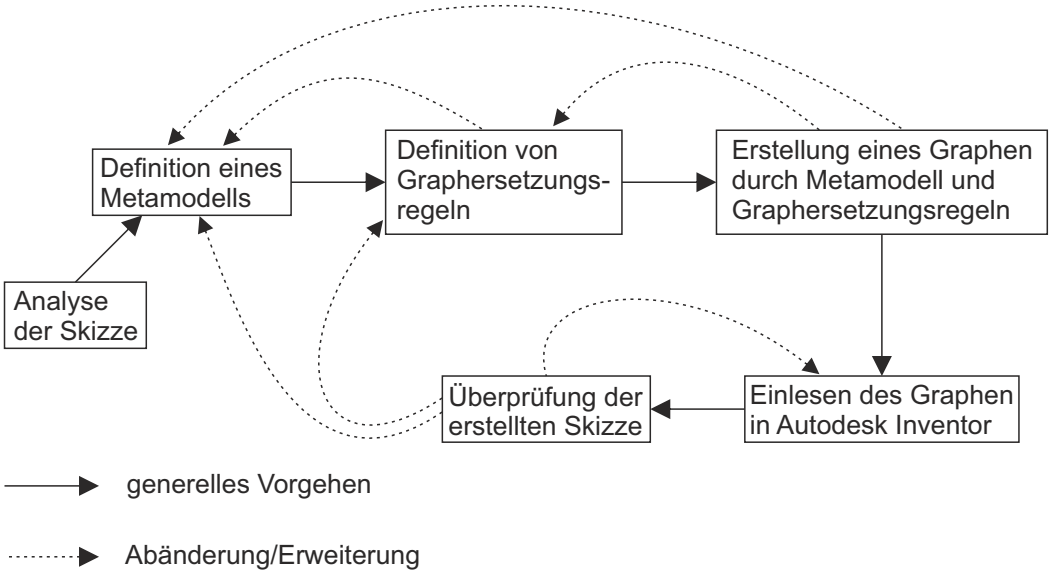

Abbildung 5.1: Iteratives Vorgehen zur Konzeption von Graphersetzungssystem und Visualisierung des Graphen

kann. Aus dieser Analyse ergibt sich eine grundlegende Struktur des Metamodells, da nun bekannt ist, welche Elemente durch Knoten und Kanten im Graph dargestellt werden müssen.

Auf Basis dieses Metamodells werden Graphersetzungsregeln erstellt und auch schon in GrGen.NET implementiert, um ihre Anwendbarkeit zu überprüfen. Durch die aufeinanderfolgende Anwendung dieser Ersetzungsregeln können verschiedene Graphen erzeugt bzw. bereits bestehende Graphen erweitert oder verändert werden, die jeweils einer Detaillierungsstufe der Skizze entsprechen. Sowohl bei der Erstellung der Regeln, als auch bei der Erzeugung des Graphen, muss simultan überprüft werden, ob das vorher definierte Metamodell in seiner Definition sinnvoll und detailliert genug ist.
Stellt sich heraus, dass im Metamodell Eigenschaften bzw. Attribute von Knoten und Kanten nicht ausreichend definiert sind, um eine fehlerfreie und sinnvolle Ausführung der Graphersetzungsregeln zu gewährleisten und so den gewünschten transformierten Graphen zu erhalten, muss eine Anpassung erfolgen. Gegebenenfalls müssen anschließend auch die Graphersetzungsregeln angepasst werden. Dieser Vorgang muss so lange wiederholt werden, bis die Erstellung des Graphen das gewünschte Ergebnis liefert.

Entspricht der resultierende Graph dem gewünschten Resultat, ist im nächsten Schritt zu überprüfen, ob der Graph korrekt weiterverarbeitet und durch das für diesen Vorgang entwickelte Software-Tool (im Folgenden G2I-Tool) in Autodesk Inventor eingelesen werden kann. Denn nur wenn die Skizze, die letztendlich durch die API erstellt wird, eindeutig dem gewünschten Ergebnis entspricht, ist die graphbasierte Repräsentation erfolgreich. Treten Fehler bei der Ausführung der API-Befehle, die die geometrischen Elemente und parametrischen Zwangsbedingungen zeichnen auf, oder ergibt sich nicht die beabsichtige Geometrie,

müssen wiederum Metamodell und Ersetzungsregeln abgeändert oder erweitert werden. Zusätzlich muss auch das G2I-Tool immer so angepasst werden, dass die dort implementierten Klassen und Methoden den Graphen korrekt interpretieren können. Beispielsweise sind bei einer Veränderung des Metamodells hier gegebenenfalls Änderungen notwendig.

Für die Erstellung eines tatsächlich funktionsfähigen Graphersetzungssystems und des darauf abgestimmten G2I-Tools wurde dieser Iterationsvorgang etliche Male wiederholt.

Im Folgenden werden die einzelnen Schritte dieses Vorgangs noch einmal ausführlich beschrieben.

5.2 Analyse der Skizze

Eine tatsächliche Skizze kann in dem hier beschriebenen Vorgang gewissermaßen als Eingangs- und als Ausgangsprodukt bezeichnet werden. Dies begründet sich darin, dass das Ziel des Graphersetzungsvorgangs die Darstellung einer solchen Skizze ist, aber auch bei der Untersuchung des Vorgangs schon von Anfang an das gewünschte Endergebnis analysiert werden muss, um festzustellen, wie dieser Vorgang durchgeführt werden kann.

Es muss also zuerst eine Reihe von Skizzen, die ein reales Objekt in verschiedenen Detaillierungsstufen darstellen, untersucht werden. Dafür wurde weiterhin der bereits in Kapitel 2 abgebildete Tunnelquerschnitt in verschiedenen Detaillierungsstufen verwendet.
Um den Aufbau dieser verschiedenen Skizzen zu verstehen und formal zu beschreiben, wurden sie als erstes einzeln in Autodesk Inventor manuell gezeichnet, da auch in diesem Programm letztendlich die Visualisierung der Skizzen aus den Daten des erstellten Graphen stattfinden soll. Deshalb muss sich der komplette Aufbau des Graphen sinnvollerweise auch daran orientieren, wie geometrische Elemente und parametrische Zwangsbedingungen als Skizze in Autodesk Inventor modelliert werden können.

Die so erstellten Skizzen wurden anschließend, neben den genannten Elementen und Zwangsbedingungen, vor allem dahingehend analysiert, inwiefern sich die Zwangsbedingungen gegenseitig beeinflussen, beschränken und wie sie mit den geometrischen Elementen verknüpft werden können.

In folgender Abbildung 5.2 ist eine in Autodesk Inventor modellierte Skizze dargestellt. Dabei sind neben den geometrischen Elementen auch die parametrischen Zwangsbedingungen visualisiert.

Aus dieser Abbildung lässt sich beispielsweise ableiten, dass die Zwangsbedingung *Koinzident* (in der Abbildung durch ein gelbes Quadrat dargestellt), entweder die Endpunkte zweier Linien, einen Punkt mit dem Mittelpunkt eines Kreises, oder den Endpunkt einer Linie mit dem Kreisbogen eines Kreises verknüpfen kann.

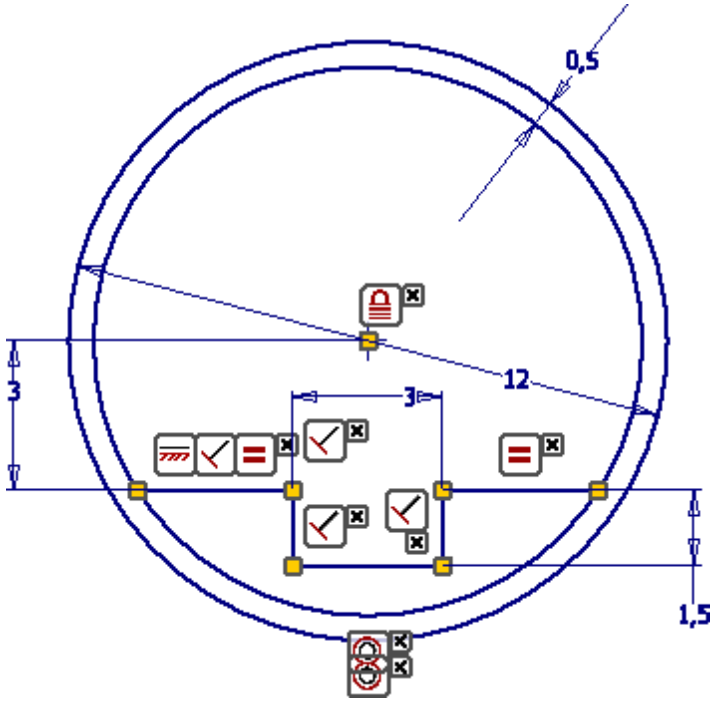

Abbildung 5.2: Darstellung der Geometrie einer Skizze und ihrer parametrischen Zwangsbedingungen in Autodesk Inventor

Weiterhin kann hinsichtlich dimensionaler Zwangsbedingungen festgestellt werden, dass sich diese entweder auf nur ein geometrisches Element (z.B. Radius eines Kreises) oder auf zwei Elemente (z.B. Abstand der Endpunkte zweier Linien) beziehen können.
Auch wenn diese Feststellungen im ersten Moment trivial klingen mögen, sind sie doch wichtige Grundlagen für die in Abschnitt 6.1 beschriebenen Voraussetzungen zum Aufbau des Graphersetzungssystems.

5.3 Definition des Metamodells

Auf Basis der durch die Analyse der Skizze gewonnenen Erkenntnisse kann anschließend ein Metamodell definiert werden. Dieses beinhaltet die Knoten und Kanten des Graphen, die die geometrischen Elemente und parametrische Zwangsbedingungen, aus denen sich die Skizze zusammensetzt, repräsentieren.

Dabei müssen für alle Elemente und Zwangsbedingungen entsprechende Typen von Knoten und Kanten angelegt werden. Zusätzlich müssen diesen Knoten und Kanten verschiedene Attribute zugewiesen werden. Außerdem wird beispielsweise festgelegt, ob es sich bei Kanten eines Typs um gerichtete oder ungerichtete Kanten handelt.

Auf Basis dieser Definition wurde das Metamodell in GrGen.NET implementiert, um zu überprüfen, ob und wie es in diesem Programm korrekt aufgebaut werden kann.

Es ist weiter anzumerken, dass das Metamodell gegebenenfalls auf Basis von Anforderungen, die sich im Laufe des weiteren Vorgehens ergeben, angepasst wird.

5.4 Graphersetzungsregeln

Um aus den verschiedenen Typen von Knoten und Kanten des Metamodells automatisiert einen Graph zu erzeugen, werden Graphersetzungsregeln benötigt. Die detaillierte Konzeption der Graphersetzungsregeln wird in Kapitel 6 erläutert.

Wesentlich ist an dieser Stelle vielmehr, dass die verschiedenen Graphersetzungsregeln direkt testweise in GrGen.NET programmiert und kompiliert wurden, um zu überprüfen, ob eine fehlerfreie Ausführung der Regeln möglich ist. Denn nur durch die praktische Anwendung der Regeln und der Untersuchung des resultierenden Graphen kann festgestellt werden, ob eine Regel die gewünschte Veränderung des Graphen bewirkt.

Sollte bei der Definition und Implementierung der Graphersetzungsregeln festgestellt werden, dass sich diese durch fehlende Typen oder Attribute im Metamodell nicht korrekt erstellen lassen, so müssten dementsprechend Anpassungen am Metamodell vorgenommen werden.

5.5 Erstellung und Export des Graphen

Durch die Anwendung der in GrGen.NET implementierten Graphersetzungsregeln ist es möglich, verschiedene Graphen auf Basis des Metamodells zu erstellen. Dabei wird der Graph nach jeder Regelanwendung in seiner aktuellen Form abgespeichert, damit nach Ausführung der gewünschten Regeln die Repräsentationen aller so durchgeführter Detaillierungsschritte vorhanden sind.
Das Speichern bzw. der Export der Daten des Graphen erfolgt dabei im von GrGen.NET unterstützten GXL-Format, welches in Kapitel 3 detaillierter beschrieben ist.

Weiterhin kann in diesem Schritt der Graph durch GrGen.NET in yComp, einem Tool zur Visualisierung von Graphen, angezeigt werden. Dort ist er wesentlich besser lesbar, als im GXL-Format. So wird eine visuelle Überprüfung des Aufbaus des Graphen ermöglicht. Dies erfolgt indem untersucht wird, ob alle Knoten und Kanten angelegt wurden und ob in der Struktur des Graphen die Knoten und Kanten richtig miteinander verknüpft sind. Des Weiteren können die Attribute von Knoten und Kanten angezeigt werden, um festzustellen, ob diese korrekt zugewiesen wurden.

Werden bei dieser Überprüfung Fehler im Aufbau des Graphen entdeckt, muss festgestellt werden, durch welche Fehldefinitionen im Kern des Graphersetzungssystems diese entstehen. Entsprechend sind anschließend Metamodell und Graphersetzungsregeln anzupassen.

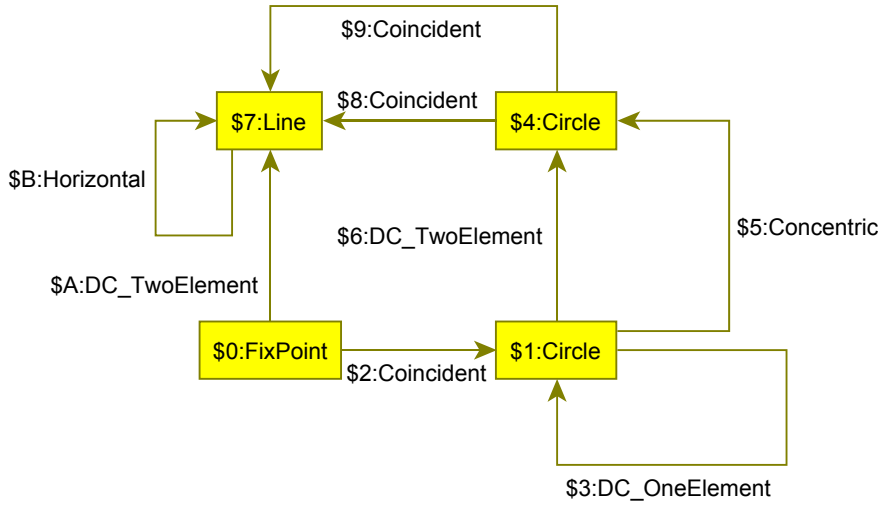

Abbildung 5.3: Visualisierung eines Graphen in yComp

5.6 Einlesen des Graphen in Autodesk Inventor

Im nächsten Schritt werden die durch GrGen.NET exportierten Daten durch das dafür eigens entwickelte G2I-Tool weiterverarbeitet. Dabei wird eine Deserialisierung der durch GrGen.NET exportierten Daten durchgeführt, um die Knoten und Kanten des Graphen innerhalb des Programms abzubilden und verwenden zu können. Dazu werden jeweils Instanzen von Klassen, die für die einzelnen Knoten und Kantentypen angelegt wurden, erstellt, die die entsprechenden Eigenschaften und Attribute enthalten.

Anschließend werden alle so erstellten Instanzen an Methoden der API von Autodesk Inventor übergeben. Dabei wird für jede Instanz jeder Klasse eine Methode aufgerufen, die je nach Typ eine Konstruktionsoperation im Inventor ausführt und dort einer Skizze entweder ein geometrisches Objekt oder eine parametrische Zwangsbedingung hinzufügt.
Prinzipiell werden dabei zuerst die geometrischen Elemente erstellt. Erst anschließend werden die parametrischen Zwangsbedingungen hinzugefügt, da die hierfür verwendeten Methoden auf die bereits erstellten geometrischen Elemente zugreifen müssen, um diesen die Zwangsbedingungen zuzuweisen bzw. sie durch die Zwangsbedingungen miteinander zu verknüpfen.

Bei diesem Vorgang muss durchgehend eine fehlerfreie Ausführung des G2I-Tools gewährleistet werden. Treten beim Kompilieren oder Ausführen Fehler auf, ist dies entweder auf tatsächliche Fehler im Code oder auf eine Inkompatibilität zwischen dem Programm und den Daten des Graphen, der verarbeitet wird, zurückzuführen.
Während Fehler im Code direkt behoben werden können, müssen bei Inkompatibilität entweder die Funktionen des Programms, das Graphersetzungssystem oder beides angepasst werden.

5.7 Überprüfung der Skizze

Nach einer fehlerfreien Ausführung des G2I-Tools muss zuletzt die in Autodesk Inventor dargestellte Skizze analysiert, beziehungsweise mit der Ausgangsskizze verglichen werden. Nur so kann festgestellt werden, ob ein Graph die Skizze tatsächlich korrekt repräsentiert bzw. ob die Graphersetzungsoperationen einen Graph erzeugen, aus dem eine parametrisch bestimmte und eindeutige Skizze erzeugt werden kann.

Einerseits muss dabei die korrekte Erzeugung und Anordnung der geometrischen Elemente geprüft werden. Andererseits ist die Parametrik der Skizze zu analysieren. Dabei muss festgestellt werden, ob die Zwangsbedingungen vollständig erzeugt wurden, ob sie die richtigen geometrischen Elemente verbinden und ob ihre Werte (evtl. in Abhängigkeit von den Werten anderer Zwangsbedingungen) korrekt sind.

Wenn bei der Überprüfung der Skizze Fehler auftreten, müssen diese auf ihre Ursache hin untersucht werden. Dabei sind im Wesentlichen zwei Möglichkeiten denkbar:
Einerseits kann durch logische Fehler im Code des G2I-Tools eine fehlerhafte Skizze erzeugt werden. In diesem Fall muss die Struktur des Programms entsprechend angepasst werden.
Andererseits kann die Ursache in der Beschaffenheit des repräsentierenden Graphen liegen. In einfachen Fällen wurden dabei Knoten, Kanten oder deren Attribute nicht erstellt und fehlen deshalb in der erzeugten Skizze. In komplizierteren Fällen liegt der Grund in einer falschen Definition der parametrischen Zwangsbedingungen, wodurch die gesamte Struktur des Graphen, beziehungsweise der Graphersetzungsregeln, die ihn erstellen, überarbeitet werden muss.

5.8 Fazit

Mit dem in diesem Kapitel beschriebenen methodischen Vorgehen konnten das Graphersetzungssystem bzw. dessen Umsetzung in der Graphersetzungssoftware GrGen.NET und das G2I-Tool immer weiter verbessert werden. So war es letztendlich möglich, einen für die in dieser Thesis behandelten Beispielskizzen fehlerfrei funktionierender Ablauf von Graphersetzung und -visualisierung als Skizze zu gewährleisten.

Durch eine Fortführung des beschriebenen iterativen Vorgehens ist es nun generell möglich, in konsekutiven Untersuchungen das bestehende Graphersetzungssystem im Umfang seiner Abbildungsmöglichkeiten zu erweitern und in seiner Flexibilität, vor allem in Bezug auf die Wiederverwendung von Graphersetzungsregeln, zu erhöhen.

Kapitel 6

Entwicklung des Graphersetzungssystems

Im folgenden Kapitel wird nun die eigentliche Umsetzung der in dieser Arbeit untersuchten Methode der graphbasierten Repräsentation zweidimensionaler Skizzen beschrieben und in diesem Zusammenhang auch auf die in Kapitel 2 herausgearbeiteten Problemstellungen eingegangen.

Dazu werden zuerst die wesentlichen Erkenntnisse, die sich aus dem in Kapitel 5 ausgearbeiteten Vorgehen ergeben haben, beschrieben. Anschließend werden die Elemente und Informationen, die in der graphbasierten Repräsentation enthalten sein müssen, aufgeführt und untersucht.
Darauf aufbauend wird die Definition eines Metamodells herausgearbeitet und hinsichtlich ihrer Abbildungsmächtigkeit eingeordnet. Nun ist erkennbar, welche Arten von Graphen auf Basis des Metamodells erzeugt und welche Skizzen dementsprechend repräsentiert werden können.

Auf Grundlage des Metamodells werden anschließend die grundlegenden Vorgaben, nach denen Ersetzungsregeln aufgebaut sein müssen, festgelegt. Konkrete Ersetzungsregeln werden danach für verschiedene Beispiele definiert, um das richtige Vorgehen bei der Erstellung der Regeln zu beschreiben.

Übergeordnet wird dabei versucht, das Konzept in seiner Formulierung so weit wie möglich allgemeingültig zu halten, um eine fallspezifische Anwendung auf die Modellierung und Darstellung verschiedener Szenarien zu ermöglichen.
Als Fallbeispiel wird aber im Wesentlichen das schon aus Kapitel 2 bekannte Beispiel der Detaillierungsschritte eines vereinfachten Tunnelquerschnitts verwendet.

6.1 Grundsätzliche Feststellungen

Durch die in Kapitel 5 beschriebene Vorgehensweise konnten mehrere grundlegende Ansprüche, die von Graphersetzungssystem und G2I-Tool erfüllt werden müssen, herausgearbeitet werden.
Auf Basis der in diesem Abschnitt aufgeführten Erkenntnisse werden im restlichen Teil des Kapitels das Metamodell, die Graphersetzungsregeln und die Funktionsweise des G2I-Tools beschrieben.

6.1.1 Eigenschaften des Graphen

Bei den in dieser Arbeit zur Repräsentation von Skizzen genutzten Graphen handelt es sich um gerichtete Multigraphen.
Multigraphen können im Gegensatz zu sogenannten einfachen Graphen auch Schleifen und parallele Kanten enthalten.

Die Verwendung von gerichteten Graphen ergibt sich durch die in Abschnitt 6.1.3 beschriebenen Andockstellen. Da diese sich jeweils auf einen der beiden Knoten, die eine Kante verbinden, beziehen, muss durch die Richtung der Kante festgelegt werden, welcher der beiden Knoten (Anfangs- oder Endknoten) gemeint ist.
Obwohl dies nur für einen Teil der Kantentypen relevant ist, werden trotzdem alle Kanten mit einem Richtungssinn versehen, weil sich ansonsten Probleme beim objektorientierten Aufbau des Metamodells ergäben. Dies bedingt sich dadurch, dass es in der Graphersetzungssoftware GrGen.NET nicht möglich ist, ungerichtete Kanten von gerichteten Kanten (bzw. umgekehrt) abzuleiten und sich durch eine konsequente Verwendung gerichteter Kanten auch keine Nachteile ergeben.

Auf die Verwendung von parallelen Kanten kann nicht verzichtet werden, zumal sich in vielen Fällen die Notwendigkeit ergibt, zwei Knoten durch mehrere Kanten unterschiedlichen Typs zu verbinden. Dies ist immer dann der Fall, wenn zwei geometrische Elemente durch mehr als nur eine parametrische Zwangsbedingung miteinander verknüpft sind.

Wie bereits in Kapitel 3 erläutert wurde, handelt es sich bei Schleifen um Kanten, deren Anfangs- und Endknoten identisch sind. Diese werden zur Repräsentation parametrischer Zwangsbedingung genutzt, die sich nur auf ein geometrisches Element beziehen.

Abschließend ist anzumerken, dass bei der Festlegung von Eigenschaften des Graphen immer sichergestellt werden muss, dass diese auch in der Graphersetzungssoftware GrGen.NET modelliert werden können.

6.1.2 Verwendung von Knoten und Kanten

Im Graphen werden die geometrischen Elemente in jedem Fall als Knoten und die parametrischen Zwangsbedingungen dementsprechend als Kanten modelliert.

Dies begründet sich dadurch, dass es in der Graphentheorie in sogenannten Multigraphen nur möglich ist, eine Kante zu modellieren, die genau zwei Knoten miteinander verbindet, wobei Anfangs- und Endknoten dabei identisch sein können.
Umgekehrt ist allerdings die Anzahl von Kanten, die zu einem Knoten inzident sind, beliebig bzw. sinnvollerweise immer mindestens eins. Um mehr als zwei Knoten mit einer Kante zu verbinden, müssten sogenannte Hypergraphen eingesetzt werden. Hypergraphen werden allerdings von der Graphersetzungssoftware GrGen.NET nicht unterstützt, daher wurde diese Möglichkeit von Anfang an nicht weiter in Betracht gezogen. Im Verlauf der Erstellung des Graphersetzungssystems ergaben sich auch keine Problemstellungen, deren Lösung die Nutzung von Hypergraphen erfordert hätte.

Für die graphbasierte Repräsentation ist es absehbar, dass in den meisten Fällen die Notwendigkeit besteht, ein geometrisches Element durch parametrische Zwangsbedingungen mit einer Vielzahl weiterer geometrischer Elemente zu verknüpfen. Für eine Zwangsbedingung ist es hingegen ausreichend, wenn sie zwei geometrische Elemente verbindet bzw. nur einem geometrischen Element zugeordnet ist.
Analog besitzen Knoten also die notwendigen Voraussetzungen um geometrische Elemente zu repräsentieren, während sich Kanten gut eignen, um die parametrischen Zwangsbedingungen, die diese Elemente verbinden, abzubilden.
Zu Verdeutlichung dieser Entscheidung ist in der folgenden Abbildung dargestellt, welche Form ein Graph hätte, bei dem Knoten die Zwangsbedingungen und Kanten die geometrischen Elemente repräsentieren:

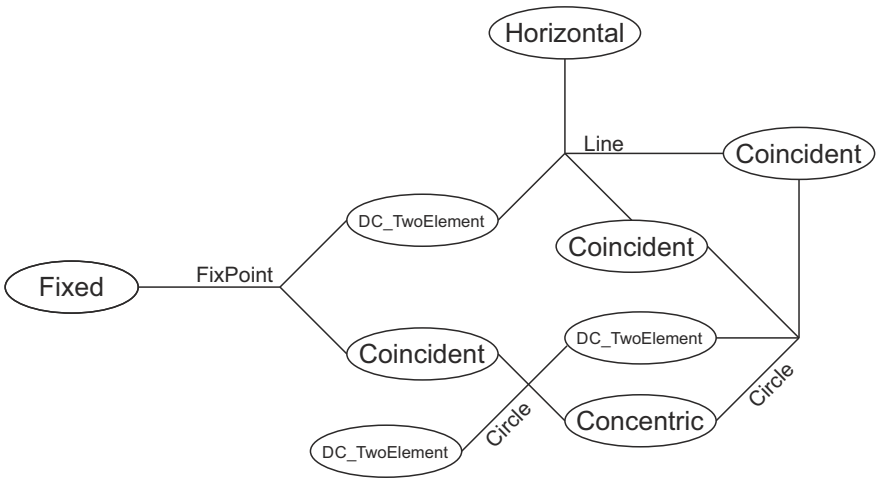

Abbildung 6.1: Form des Graphen bei alternativer Verwendung von Knoten und Kanten

Diese Abbildung repräsentiert die selbe Skizze wie Abbildung 5.3 auf Seite 40. Vergleicht man diese beiden Repräsentationsformen ist klar erkennbar, dass der verfolgte Ansatz ein einfacheres und übersichtlicheres Resultat hervorbringt.
Außerdem ist anzunehmen, dass sich bei der Erstellung der Graphersetzungsregeln durch Hyperkanten weitere nicht absehbare Probleme ergeben.

6.1.3 Andockstellen

Geometrische Elemente müssen sogenannte Andockstellen aufweisen, an denen die parametrischen Zwangsbedingungen *Coincident* und *DC_TwoElement* anknüpfen, um eine Verknüpfung mit einem anderen geometrischen Element herzustellen. Sie sind beispielsweise nötig, damit bei der Verknüpfung einer Linie mit einem Punkt durch die Zwangsbedingung *Coincident* eindeutig definiert ist, ob der Punkt koinzident mit dem Anfangs- oder dem Endpunkt der Linie sein soll.
In (Helms, 2013) werden solche Andockstellen als *Ports* bezeichnet. Dabei ist der Port Bestandteil der Knoten des Graphen. Da die Definition solcher Ports in der hier genutzten Graphersetzungssoftware GrGen.NET nicht vorgesehen ist, wird stattdessen in den Attributen der jeweiligen Zwangsbedingungen bzw. Kanten definiert, an welchen Andockstellen die Zwangsbedingung angreift. Die Werte der Attribute richten sich allerdings immer nach den möglichen Andockstellen eines geometrischen Elements. Dementsprechend muss in der Definition der Eigenschaften dieser geometrischen Elemente festgelegt werden, über welche Andockstellen sie verfügen.
Die tatsächliche Zuweisung der Werte dieser Attribute erfolgt bei der Erstellung der jeweiligen Kanten durch die Graphersetzungsregeln. Dabei muss durch den Anwender, der diese Regeln definiert, sichergestellt werden, dass nur die tatsächlich definierten Andockstellen genutzt werden.

6.1.4 Temporäre Koordinaten

Anfangs wurde versucht, bei der Konzeption des Graphersetzungssystems darauf zu verzichten, den geometrischen Elementen, mit Ausnahme der Fixpunkte, Koordinaten zuzuweisen, um dem unpräzisen Charakter von Skizzen gerecht zu werden.
Wie im Kapitel 2 aber schon angedeutet wurde, können sich dadurch kaum Graphen erzeugen lassen, die bei der Interpretation durch den GCS der parametrischen CAD-Software eine Skizze eindeutig repräsentieren. Dies begründet sich darin, dass es Systeme von Zwangsbedingungen gibt, die mehr als eine richtige Lösung aufweisen - von diesen entspricht aber nur eine der vom Nutzer gewünschten Lösung. Dies wird in Abschnitt 6.4 detaillierter beschrieben und führt zur Einführung der in diesem Abschnitt beschriebenen temporären Koordinaten.

Wird die Skizze nicht automatisch über die API der CAD-Software, sondern über deren graphische Benutzeroberfläche manuell durch einen Anwender erzeugt, der das generelle Aussehen der Skizze bereits kennt, so ergibt sich eine korrekte Lösung.
Dies begründet sich dadurch, dass der Anwender beim Zeichnen der einzelnen geometrischen Elemente bereits ungefähr festlegen kann, an welcher Stelle in der Skizze diese tatsächlich liegen. Durch diese vorläufige Positionierung ergibt sich für den GCS bereits eine annähernd richtige Lösung in Form der vorliegenden Geometrie (Jubierre, 2009). Da der GCS immer die naheliegendste Lösung wählt, kann davon ausgegangen werden, dass die Geometrie nach Definition aller parametrischer Zwangsbedingungen dem gewünschten Ergebnis entspricht.
Bei der automatisierten Erstellung der geometrischen Elemente mit Hilfe von Befehlen der API war in einer ersten Version des G2I-Tools nicht vorgesehen, diese Elemente an ihrer endgültigen Position einzufügen. Stattdessen wurden alle Kreise und Linien der Reihe nach am selben vordefinierten Ort eingefügt. Erst anschließend wurden die parametrischen Zwangsbedingungen der Skizze hinzugefügt, um so eine Positionierung der geometrischen Elemente zu veranlassen. Dieses Vorgehen führt aber zu keiner praktikablen Lösung, da vom GCS infolge der fehlenden relativen Positionierung der geometrischen Elemente zueinander keine eindeutige Lösung gefunden werden konnte.. Beispielhaft hierfür sind die Abbildungen 2.3 (S. 11) und 6.8 (S. 60), da sich hier ohne temporäre Koordinaten verschiedene Lösungen ergeben können, die zwar richtig aber nicht zwangsläufig gewünscht sind.

Es wurde daher versucht, die Informationen zur relativen Lage der geometrischen Elemente zueinander, die sich beim manuellen Anlegen einer Skizze automatisch ergeben, in den Graphen zu integrieren.
Dies ließ sich durch sogenannte temporäre Koordinaten umsetzen. Diese legen für jedes geometrische Element fest, an welcher Position es in der Skizze angelegt werden soll. Gespeichert werden diese temporären Koordinaten als Attribute der Knoten, die die geometrischen Elemente im Graphen repräsentieren. Sie beziehen sich dabei auf die Koordinaten bereits bestehender geometrische Elemente, sodass die neuen Elemente relativ zu diesen positioniert werden.
Diese Koordinaten werden „temporär“ genannt, da sie vom GCS nach erfolgreicher Lösung des Systems der Zwangsbedingungen gegebenenfalls überschrieben werden können.

Durch diese zusätzlichen Informationen können nun die API -Befehle im G2I-Tool so angepasst werden, dass die geometrischen Elemente schon bei der Erstellung annähernd an ihrer finalen Position eingefügt werden, ohne sie jedoch dabei dort fest zu verankern. Dies geschieht erst anschließend durch das Hinzufügen der parametrischen Zwangsbedingungen. Somit ist einerseits die richtige Topologie der Skizze festgelegt und andererseits können Abmessungen und Abstände weiterhin durch das Ändern von Parametern angepasst werden.

Des Weiteren kann von Graphersetzungsregeln auf die Werte der Attribute, die die temporären Koordinaten abbilden, zugegriffen werden. Dies ist nötig, um bei durch Graphersetzungsregeln neu angelegten Knoten wiederum deren temporäre Koordinaten zu berechnen.

6.1.5 Wiederverwendbarkeit von Graphersetzungsregeln

Prinzipiell ist es wünschenswert, dass Graphersetzungsregeln flexibel angewendet werden können.
Konkret ist damit gemeint, dass die Ausführung einer Regel nicht nur exakt eine bestimmte Veränderung einer bestimmten Skizze repräsentieren kann. Stattdessen sollte es möglich sein, die Graphersetzungsregel entweder mehrfach oder auf verschiedene Elemente anzuwenden.
Auf die flexible Verwendbarkeit von Regeln wird in Kapitel 7 im Rahmen der prototypischen Umsetzung weiter eingegangen.

6.1.6 Resultat in korrekter Repräsentation

Die vielleicht wichtigste Schlussfolgerung, die sich im Prozess der Konzeption des Graphersetzungssystems ergeben hat, lautet, dass jeder Graph eine Skizze repräsentieren muss, die auch innerhalb der parametrischen Modellierungssoftware Autodesk Inventor korrekt modelliert werden kann.
Dies ist nicht der Fall, wenn die resultierende Skizze parametrisch über- oder unterbestimmt ist. Eine überbestimmte Skizze liegt vor, wenn zu viele bzw. konkurrierende Zwangsbedingungen definiert sind. Eine solche Skizze kann in Autodesk Inventor dann nicht mehr bearbeitet werden, ohne dass eine oder mehrere Zwangsbedingungen manuell entfernt werden. Eine unterbestimmte Skizze enthält hingegen zu wenige parametrische Zwangsbedingungen, so dass die Skizze bei einer Veränderung eines Parameters nicht mehr korrekt dargestellt wird.
Die Ursache für diese Problematik liegt entweder direkt in der Definition oder in der unplanmäßigen Anwendung einzelner Graphersetzungsregeln.

6.2 Konkrete Definition des Metamodells

Um festzustellen, welche Eigenschaften und Elemente ein Graph, der eine zweidimensionale Skizze repräsentieren soll, aufweisen muss, werden in diesem Abschnitt die Bestandteile einer Skizze in Form von geometrischen Elementen und parametrischen Zwangsbedingungen aufgeführt, beschrieben und analysiert. Es wird differenziert auf jedes einzelne Element und jede einzelne Zwangsbedingung eingegangen, die in ihrer Gesamtheit im Folgenden als die Entitäten der Skizze bezeichnet werden.
Dabei werden die Funktionen der Entitäten in der Skizze untersucht und herausgearbeitet,

welche Informationen in den jeweiligen korrespondierenden Elementen des Graphen enthalten sein müssen, um eine korrekte Repräsentation zu realisieren. Es wird auch darauf eingegangen, wie sich diese Entitäten durch die API von Autodesk Inventor automatisiert in eine Skizze einfügen lassen, und welche Eingabewerte dabei durch die API verlangt werden.

Anschließend wird der Aufbau eines möglichst allgemeingültigen Metamodells beschrieben, das die zuvor definierten generellen Anforderungen an das Graphersetzungssystem erfüllt.

6.2.1 Geometrische Elemente

Die Menge der geometrischen Elemente einer Skizze setzt sich aus dem für einen Anwender sichtbaren Teil dieser Skizze zusammen. Die im Rahmen dieser Thesis untersuchten Skizzen beschränken sich dabei auf Punkte, Linien und Kanten. Auch weitere geometrische Elemente, wie beispielsweise Ellipsen, Bögen oder Splines, könnten durch einen Graphen repräsentiert werden - dies würde allerdings zu einer erheblichen Erweiterung des Graphersetzungssystems und des Software-Tools zur Darstellung des Graphen in Autodesk Inventor führen.

Grundsätzlich werden den geometrischen Elementen keine festen Koordinaten zugewiesen, um sie in der Skizze anzuordnen, da dies dem Ansatz einer möglichst vollständigen Parametrisierung widerspräche. Stattdessen ergibt sich die endgültige Lage der Elemente in der Skizze erst durch die Definition der parametrischen Zwangsbedingungen und die im Abschnitt 6.1.4 eingeführten temporären Koordinaten. Die einzige Ausnahme bilden dabei sogenannte *FixPoints*, die die Lage der gesamten Skizze im Koordinatensystem festlegen. Dies ist nötig, um eine vollständig bestimmte Skizze zu erzeugen, die nicht auf einer beliebigen Position im Koordinatensystem liegen kann.

Allen geometrischen Elementen ist außerdem gemein, dass der jeweilige zur Repräsentation genutzte Knoten ein Attribut besitzen muss, das den Namen des Elements beinhaltet. Dies ist vor allem notwendig, damit eine Graphersetzungsregel, die einen bestimmten Knoten der graphbasierten Repräsentation einer Skizze innerhalb eines Mustergraphen erkennen soll, weiß welcher Knoten gemeint ist. Zusätzlich erleichtert die Benennung der einzelnen geometrischen Elemente die Lesbarkeit des Graphen.

Punkte und Fixpunkte

Punkte sind die einfachsten geometrischen Elemente einer Skizze.

Knoten, die einen Punkt repräsentieren, müssen als Attribut den Namen dieses Punktes beinhalten. Außerdem müssen Attribute vorhanden sein, mit denen die temporären Koordinaten des Punktes definiert werden können.

Einfache Punkte sind für die in dieser Arbeit untersuchten Beispielskizzen nur als Konstruktionselemente notwendig. So können beispielsweise die Enden zweier Linien mit demselben Punkt koinzident gesetzt werden, um sie zu verbinden. Der Punkt an sich ist dabei nur in der graphbasierten Repräsentation relevant, in der eigentlichen Skizze ist er nicht mehr ersichtlich.

Durch den objektorientierten Aufbau des Metamodells ist es außerdem sinnvoll, von der allgemeinen Knotenklasse *Point* die im Vorfeld bereits beschriebene Knotenklasse *FixPoint* abzuleiten. Dadurch können Anforderungen, die sich eventuell an Knoten im Allgemeinen ergeben, ohne Weiteres in alle Knotenklassen integriert werden.
Diese Fixpunkte zeichnen sich durch explizite Koordinaten aus, die auch als Attribute im repräsentierenden Knoten enthalten sein müssen. Es werden dafür die vererbten temporären Koordinaten der übergeordneten Klasse *Point* genutzt.
Es ist weiterhin anzumerken, dass sich durch die Definition eines Punktes als Fixpunkt implizit die Notwendigkeit ergibt, diesen Punkt mit der geometrischen Zwangsbedingung *Fixed* zu versehen. Dadurch wird dieser Punkt fest an seinen Koordinaten verankert und kann nicht mehr in seiner Position verändert werden.

Linien

Gerade Linien sind ein zentraler Bestandteil des im Rahmen dieser Arbeit untersuchten Beispiels, da sich aus ihnen eine Vielzahl von weiteren geometrischen Elementen, wie beispielsweise Polygone, zusammensetzten lassen. Aus diesem Grund wird auch auf die Definition von allen Elementen, die sich aus Linien zusammensetzten lassen, verzichtet.

Knoten, die Linien repräsentieren, müssen neben einem Attribut, das den Namen des Knoten beschreibt, weitere Attribute besitzten, mit denen die Koordinaten des Anfangs- und Endpunktes der Linie definiert werden können.

Linien haben im Gegensatz zu Punkten nicht nur einen Bestandteil, an dem sie beispielsweise durch die geometrische Zwangsbedingung *Coincident* mit einem anderen Element verknüpft werden können. Sie weisen konkret einen Anfangs- und einen Endpunkt auf, durch die eine Linie in der Skizze platziert wird, indem sie durch Zwangsbedingungen mit anderen Elementen verknüpft wird. Zusätzlich wäre es auch denkbar, den Mittelpunkt einer Linie als Andockstelle für Zwangsbedingungen zu definieren. Dies wurde jedoch im Rahmen dieser Arbeit nicht implementiert, da in den behandelten Beispielen keine Notwendigkeit dafür besteht.
Bei einigen Zwangsbedingungen ist es allerdings nicht erforderlich Andockstelle zu definieren. Dies ist beispielsweise bei den Zwangsbedingungen *Parallel* und *Collinear* der Fall.

Durch diese Anfangs- und Endpunkte wird auch die Richtung, in die eine Linie „zeigt“, in Autodesk Inventor festgelegt. Bei der Konzeption von Ersetzungsregeln, die in der Skizze

Linien oder Zwangsbedingungen, die Linien beeinflussen, erzeugen bzw. verändern, muss deshalb immer beachtet werden, dass keine Widersprüche oder Inkonsistenzen bei Definition von oder dem Zugriff auf Anfangs- und Endpunkten entstehen.

Kreise

Kreise definieren sich über ihren Mittelpunkt und ihren Radius. Als Andockstellen für Zwangsbedingungen kommt entweder der Mittelpunkt oder der Kreisbogen in Frage.
Dabei ist der Kreisbogen als Andockstelle insofern problematisch, als dass sich über nur eine Zwangsbedingung nicht definieren lässt, welche Stelle des Kreisbogens gemeint ist. Es muss hier dementsprechend bei der Konzeption von Graphersetzungsregeln beachtet werden, dass ausreichend weitere Zwangsbedingungen erstellt werden, die eine eindeutige Definition ermöglichen.

Neben Attributen, die die temporären Koordinaten des Kreismittelpunktes definieren, muss außerdem ein Attribut vorhanden sein, das für den Kreis die temporäre Länge des Radius definiert.

6.2.2 Parametrische Zwangsbedingungen

Betrachtet man eine fertige Skizze sind die geometrischen Zwangsbedingungen für den Nutzer normalerweise nicht mehr sichtbar. Sie wirken nur im Hintergrund, um die Lage der geometrischen Elemente zu definieren. Es ist nur dann relevant sie anzuzeigen, wenn der Nutzer den parametrischen Aufbau der Skizze nachvollziehen oder verändern möchte.

Parametrische Zwangsbedingungen werden, wie schon in Kapitel 4 beschrieben, in dimensionale und geometrische Zwangsbedingungen aufgeteilt. Für die Repräsentation im Graphen hat sich außerdem erwiesen, dass es sinnvoll ist, zu unterscheiden, ob eine Zwangsbedingung nur die Positionierung oder Größe eines geometrisches Elements definiert, oder ob sie zwei verschiedene geometrische Elemente miteinander verknüpft.
Dies resultiert in der graphbasierten Darstellung entweder in einer Schleife, bei der eine Kante denselben Knoten als Anfangs- und Endknoten verbindet oder in der normalen Verbindung zweier verschiedener Knoten durch eine Kante.

Dimensionale Zwangsbedingungen

Dimensionale Zwangsbedingungen, die nur auf ein geometrisches Element angewandt werden, definieren die Größe dieses Elements. Sie werden im Weiteren als *DC_OneElement* bezeichnet (*DC* steht dabei für *dimensional constraint*).
Bei einer Linie legen sie deren Länge fest, die dabei formal durch den Abstand zwischen dem

Anfangs- und dem Endpunkt definiert ist. Außerdem wird durch sie der Radius eines Kreises festgelegt.

Verknüpft eine dimensionale Zwangsbedingung zwei verschiedene geometrische Elemente, wird prinzipiell der Abstand zwischen diesen Element festgelegt und im Folgenden mit *DC_TwoElement* bezeichnet. Je nach Art der beiden geometrischen Elemente ergeben sich hier verschiedene Möglichkeiten hinsichtlich der tatsächlichen Bedeutung der Zwangsbedingung. Um diese Möglichkeiten eindeutig unterscheiden zu können, kommen die oben genannten Andockstellen zum Einsatz. An dieser Stelle werden die verschiedenen Fälle, die sich so ergeben können, aufgeführt:

- **Punkt - Punkt:** Definition des Abstands zweier Punkte. Es müssen keine Andockstellen festgelegt werden.
- **Linie - Linie:** Definition des Abstands zwischen zwei Linien. Je nachdem welche Andockstellen in den Attributen der Zwangsbedingungen festgelegt sind, ist beschreibt die Zwangsbedingung entweder den Abstand zwischen zwei parallelen Linien oder den Abstand zwischen den Anfangs- oder Endpunkten der Linien.
- **Kreis - Kreis:** Hier kann entweder der Abstand zwischen den Mittelpunkten zweier Kreise oder der Versatz zweier konzentrischer Kreise gemeint sein.
- **Punkt - Linie:** Es ist möglich, entweder den lotrechten Abstand zwischen einem Punkt und einer Linie oder den Abstand zwischen einem Punkt und dem Anfangs- oder Endpunkt einer Linie zu definieren.
- **Punkt - Kreis:** Abstand des Punktes zum Mittelpunkt des Kreises.
- **Linie - Kreis:** Abstand zwischen der Linie und dem Mittelpunkt des Kreises.
  Siehe *Punkt - Linie*.

Alle Kanten, die dimensionale Zwangsbedingungen repräsentieren, müssen außerdem verschiedene Attribute aufweisen.
Zum einen ist festzulegen, ob es sich um eine getriebene Zwangsbedingung handelt, d.h. ob die Zwangsbedingung durch einen Wert definiert ist, oder ob sie nur eine Bemaßung darstellt, deren Größe durch die bemaßten Objekte definiert wird. Handelt es sich um eine getriebene dimensionale Zwangsbedingung, ist es nicht nötig ihr einen Wert zuzuweisen.
Ist die Zwangsbedingung hingegen nicht getrieben, so muss ihre Größe durch einen Wert bestimmt sein. Dieser Wert kann entweder durch einen numerischen Wert fest bestimmt oder durch eine mathematische Funktion ausgedrückt sein, deren Ergebnis anhand der Größe der Werte anderer dimensionaler Zwangsbedingungen berechnet wird. Dies gilt sowohl für Zwangsbedingungen, die sich nur auf ein geometrisches Element beziehen, als auch für solche, die zwei Elemente verknüpfen.

Wie in diesem Abschnitt weiter oben schon beschrieben, müssen bei Kanten, die Zwangsbedingungen des Typs *DC_TwoElement* repräsentieren, außerdem jeweils zwei Attribute existieren, die festlegen, welche Andockstelle der Elemente, die durch sie verknüpft werden, genutzt werden soll.
Außerdem ist es teilweise nötig, einer Zwangsbedingung einen Namen zuzuweisen. Dies ist immer dann der Fall, wenn eine Abhängigkeit zu einer anderen Zwangsbedingung besteht (siehe Abschnitt 4.1.1 auf S. 29). Kanten müssen für diesen Fall also ein weiteres Attribut aufweisen.

Geometrische Zwangsbedingungen

Geometrische Zwangsbedingungen beschreiben entweder die Lage zweier Elemente zueinander oder die Positionierung eines einzelnen Elements.
Kanten, die diese Zwangsbedingungen repräsentieren, müssen nur in wenigen Fällen Attribute aufweisen. Im Folgenden werden zuerst die drei geometrischen Zwangsbedingungen, die sich nur auf ein Element beziehen können, beschrieben. Anschließend wird auf solche eingegangen, die zwei Elemente verbinden.

Die Bedingungen *Horizontal* und *Vertical* beziehen sich immer auf eine Linie und richten diese im Koordinatensystem parallel zur x- bzw. zur y-Achse aus.

Wird die Bedingung *Fixed* auf ein geometrisches Element angewandt, ist dieses Element weder in seiner Position im Koordinatensystem noch in seinen Abmessungen veränderbar. Um eine Skizze vollständig zu parametrisieren, muss mindestens einem geometrischen Element der Skizze diese Zwangsbedingung zugewiesen werden.

Die Zwangsbedingungen *Collinear*, *Parallel* und *Perpendicular* verbinden immer zwei Linien. *Collinear* definiert, dass zwei Linien auf derselben Gerade liegen, während *Parallel* und *Perpendicular* die Linien entweder parallel oder orthogonal zueinander ausrichten.

Durch die Zwangsbedingung *Concentric* werden zwei Kreise miteinander verknüpft. Diese Kreise werden durch die Zwangsbedingung vom selben Mittelpunkt abhängig gemacht.

Sollen zwei Kreise oder Linien den gleichen Radius oder die gleiche Länge haben, kann die Zwangsbedingung *Equal* genutzt werden.

Die im Rahmen der in dieser Arbeit untersuchten Beispiele am häufigsten genutzte Zwangsbedingung ist die des Typs *Coincident*. Diese Bedingung legt fest, dass zwei geometrische Elemente koinzident sind, also dass sie am selben Ort liegen müssen.
Ähnlich wie bei der dimensionalen Zwangsbedingung *DC_TwoElement* ist es nötig zu definieren, welche Andockstellen der verknüpften geometrischen Elemente durch die Zwangsbedingung koinzident gesetzt werden. Je nach Art der geometrischen Elemente ergeben sich auch hier verschiedene Möglichkeiten:

- **Punkt - Punkt:** Zwei Punkte haben dieselbe Position. Es müssen keine Andockstellen definiert werden.
- **Linie - Linie:** Anfangs- oder Enpunkte zweier Linien fallen aufeinander. Es muss festgelegt werden welche Punkte durch die Zwangsbedingung koinzident gesetzt werden.
- **Kreis - Kreis:** Der Mittelpunkt eines Kreises liegt auf dem Kreisbogen des anderen Kreises. Zwei Kreise deren Mittelpunkte koinzident sind, sind bereits durch die Zwangsbedingung *Concentric* abgedeckt.
- **Punkt - Linie:** Ein Punkt liegt auf dem Anfangs- oder Endpunkt einer Linie. Es muss definiert sein, welcher Punkt der Linie gemeint ist.
- **Punkt - Kreis:** Ein Punkt ist entweder koinzident mit dem Mittelpunkt eines Kreises oder der Punkt liegt auf dem Kreisbogen des Kreises. In den Attributen der Kante, die die Zwangsbedingung repräsentiert, muss festgelegt werden, ob der Punkt auf dem Mittelpunkt oder auf dem Kreisbogen liegen soll.
- **Linie - Kreis:** Der Anfangs- oder Endpunkt einer Linie liegt entweder auf dem Mittelpunkt eines Kreises oder auf dem Kreisbogen des Kreises. Auch hier muss explizit festgelegt werden, welche dieser Punkte gemeint sind.

In diesem und dem vorangegangenen Abschnitt wurden geometrische Elemente und parametrische Zwangsbedingungen beschrieben, aus denen sich eine Skizze zusammensetzen kann. Dabei wurden die Anforderungen, die sich bei der graphbasierten Repräsentation dieser Elemente und Zwangsbedingungen ergeben, formuliert. Auf dieser Basis kann nun ein Metamodells erstellt werden. In den Abbildungen 6.2 und 6.3 ist der Aufbau dieses Metamodells getrennt für Knoten und Kanten dargestellt.

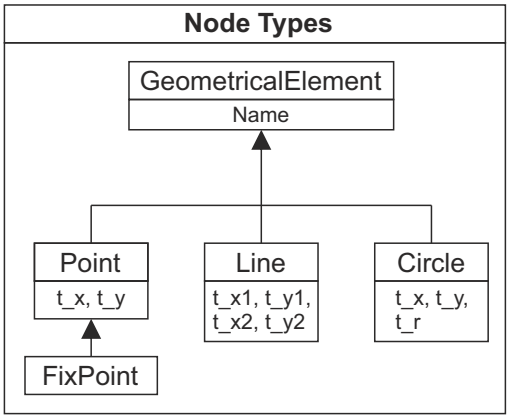

Abbildung 6.2: Hierarchische Darstellung der im Metamodell definierten Knoten

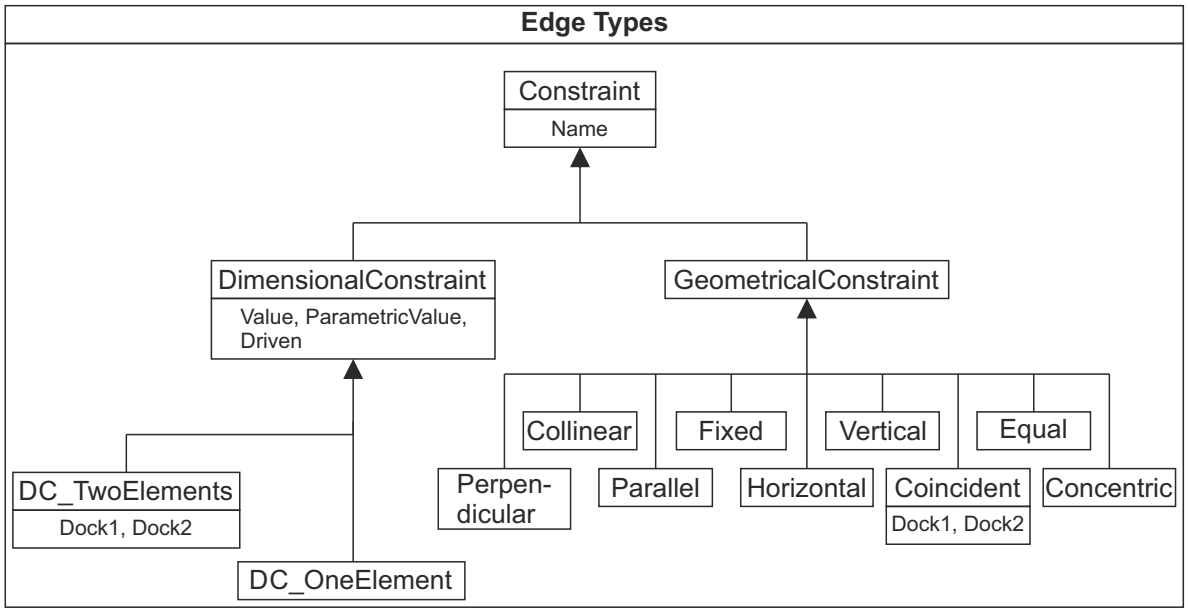

Abbildung 6.3: Hierarchische Darstellung der im Metamodell definierten Kanten

6.3 Definition von Graphersetzungsregeln

Im vorangegangenen Teil dieses Kapitels wurden bereits die finale Definition des Metamodells und die Überlegungen, die zu dieser Definition geführt haben, beschrieben. Dabei wurde auch schon an verschiedenen Stellen erwähnt, inwiefern dieses Metamodell in einer direkten Beziehung zu den Graphersetzungsregeln steht, und wie sich diese beiden Teile des Graphersetzungssystems gegenseitig beeinflussen können.
In diesem Abschnitt werden nun die generellen Anforderungen, die sich auf Basis der untersuchten Detaillierungsvorgänge an die Graphersetzungsregeln im Allgemeinen stellen, erläutert. Diese werden anhand von Aufgabe, Aufbau und Beispielen von Graphersetzungsregeln hergeleitet. Es wird dabei auch ein direkter Bezug zu den in Kapitel 3 recherchierten Grundlagen hergestellt, da diese die theoretische und formale Basis für die Definition der Graphersetzungsregeln bilden.

6.3.1 Aufgabe der Graphersetzungsregeln

Es ist prinzipiell möglich, auf Basis eines Metamodells manuell einen Graphen zu erstellen, indem Knoten erstellt und durch Kanten verbunden werden, um ihnen anschließend Attribute zuzuweisen.
Durch Graphersetzungsregeln kann dieses manuelle Vorgehen automatisiert werden. Konkret bedeutet dies, dass auf einen Graphen, der eine bestimmte Skizze repräsentiert, eine Ersetzungsregel angewendet wird, die ihn so verändert oder erweitert, dass ein neuer Graph

entsteht. Dieser neue Graph repräsentiert dann eine veränderte bzw. detailliertere Version der Skizze.
Die Automatisierung dieses Vorgangs der Veränderung bzw. der Erweiterung des Graphen ist die eigentliche Aufgabe der Graphersetzungsregeln.

6.3.2 Konzeption der Graphersetzungsregeln

Formal besteht eine Graphersetzungsregel aus einem Mustergraphen und einem Ersetzungsgraphen, in dem gegebenenfalls auch Attribute von Knoten und Kanten zugewiesen bzw. berechnet werden. Um eine Graphersetzungsregel zu erstellen, ist es also in erster Linie nötig, diese Muster- und Ersetzungsgraphen zu definieren.
Dazu muss allerdings zuerst festgestellt werden, welchem Zweck eine Graphersetzungsregel dient, d.h. wie sie den Graphen verändern oder erweitern soll. Dies bedeutet prinzipiell, dass der Arbeitsgraph und der modifizierte Arbeitsgraph, also das gewünschte Ergebnis der Graphersetzung, verglichen werden müssen. Denn nur so ist feststellbar, welche Teile des Arbeitsgraphen modifiziert werden sollen und wie diese Modifikation aussehen kann.

Im Konkreten bedeutet dies im Rahmen dieser Arbeit, dass für die Erstellung einer Graphersetzungsregel zwei Skizzen, bzw. die graphbasierte Repräsentation dieser Skizzen, gegenübergestellt werden. Die Graphen, die diese Skizzen repräsentieren, entsprechen dem Arbeitsgraphen und dem modifizierten Arbeitsgraphen.
Die beiden Skizzen sind auf der einen Seite die Ausgangsskizze, die verändert wird, und andererseits die modifizierte Version dieser Skizze, die letztendlich als Ergebnis aus der Graphersetzungsoperation hervorgehen soll.

Diese Zusammenhänge sind in der folgenden Abbildung dargestellt:

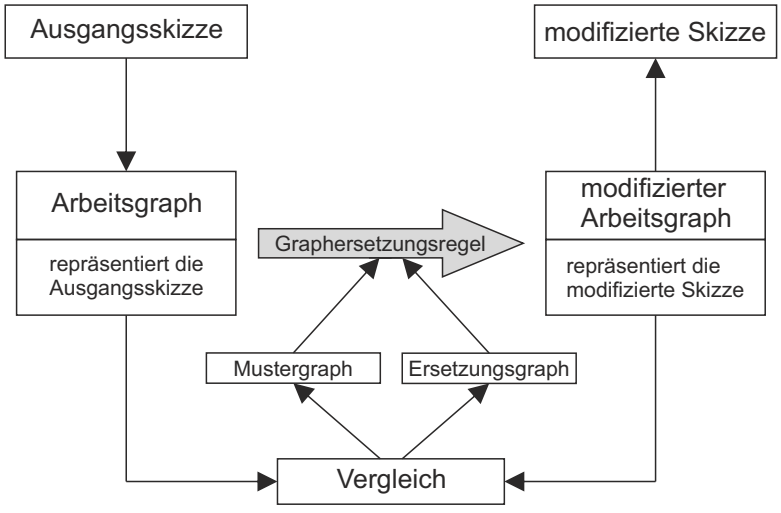

Abbildung 6.4: Vorgehen zur Definition einer Graphersetzungsregel

Aus diesem Vergleich des Arbeits- und des modifizierten Graphen kann nun hergeleitet werden, welche Teile des Arbeitsgraphen die Graphersetzungsregel überhaupt beeinflusst, mithin welche Knoten und Kanten überhaupt gelöscht, neu erstellt oder verändert werden müssen. Durch diese Herleitung kann nun der Mustergraph definiert werden. Er muss alle Bestandteile der graphbasierten Repräsentation der Ausgangsskizze beinhalten, die entweder selbst modifiziert werden oder deren Attribute für den Ersetzungsvorgang relevant sind.

Als nächstes ist zu analysieren, welche Knoten und Kanten im modifizierten Arbeitsgraphen neu erstellt werden müssen und was mit den im Mustergraphen enthaltenen Knoten und Kanten geschehen soll. Diese Knoten und Kanten, die schon im Arbeitsgraphen enthalten sind, können entweder gelöscht, unverändert beibehalten oder modifiziert werden. Modifiziert werden dabei in erster Linie die Attribute. Kanten können außerdem „umgelenkt“ werden - dabei wird ihnen ein anderer Start- oder Zielknoten zugewiesen.
Auf Basis dieser Analyse ist es anschließend möglich, den Ersetzungsgraphen zu definieren. In ihm muss festgelegt werden, welche Knoten und Kanten neu erstellt werden und wie diese untereinander und mit den im Mustergraphen enthaltenen Knoten des Arbeitsgraphen verbunden sind. Zusätzlich muss auch angegeben sein, wie generell weiter mit den bereits bestehenden, also durch den Mustergraphen definierten Knoten, verfahren wird.
Weiterhin müssen den neu erstellten Knoten und Kanten Attribute zugewiesen, sowie die Attribute von bereits bestehenden Knoten und Kanten gegebenenfalls verändert werden. Diese Attribute können dabei entweder direkt definiert oder auf Basis von Attributen der Knoten und Kanten, die bereits im Mustergraph enthalten sind, übernommen bzw. im Fall von numerischen Werten auch neu berechnet werden.

Aus diesen Definitionen von Muster und Ersetzungsgraphen lässt sich schließlich die eigentliche Graphersetzungsregel zusammensetzen.

6.3.3 Beispielhafter Aufbau einer Graphersetzungsregel

In Abbildung 6.5 ist ein Detaillierungsschritt der vereinfachten Skizze eines Tunnelquerschnitts dargestellt.

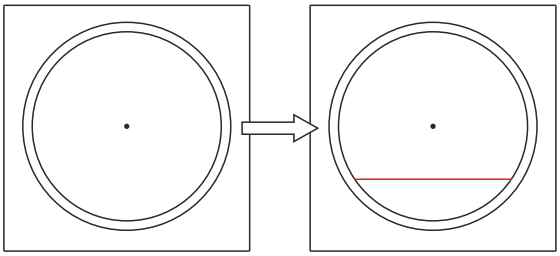

Abbildung 6.5: Detaillierungsschritt

Dieser Detaillierungsvorgang soll nun, auf Basis des schon beschriebenen Metamodells, durch eine Graphersetzungsregel dargestellt werden, die auf die graphbasierte Repräsentation der linken Skizze angewandt wird und so einen Graphen erzeugt, der die rechte Skizze repräsentiert. Dazu muss zum einen die in Rot dargestellte Linie als neuer Knoten in den Graph eingefügt werden, die diesen zum anderen über mehrere neu zu erstellende Kanten mit dem bereits bestehenden Graphen verbinden.
Zur Veranschaulichung dieses Vorgangs wird in der folgenden Abbildung 6.6 eine Regel formal dargestellt, die diesen Vorgang durchführen kann. In der Abbildung nicht dargestellt sind dabei die Attribute der einzelnen Knoten und Kanten.

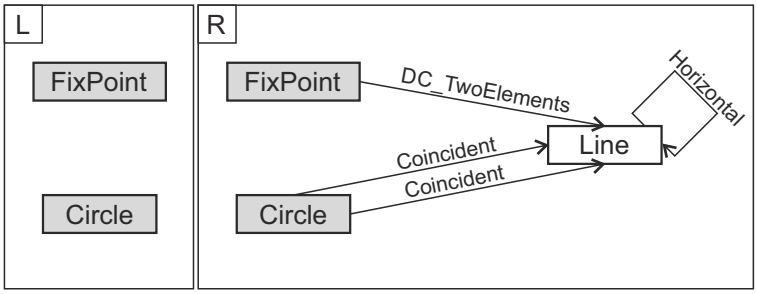

Abbildung 6.6: Ersetzungsregel, durch die eine horizontale Linie, deren Endpunkte koinzident mit einem bereits bestehenden Kreisbogen sind, in eine Skizze eingefügt wird

Auf der linken Seite sind als Mustergraph die Knoten dargestellt, die bei der Ausführung der Ersetzungsregel im bereits existierenden Graphen gesucht werden. An diese beiden Knoten werden nun die auf der rechten Seite der Abbildung als Ersetzungsgraph dargestellten Knoten und Kanten angehängt und der Graph dadurch erweitert.
In der folgenden Abbildung 6.7 ist dargestellt, wie sich die Anwendung der Regel auf den Graphen, der die linke Skizze repräsentiert, auswirkt und so festlegt, welche Form der resultierende modifizierte Arbeitsgraph hat.

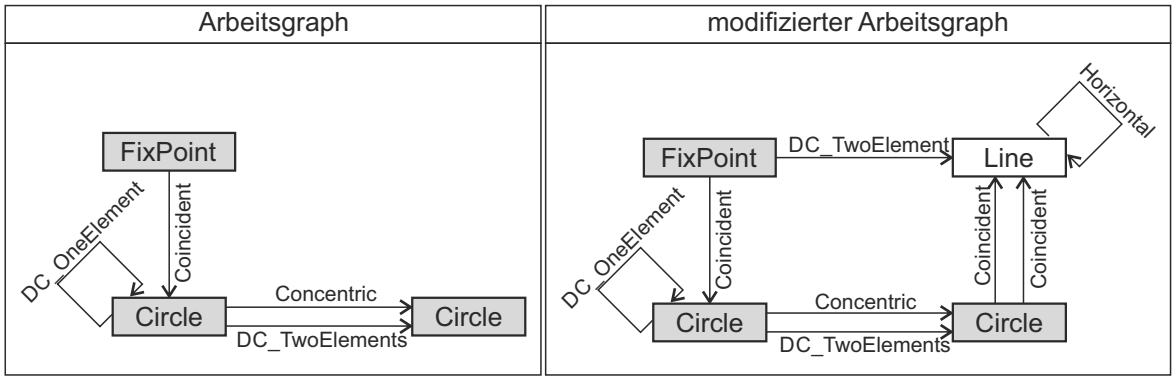

Abbildung 6.7: Anwendung der in Abbildung 6.6 dargestellten Ersetzungsregel auf den Graphen

In den beiden vorangegangenen Abbildungen 6.6 und 6.7 ist allerdings nur die generelle Struktur des Graphen dargestellt. Erst durch durch die Attribute der Knoten und Kanten entsteht eine Repräsentation, aus der auch tatsächlich eine eindeutige Skizze abgeleitet werden kann.

In der Regel muss deshalb auch definiert sein, welche Werte den Attributen der neu erstellten Knoten und Kanten zugewiesen werden. Die erforderlichen Zuweisungen von Attributen werden im Folgenden aufgeführt.

Den Attributen der Kante des Typs *DC_TwoElement* müssen Werte für einen Namen (der hier beliebig gewählt werden kann), ein Größe und die Bezeichnungen der Andockstellen zugewiesen werden. Außerdem muss definiert sein, dass es sich nicht um eine getriebene Bemaßung handelt. Dies geschieht, indem der Wert des Attributs *Driven* als *false* angegeben wird.
Hinsichtlich der Andockstellen muss nur dem Attribut *Dock2* der Kante ein Wert zugewiesen werden, da bei dem durch den Anfangsknoten *FixPoint* repräsentierten geometrischen Element „Punkt“ die Andockstelle implizit definiert ist.
Wie in Abschnitt 6.2.2 beschrieben, kann es sich bei der durch diese Kante repräsentierte Zwangsbedingung entweder um den Abstand zwischen dem Fixpunkt und dem Anfangs- oder Endpunkt der Linie oder um den kürzesten (also zur Linie lotrechten) Abstand zwischen dem Punkt und der Linie an sich handeln. Da hier der lotrechte Abstand zwischen Punkt und Linie gemeint ist, erhält das Attribut *Dock2* den Wert *Line*.
Wäre der Abstand zum Anfangs- oder Endpunkt der Linie gemeint, müsste als Wert alternativ entweder *StartPoint* oder *EndPoint* zugewiesen werden.

Auch bei den beiden Kanten des Typs *Coincident* muss durch die Attribute *Dock1* und *Dock2* definiert werden, welche Andockstellen der geometrischen Elemente gemeint sind. Dem Attribut *Dock1* muss dabei bei beiden Kanten der Wert *Arc* zugewiesen werden. Das Attribut *Dock2* der einen Kante erhält den Wert *StartPoint*, das Attribut der anderen Kante den Wert *EndPoint*.
So wird sichergestellt, dass Anfangs- und Endpunkt der Linie auf dem Kreisbogen des Kreises liegen.

Dem Knoten des Typs *Line*, der durch die Regel neu erstellt wird, müssen ein Wert für sein Attribut *Name* und die Werte der temporären Koordinaten des Start- und Endpunktes der Linie zugewiesen werden. Während der Name der Linie frei gewählt werden kann, müssen die temporären Koordinaten innerhalb der Graphersetzungsregel aus den schon vorhandenen temporären Koordinaten der beiden im Mustergraphen enthaltenen Knoten und dem Wert der Kanten *DC_TwoElements* berechnet werden.

Für die Kante des Typs *Horizontal* ist keine Zuweisung erforderlich.

6.4 Vorgehen zur Erstellung der Skizze in Autodesk Inventor

Im Folgenden wird schrittweise dargestellt, wie generell bei der Verarbeitung der graphbasierten Daten zur Erstellung einer tatsächlichen Skizze in Autodesk Inventor vorgegangen wird.

Auf die programmiertechnische Umsetzung und die genutzte Programmiersprache, sowie die eingebundenen Bibliotheken wird im nächsten Kapitel eingegangen.

Die Schritte, die durchgeführt werden, um aus dem Graphen eine Skizze in Autodesk Inventor zu erstellen, sind im Folgenden aufgelistet:

1. Export der Daten des erstellten Graphen aus der Graphersetzungssoftware GrGen.NET in ein von diesem Programm unterstütztes Datenformat. Konkret wird die Graph eXchange Language (GXL) verwendet (siehe Abschnitt 3.2.5).
2. Einlesen der exportierten Daten in das erstellte Programm.
3. Umwandlung der Daten innerhalb des G2I-Tools in eine für die weitere Verarbeitung sinnvolle Struktur. Mit den Daten der Knoten und Kanten werden dabei Instanzen entsprechender Klassen innerhalb des G2I-Tools erzeugt.
4. Zugriff des Tools auf eine laufende Instanz der CAD-Software Autodesk Inventor durch die entsprechende API.
5. Aufruf geeigneter Methoden der API, die die geometrischen und parametrischen Elemente als Skizze aufbauen.
6. Anzeige der fertiggestellten Skizze in der CAD-Software.

Die Durchführung der Punkte 1 bis 4 stellt sich dabei als weniger herausfordernd heraus, während in Punkt 5 die eigentliche Problematik liegt. Denn hier erfolgt sequentiell die Interpretation der aus dem Graph exportierten Daten. In einem ersten Versuch wurde dabei folgendermaßen vorgegangen:

Zuerst werden die nötigen geometrischen Elemente über entsprechende Funkionen der API in eine leere 2D-Skizze im Inventor eingefügt. Dabei werden jeweils der Reihe nach die Elemente eines bestimmten Typs gezeichnet. Beispielsweise zuerst Punkte, dann Linien und anschließend Kreise. Jedes Element wird dabei einzeln über einen bestimmten API-Befehl gezeichnet - die Elemente entstehen in der Skizze nacheinander. Außerdem wird jedem Element dabei eine Variable im G2I-Tool zugewiesen und ist anhand dieser für weitere Operationen identifizierbar.
Die geometrischen Elemente werden dabei, mit Ausnahme von Fixpunkten, nicht an ihrer endgültigen Position eingefügt, da die definitive Positionierung erst durch die parametrischen Zwangsbedingungen erfolgt. Bei der Konstruktionsoperation werden aber durch den Inventor Koordinaten gefordert, insofern werden die jeweiligen Elemente auf eine temporäre Position in die Skizze gezeichnet.

Im nächsten Schritt werden die dimensionalen und geometrischen Zwangsbedingungen in die Skizze eingefügt. Auch hierbei werden nacheinander die Zwangsbedingungen eines be-

stimmten Typs abgearbeitet. Je nach Art der Zwangsbedingung wird dabei auf ein oder zwei geometrische Elemente verwiesen, die so miteinander verknüpft werden. Bei dimensionalen Zwangsbedingungen wird dem zugehörigen Parameter zusätzlich sein entsprechender Wert zugewiesen.
Bei diesem Vorgang wird die Position der geometrischen Elemente teils schon verändert, die finalen Positionen ergeben sich jedoch erst nach Erstellung aller parametrischer Zwangsbedingungen.

Nachdem alle Konstruktionsoperationen durchgeführt wurden, wird die Skizze mit einem *refresh*-Befehl neu berechnet und erhält erst so ihre finale Gestalt.

An dieser Stelle manifestiert sich das schon in Kapitel 2 beschriebene Problem der Interpretation durch den GCS. Da je nach Art der darzustellenden Skizze meist mehrere richtige Lösungen möglich sind, wird vom GCS die naheliegendste gewählt. Dadurch kann es zu einem Ergebnis kommen, das bei der Erstellung des Graphen nicht vorgesehen war und nicht der gewünschten Skizze entspricht.

Ein Grund hierfür liegt in der temporären Anordnung der geometrischen Elemente, die sich wesentlich von der endgültigen Positionierung unterscheiden können. Dies lässt sich an Hand der folgenden Abbildung verdeutlichen, da sich je nach ursprünglicher Position der einzelnen Linien zwei Lösungen ergeben können. Liegen diese oberhalb der endgültigen Position des Dreiecks, die durch die Endpunkte der horizontalen blauen Linie vorgegeben sind, zeigt die Spitze des Dreiecks nach oben. Liegen sie unterhalb zeigt es nach unten.

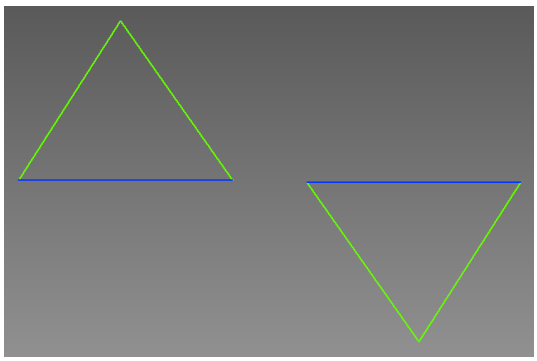

Abbildung 6.8: Unterschiedliche Ergebnisse der graphbasierten Repräsentation eines Dreiecks

Hieraus ergibt sich die Fragestellung, wie diese Mehrdeutigkeit der graphbasierten Repräsentation vermieden werden kann. Da sich eine unerwünschte Lösung auch durch die ursprüngliche Position, an der die geometrischen Elemente während des Konstruktionsvorgangs platziert werden, ergeben kann, liegt hier ein potentieller Ansatzpunkt für eine Lösungsstrategie.
Im Rahmen dieser Arbeit wurde der Ansatz verfolgt, im Graphen für die geometrischen Elemente temporäre Koordinaten ihrer Konstruktionsposition anzugeben und sie so schon direkt oder zumindest annäherungsweise in ihrer endgültigen Lage anzuordnen. Für diese Methode

müssen sowohl das Metamodell als auch die Ersetzungsregeln des Graphersetzungssystems erweitert werden. Zusätzlich ist es nötig, im G2I-Tool die Konstruktionsoperationen anzupassen.
Durch die im folgenden Kapitel beschriebene prototypische Umsetzung konnte die Funktionsweise dieses Lösungsansatzes erfolgreich überprüft werden.
Im Falle des Beispiels in Abbildung 6.8, kann die Ausrichtung des Dreiecks durch die temporären Koordinaten der Anfangs- und Endpunkte der grünen Linien beeinflusst werden.

Als Nachteil ergeben sich allerdings wesentlich mehr Informationen im Graph und dadurch dementsprechend kompliziertere Ersetzungsregeln, deren Erstellung einen erhöhten Aufwand erfordert.
Die Einführung solcher temporären Koordinaten wurde bereits in Abschnitt 6.1.3 beschrieben und die temporären Koordinaten schon in das Metamodell integriert. Da dies die Lösung einer der wesentlichen Problematiken der graphbasierten Repräsentation und der Visualisierung darstellt, wurde die Vorgehensweise, die zu einer funktionsfähigen Lösung geführt hat, in diesem Abschnitt trotzdem explizit beschrieben.

6.5 Zusammenfassung

Dieses Kapitel beinhaltete die Ansätze und Ideen mit denen eine erfolgreiche eindeutige graphbasierte Repräsentation von Skizzen und deren Detaillierung umgesetzt werden kann.
Auf Basis des in Kapitel 5 beschriebenen methodischen Vorgehens konnten verschiedene grundsätzliche Entscheidungen hinsichtlich der Anforderungen an die Eigenschaften des Graphen, der für die Repräsentation genutzt wird, herausgearbeitet werden. So konnte ein Metamodell definiert werden, das alle nötigen Typen von Knoten und Kanten beinhaltet, die für die Erstellung eines prototypischen Graphersetzungssystem, entsprechend dem in Kapitel 2 aufgeführten Beispiel, nötig sind.
Weiterhin wurde der grundlegende Aufbau von nutzbaren Graphersetzungsregeln definiert und außerdem beschrieben, wie vorgegangen werden soll, um aus einem Graphen eine Skizze in Autodesk Inventor zu erstellen.

Kapitel 7

Prototypische Umsetzung

Das folgende Kapitel gliedert sich in zwei Abschnitte.

Im ersten Abschnitt 7.1 wird die praktische Umsetzung des im vorigen Kapitel erarbeiteten Graphersetzungssystems dargestellt. Dabei wird zuerst die Implementierung des Metamodells in GrGen.NET beschrieben. Anschließend wird auf die Erstellung der Graphersetzungsregeln eingegangen.

Als konkretes Beispiel wird weiterhin auf die vereinfachte Darstellung eines Tunnelquerschitts in verschiedenen Detaillierungsstufen zurückgegriffen. Um die Variabilität einzelner Ersetzungsregeln zu demonstrieren, werden dabei allerdings die in folgender Abbildung 7.1 dargestellten Detaillierungsstufen bzw. Detaillierungsschritte verwendet. Ab Schritt 4 gibt es hier zwei verschiedene Varianten für den Detaillierungsvorgang.

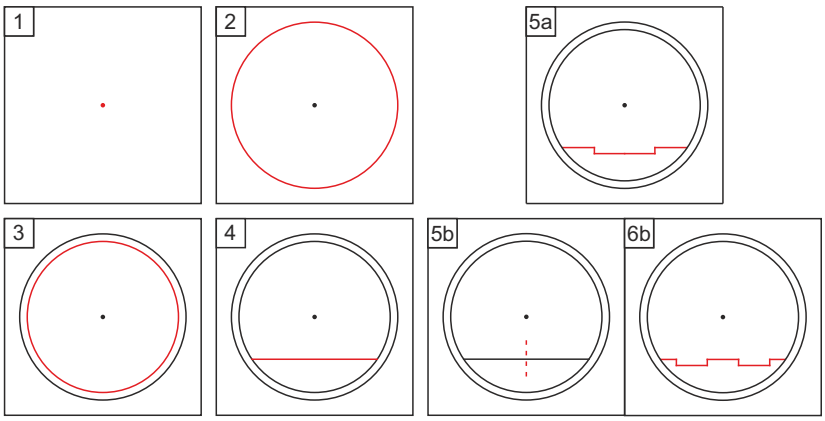

Abbildung 7.1: Detaillierungsvorgang der Skizze, der in diesem Kapitel als Graphersetzungssystem umgesetzt wird

Im zweiten Abschnitt 7.2 wird die Entwicklung des G2I-Tools, das die graphbasierten Daten einlesen und in Autodesk Inventor darstellen kann, beschrieben.

7.1 Programmierung des Graphersetzungssystems in GrGen.NET

Um in GrGen.NET ein Graphersetzungssystem zu programmieren muss, wie in Kapitel 3 Abschnitt 3.2 beschrieben, ein Metamodell in Form einer *graph model description file* erstellt werden.
Aufbauend auf diesem Metamodell werden anschließend die einzelnen Graphersetzungsregeln in einer *rule set file* definiert.
Um die Regeln dann auszuführen, können diese letztendlich in der GrShell aufgerufen werden. In diesem Beispiel wird dies durch ein *graph rewrite script* durchgeführt, in dem die Befehle, die sonst einzeln in die GrShell eingegeben werden müssten, enthalten sind.

Es müssen demnach drei Textdateien erstellt werden mit denen GrGen.NET arbeitet:

- Die Datei *metamodel.gm* beinhaltet das Metamodell.
- In der Datei *rules.grg* werden die einzelnen Graphersetzungsregeln definiert.
- Die Befehle, die in der GrShell die Graphersetzungsoperationen einleiten, sind in der Datei *example.grs* aufgeführt.

Anhand des Aufbaus dieser drei Dateien wird im Folgenden die Umsetzung des Graphersetzungssystems in GrGen.NET beschrieben.

7.1.1 Metamodell

Eine schematische Darstellung des Metamodells wurde bereits in Kapitel 6 erarbeitet und ist in den Abbildeungen 6.2 und 6.3 (S. 54) getrennt für Knoten und Kanten dargestellt.

Der Übersichtlichkeit halber wird die Konzeption der Datei *metamodel.gm*, die das Metamodell beinhaltet, getrennt für Knoten und Kanten beschrieben.
Dabei ist zu beachten, dass die Begriffe „Typ“ und „Klasse“ im Rahmen des Metamodells als synonym zu betrachten sind. Es kann also sowohl vom Typ eines Knotens als auch von der Klasse eines Knotens gesprochen werden. Für Kanten gilt dies analog.

Im folgenden Codebeispiel ist zuerst die Definition der einzelnen Knotentypen bzw. Knotenklassen aufgeführt.
Als erstes wird dabei die abstrakte Klasse *GeometricalElement* definiert. Von dieser Klasse werden die Klassen abgeleitet, die in der graphbasierten Repräsentation die geometrischen Elemente der Skizze beschreiben. Da *GeometricalElement* das Attribut *Name* besitzt, in dem die Bezeichnung des geometrischen Elements als *string* abgespeichert werden kann, verfügen

auch alle abgeleitete Klassen über dieses Attribut.
Anschließend werden die Klassen *Point*, *FixPoint* (die sich wiederum von *Point* ableitet), *Line* und *Circle* definiert. Diese Klassen besitzen als Attribute des Datentyps *double* die jeweils nötigen temporären Koordinaten (siehe Abschnitt 6.1.4).

Definition der Knotentypen

```
1  abstract node class GeometricalElement{
2          Name: string;}
3  node class Point extends GeometricalElement{
4          t_x: double;
5          t_y: double;}
6  node class FixPoint extends Point{}
7  node class Line extends GeometricalElement{
8          t_x1: double;
9          t_y1: double;
10         t_x2: double;
11         t_y2: double;}
12 node class Circle extends GeometricalElement{
13         t_x: double;
14         t_y: double;
15         t_r: double;}
```

Das nächste Codebeispiel zeigt die Definition der Kantentypen bzw. Kantenklassen.

Constraint bildet dabei die abstrakte Basisklasse der Kanten von der sich die weiteren Klassen ableiten, mit deren Hilfe die geometrischen und dimensionalen Zwangsbedingungen repräsentiert werden. Auch hier kann über das Attribut *Name* eine Bezeichnung für die parametrischen Zwangsbedingungen festgelegt werden.
Von *Constraint* leiten sich die beiden wiederum abstrakten Kantenklassen *DimensionalConstraint* und *GeometricalConstraint* ab.

Definition der Kantentypen

```
16 abstract directed edge class Constraint{
17         Name: string;}
18 abstract directed edge class DimensionalConstraint
19 extends Constraint {
20         Value: double;
21         ParametricValue: string;
22         Driven: boolean;}
23 directed edge class DC_OneElement extends DimensionalConstraint;
24
25 directed edge class DC_TwoElement extends DimensionalConstraint{
26         Dock1: string;
27         Dock2: string;}
```

```
28 abstract directed edge class GeometricalConstraint
29 extends Constraint;
30 directed edge class Concentric extends GeometricalConstraint;
31 directed edge class Equal extends GeometricalConstraint;
32 directed edge class Horizontal extends GeometricalConstraint;
33 directed edge class Vertical extends GeometricalConstraint;
34 directed edge class Fixed extends GeometricalConstraint;
35 directed edge class Parallel extends GeometricalConstraint;
36 directed edge class Perpendicular extends GeometricalConstraint;
37 directed edge class Collinear extends GeometricalConstraint;
38
39 directed edge class Coincident extends GeometricalConstraint{
40         Dock1: string;
41         Dock2: string;}
```

DimensionalConstraint besitzt die Attribute *Value*, *ParametricValue* und *Driven*. Die Funktion dieser Attribute ist in Abschnitt 6.2.2 beschrieben.
Es ist dabei zu beachten, dass sowohl *Value* als auch *ParametricValue* die Größe der dimensionalen Zwangsbedingung speichern können. Im Falle eines konkreten numerischen Wertes wird dieser in *Value* als *double* abgelegt. Ist die Zwangsbedingung allerdings von einer anderen Zwangsbedingung abhängig, wird die mathematische Funktion, die diese Abhängigkeit beschreibt, in *ParametricValue* als *string* gespeichert.
Von *DimensionalConstraint* leiten sich die Klassen *DC_OneElement* und *DC_TwoElement* ab, mit denen dimensionale Zwangsbedingungen repräsentiert werden, die entweder die Abmessung eines geometrischen Elements oder den Abstand zwischen zwei Elementen festlegen. *DC_TwoElement* besitzt hier noch die Attribute *Dock1* und *Dock2*, mit denen die jeweiligen Andockstellen (siehe Abschnitt 6.1.3) der durch die Zwangsbedingung verknüpften geometrischen Elemente festgelegt werden.

GeometricalConstraint dient als Basisklasse für die Kantenklassen, die die geometrischen Zwangsbedingungen repräsentieren.
Es leiten sich davon die Klassen *Concentric*, *Equal*, *Horizontal*, *Vertical*, *Fixed*, *Parallel*, *Perpendicular* und *Collinear* ab. Diese benötigen keine weiteren Attribute.
Die abgeleitete Klasse *Coincident* besitzt allerdings noch die Attribute *Dock1* und *Dock2*, mit denen die jeweiligen Andockstellen (siehe Abschnitt 6.1.3) festgelegt werden.

In dieser Form entspricht die Umsetzung des Metamodells in GrGen.NET den Vorgaben, die in Kapitel 6 erarbeitet wurden.
Eine Weiterentwicklung des Metamodells ist dabei jederzeit möglich, muss aber, vor allem bei der Veränderung bereits bestehender Knoten- und Kantentypen, mit der Umsetzung der Graphersetzungsregeln, die im folgenden Abschnitt beschrieben werden, abgestimmt werden.

7.1.2 Graphersetzungsregeln

Um aus dem in *metamodel.gm* definierten Metamodell einen Graphen zu erstellen, müssen Graphersetzungsregeln in einer *rule set file* erstellt werden. Dafür wird die Textdatei *rules.grg* verwendet.
Es wird im Folgenden allerdings nur eine Auswahl der erstellten Regeln besprochen, um den Aufbau der Regeln in GrGen.NET nachvollziehbar darzustellen. Alle weiteren Regeln können in der *rule set file* auf der beiliegenden Compact Disc eingesehen werden.

Als erstes ist in der Datei *rule set file* zu definieren, auf welche *graph model description file* bzw. auf welches Metamodell sich, die Graphersetzungsregeln beziehen sollen.

Dies geschieht wie folgt:

Definition der Knotentypen

```
1 using metamodel;
```

metamodel ist dabei der Dateiname der *graph model description file*.

Anschließend werden der Reihe nach die einzelnen Graphersetzungsregeln definiert. Im Folgenden wird eine Auswahl dieser Regeln aufgeführt und beschrieben. Die hier nicht aufgeführten Regeln sind in der *rule set file rules.grg* auf der beiliegenden Compact Disc einsehbar.

Die Graphersetzungsregeln beziehen sich auf die nummerierten Detaillierungsstufen in Abbildung 7.1.

Im ersten Schritt wird dabei der Mittelpunkt des Tunnelquerschnitts erzeugt. Im Graphen geschieht dies, indem ein neuer Knoten des Typs *FixPoint* erstellt wird. Der Name der Graphersetzungsregel, die diese Operation ausführt lautet *CreateCenter*.

Graphersetzungsregel, die den Knoten zur Repräsentation des Mittelpunkts der Skizze erzeugt

```
1 rule CreateCenter(var x:double, var y:double){
2 modify{
3         c:FixPoint;
4 eval{
5         c.Name = "Center";
6         c.t_x = x;
7         c.t_y = y;
8 }}}
```

In der ersten Zeile werden dabei der Name der Regel und die Parameter (hier x und y), die bei der Ausführung der Regel übergeben werden müssen, definiert. Mit Hilfe der Parameter können die Koordinaten des *FixPoints* festgelegt werden.

Da diese Graphersetzungsregel genutzt wird, um den ersten Knoten in einem noch leeren Graphen zu erzeugen, wird kein Mustergraph definiert.
Durch *modify* wird die Definition des Ersetzungsgraphen eingeleitet. In Zeile 3 wird dabei der neu zu erstellende Knoten anhand seines Typs *FixPoint* deklariert. Außerdem wird diesem Knoten innerhalb der Regel ein Name *c* zugewiesen, über den auf diesen Knoten verwiesen werden kann.
Der Teil der Regel, in dem die Werte der Attribute von Knoten und Kanten bestimmt werden können, wird mit *eval* eingeleitet. Hier wird das Attribut *Name* des Knotens *c* als *Center* definiert. Danach werden die temporären Koordinaten *t_x* und *t_y* des durch den Knoten repräsentierten Fixpunkts festgelegt. Dabei werden die Parameter der Regel genutzt.

Durch die nächste Regel *CreateOuterHull* wird die äußere Hülle des Tunnelquerschnitts erzeugt. Der Parameter *r* der Regel legt dabei der Radius des Tunnels fest.
Hier wird als Mustergraph ein Knoten des Typ *Fixpoint* definiert. Um sicherzustellen, dass es sich um den richtigen Knoten handelt, wird in Zeile 3 geprüft, ob der Name des Knotens tatsächlich *Center* lautet (bzw. ob der Wert des Attributs *Name* mit *Center* übereinstimmt). Nur wenn dies der Fall ist, wird die Regel tatsächlich ausgeführt.
Anschließend wird der Knoten, der den Kreis bzw. die äußere Hülle des Tunnels repräsentiert, erzeugt. Des Weiteren werden die Kanten, die die nötigen parametrischen Zwangsbedingungen repräsentieren, erstellt. Diese setzen den Mittelpunkt des Kreises koinzident mit dem Fixpunkt *Center* und definieren den Radius des Tunnels in Form einer dimensionalen Zwangsbedingung.
Letztendlich müssen den Attributen der neu erstellten Knoten und Kanten noch ihre korrekten Werte zugewiesen werden. Neben dem Namen des Kreises ist es dabei vor allem wichtig zu definieren, dass sich die Zwangsbedingung *Coincident* auf den Mittelpunkt des Kreises bezieht - insofern wird der Wert des Attributs *Dock2* mit *CenterPoint* festgelegt. Die Größe der dimensionalen Zwangsbedingung *DC_OneElement* und die des temporären Radius des Kreises ergibt sich aus dem Parameter *r* der Regel. Die temporären Koordinaten des Kreismittelpunkts werden dabei von den temporären Koordinaten des im Mustergraphen definierten Fixpunkts *Center* übernommen.

Graphersetzungsregel, die die zweite Detaillierungsstufe der Skizze erzeugt

```
1  rule CreateOuterHull(var r:double){
2  x:FixPoint;
3  if{x.Name == "Center";}
4  modify {
5          x-y:Coincident->z:Circle;
6          z-c:DC_OneElement->z;
7  eval{
8          z.Name = "OuterHull";
9          y.Dock2 = "CenterPoint";
10         c.Value = r;
```

```
11          z.t_x = x.t_x;
12          z.t_y = x.t_y;
13          z.t_r = r;
14  }}}
```

Als letztes Beispiel wird die Ersetzungsregel besprochen, die den Graphen erzeugt, der die Skizze in der vierten Stufe des Detaillierungsvorgangs repräsentiert.
Durch diese Regel wird der Boden des Tunnels in Form einer Linie in die Skizze eingefügt. Die Endpunkte dieser Linie müssen dabei koinzident mit Kreisbogen sein, der die innere Hülle des Tunnels darstellt. Weiterhin muss festgelegt werden, dass diese Linie horizontal ist und einen bestimmten Abstand vom Mittelpunkt des Tunnels hat.

Graphersetzungsregel, die den Boden des Tunnels erzeugt

```
1   rule CreateBottomFloor(var offset:double){
2   x:Circle;
3   x1:FixPoint;
4   if{x.Name == "InnerHull" && x1.Name == "Center";}
5   modify {
6           x-y:Coincident->z:Line;
7           x-y1:Coincident->z;
8           x1-y2:DC_TwoElement->z;
9           z-y3:Horizontal->z;
10  eval{
11          z.Name = "Floor";
12          y.Dock1 = "Arc";
13          y.Dock2 = "StartPoint";
14          y1.Dock1 = "Arc";
15          y1.Dock2 = "EndPoint";
16          y2.Value = offset;
17          y2.Name = "Vertical";
18          y2.Dock2 = "Line";
19          z.t_x1 = x1.t_x-pow(pow(x.t_r,2.0-pow(offset,2.0),0.5);
20          z.t_y1 = x1.t_y-offset;
21          z.t_x2 = x1.t_x+pow(pow(x.t_r,2.0)-pow(offset,2.0),0.5);
22          z.t_y2 = x1.t_y-offset;
23  }}}
```

Es ist dabei in erster Linie anzumerken, dass die temporären Koordinaten des Anfangs- und des Endpunktes der Linie hier innerhalb der Regel berechnet werden müssen. Dies geschieht in den Zeilen 19 bis 22 durch entsprechende mathematische Funktionen.
Durch die Attribute *Dock1* und *Dock2* der beiden Kanten des Typs *Coincident* wird festgelegt, dass der Anfangs- und der Endpunkt der Linie jeweils koinzident mit dem Kreisbogen

des Kreises ist. Der weitere Aufbau der Regel enthält keine Funktionen, die noch nicht in den bisher besprochenen Regeln erläutert wurden.

Die Regeln, die die weitere Detaillierung der Skizze durchführen, sind hier nicht mehr abgehandelt, da sie aufgrund ihrer hohen Komplexität sehr umfangreich sind. Es werden in ihnen aber keine Funktionen genutzt, die in diesem Abschnitt nicht erklärt wurden. Wie am Anfang des Abschnitts beschrieben, können sie in der *rule set file*, die dieser Arbeit beiliegt, eingesehen werden.

7.1.3 Ausführung der Graphersetzungsoperationen

Letztendlich müssen die definierten Graphersetzungsregeln angewendet werden. Dies geschieht in der Konsolenanwendung GrShell über die GrGen.NET gesteuert werden kann.

Um die Befehle nicht nacheinander manuell in die Konsolenanwendung eingeben zu müssen, werden diese in der Textdatei *example.grs* eingetragen. Wird diese Datei mit der GrShell aufgerufen, werden die Befehle der Reihe nach ausgeführt.

Im Folgenden ist der Inhalt von *example.grs* dargestellt, der die Regeln ausführt, die den in Abbildung 7.1 dargestellten Detaillierungsvorgang repräsentieren. Dabei wird der Inhalt ab Schritt 4 getrennt für die Varianten a und b aufgeführt.

Anlegen eines neuen Graphen und Ausführen der Graphersetzungsregeln für die Detaillierungsstufen 1 bis 4

```
1  new graph rules "tunnel"
2
3  exec CreateCenter(0.0,0.0)
4  export export1.gxl
5  exec CreateOuterHull(10.0)
6  export export2.gxl
7  exec CreateInnerHull(1.0)
8  export export3.gxl
9  exec CreateBottomFloor(6.0)
10 export export4.gxl
```

In Zeile 1 wird hier der neue Graph *tunnel* erstellt, der auf der *rule set file rules.grg* basiert. Anschließend werden die einzelnen Graphersetzungsregeln ausgeführt und die geforderten Parameter innerhalb der runden Klammern übergeben.
Nach der Ausführung jeder Regel wird der Graph in eine GXL-Datei exportiert.

Um die beiden Varianten a und b darzustellen, müssen jeweils alternative Graphersetzungsregeln angewandt werden:

Detaillierungsstufen 5a

```
11 exec CreateGap(1.0)
12 export export5.gxl
```

Detaillierungsstufen 5b bis 6b

```
11 exec SplitLine("Floor")
12 exec CreateGapL(1.0,"LeftFloor")
13 exec CreateGapR(1.0,"RightFloor")
```

Die so erzeugten Graphen bzw. GXL-Dateien können nun durch das G2I-Tool weiterverarbeitet werden.

7.2 Entwicklung des G2I-Tools

Um den erzeugten Graphen wieder in eine parametrische Skizze umzuwandeln bzw. die Skizze in Autodesk Inventor darzustellen, müssen die durch GrGen.NET exportierten Daten interpretiert werden.
Anschließend werden sie genutzt, um mit Hilfe der API von Inventor Konstruktionsoperationen auszuführen, die die jeweiligen, durch die Knoten und Kanten des Graphen repräsentierten, geometrischen Elemente und parametrischen Zwangsbedingungen erstellen. Die Interpretation des durch GrGen.NET erzeugten GXL-Exports und der Zugriff auf die API erfolgen dabei durch das im Rahmen dieser Arbeit entwickelte Programm G2I (*Graph to Inventor*).
In den folgenden Abschnitten wird zuerst der Funktionsumfang des Tools beschrieben und dann kurz auf die genutzte Programmiersprache und die nötigen Bibliotheken eingegangen. Danach werden die einzelnen Schritte, die das Tool bei seiner Ausführung abarbeitet, beschrieben.

7.2.1 Funktionsumfang

Die wesentliche Funktionalität des Programms ist die Umwandlung des Graphen in eine Skizze in Autodesk Inventor.
Der Input, mit dem das Tool arbeitet, ist dabei eine GXL-Datei, die eingelesen wird. Als Output wird in Autodesk eine neue Skizze erstellt, die die durch den Graphen repräsentierten geometrischen Elemente und parametrischen Zwangsbedingungen enthält.

Konkret handelt es sich bei dem Tool um eine einfache Konsolenanwendung, die in ihrem aktuellen Stadium in erster Linie dazu dient, zu überprüfen, ob aus einem Graphen eine korrekte Skizze erstellt werden kann. Während der Ausführung des Programms sind keine Eingaben durch den Anwender gefordert oder möglich. Informationen, wie beispielsweise der Pfad der einzulesenden GXL- Datei, müssen also direkt in den Quellcode eingefügt werden.

In folgender Grafik ist schematisch dargestellt, welche Schritte der Reihe nach bei der Ausführung des Programms durchgeführt werden.

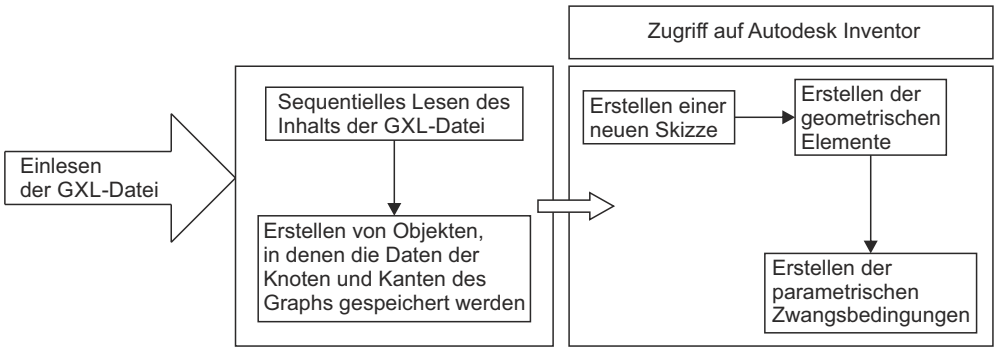

Abbildung 7.2: Ablaufschema des G2I-Tools

Der Quellcode des G2I-Tools ist auf der beiliegenden Compact Disc enthalten (siehe Anhang A).

7.2.2 Programmiersprache und Bibliotheken

Für die Erstellung des G2I-Tool wurde die von Microsoft entwickelte Programmiersprache C# genutzt. Die Umsetzung erfolgte in Visual Studio 2013.

Um auf die Funktionen der API von Autodesk Inventor zuzugreifen, wurde innerhalb des Visual Studio Projekts auf die entsprechende Bibliotheksdatei, die über das Software Developer Kit von Inventor zur Verfügung gestellt wird, verwiesen.

7.2.3 Datenimport und Klassenstruktur

Zum Import der durch GrGen.NET im GXL-Format exportierten Daten, wird die entsprechende Datei durch das G2I-Tool eingelesen. Wie in Abschnitt 3.2.5 dargestellt, sind die Daten des Graphen in Form von Knoten und Kanten der Reihe nach inklusive der zugehörigen Attribute aufgelistet.

Um diese Daten durch das G2I-Tool weiterzuverarbeiten, müssen sie deserialisiert und in Instanzen entsprechender Klassen abgelegt werden. Die dafür nötige hierarchische Klassenstruktur entspricht der des in GrGen.NET definierten Metamodells (siehe Abschnitt 6.3). Die Attribute von Knoten und Kanten werden dabei innerhalb der Klassen durch entsprechende Variablen gespeichert.
Für diesen Vorgang wurde entsprechender Code entwickelt, der die GXL-Datei parst und dabei die dort abgespeicherten Daten in neu erzeugte Objekte einfügt. Die Methode, die dabei den tatsächlichen Einlesevorgang durchführt, ist innerhalb der definierten Klassen definiert.

Dies ist nötig, da je nach Art des Knotens oder der Kante, die eingelesen wird, unterschiedliche Attribute innerhalb des jeweiligen XML-Elements gelesen werden müssen.
Der Quellcode der Klassendefinition des G2I-Tools ist in der Datei *structure.cs* enthalten.

Neben der Deklaration von Variablen gemäß des Metamodells und einem Konstruktor enthalten die Klassen, die sich auf die Daten der Knoten bzw. der geometrischen Elemente beziehen, außerdem eine Variable, die mit *DrawnElement* bezeichnet wird. Diese wird benötigt, um in der jeweiligen Instanz auf ein Objekt in Autodesk Inventor zu verweisen, durch das das zugehörige geometrische Element innerhalb einer Skizze in Inventor repräsentiert wird.
So kann zu einem späteren Zeitpunkt auf jedes geometrische Element in einer Skizze zugegriffen werden. Dies ist nötig, um bei der Erstellung der parametrischen Zwangsbedingungen auf die durch sie verknüpften geometrischen Elemente zu verweisen.
Alle Instanzen der Knoten und Kanten werden je nach Typ in separaten Listen abgelegt.

Nachdem die GXL-Datei erfolgreich eingelesen und damit einhergehend die Instanziierung aller Objekte durchgeführt wurde, können die so verfügbaren Daten für die Erstellung der Skizze in Autodesk Inventor genutzt werden.

7.2.4 Zugriff auf Autodesk Inventor

Um über die API auf Funktionen von Inventor zuzugreifen bzw. Befehle im Programm auszuführen, muss zuerst auf eine laufende Instanz des Programms zugegriffen werden. Falls Inventor nicht ausgeführt wird, wird das Programm durch das G2I-Tool automatisch gestartet. Es handelt sich also wie in Abschnitt 4.2.3 beschrieben, um ein Plug-In, da es, im Gegensatz zu einem Add-In, als eigene Anwendung ausgeführt wird.
Anschließend wird eine neue Bauteil-Datei im Inventor geöffnet und in dieser Datei eine neue 2D-Skizze eingefügt. In dieser Skizze können nun geometrische Elemente und die zugehörigen geometrischen Zwangsbedingungen erstellt werden.

7.2.5 Erstellung der geometrischen Elemente und der dimensionalen Zwangsbedingungen in Autodesk Inventor

Da bei der Erstellung der parametrischen Zwangsbedingungen angegeben werden muss, auf welche geometrischen Elemente sich diese beziehen, müssen die geometrischen Elemente zuerst konstruiert werden.
Dazu werden die Elemente der zuvor erstellten Listen für Punkte, Linien und Kreise der Reihe nach abgearbeitet und mit Hilfe einer entsprechenden Konstruktionsmethode in die Skizze eingefügt. Die Position, an der ein Element eingefügt wird, richtet sich dabei nach den temporären Koordinaten.
Nachdem alle Elemente erstellt wurden, entspricht die Skizze demnach schon ihrem endgültigen

Aussehen. So ist es schon bevor die parametrischen Zwangsbedingungen eingefügt werden möglich, zu überprüfen ob alle geometrischen Elemente korrekt eingelesen und die temporären Koordinaten durch die Graphersetzungsregeln richtig zugewiesen wurden.

Anschließend werden die parametrischen Zwangsbedingungen erstellt. Auch hier werden die Elemente der Listen, in denen die Daten der einzelnen Zwangsbedingungen abgelegt sind, der Reihe nach durch das Programm abgearbeitet.
Je nach Art der Zwangsbedingungen werden dabei unterschiedliche Konstruktionsmethoden der API von Inventor genutzt. Welche Konstruktionsmethode konkret genutzt wird, richtet sich dabei nach den Definitionen in Abschnitt 6.2.2, da die API je nach den geometrischen Elementen, die der Zwangsbedingung zugeordneten sind, verschiedene Konstruktionsmethoden zur Verfügung stellt, von denen die richtige genutzt werden muss.
Soll beispielsweise eine dimensionale Zwangsbedingung erstellt werden, die dem geometrischen Element Kreis zugeordnet ist, wird durch das Programm automatisch entschieden, dass hier innerhalb von Inventor eine Zwangsbedingung vom Typ Radius erforderlich ist.

Nachdem alle Zwangsbedingungen erstellt wurden, ist die Skizze fertig gestellt und die Ausführung des G2I-Tools wird beendet. Nun kann die Skizze in Autodesk Inventor angezeigt und auch bearbeitet oder abgespeichert werden.

Kapitel 8

Fazit und Ausblick

In diesem letzten Kapitel werden zuerst die wesentlichen Ergebnisse der Arbeit zusammengefasst. Anschließend wird ein kurzer Ausblick auf Anwendungs- und Erweiterungsmöglichkeiten des Graphersetzungssystems gegeben.

8.1 Zusammenfassung der Ergebnisse

In dieser Arbeit wurde ein Graphersetzungssystem entwickelt, mit dem es möglich ist, parametrische Skizzen graphbasiert zu repräsentieren. Über die Anwendung von Graphersetzungsregeln ist es zudem möglich, den Graph, durch den eine Skizze repräsentiert wird, zu verändern.
Weiterhin kann durch ein eigens dafür erstelltes Programm die durch den Graphen repräsentierte Skizze in die parametrische CAD-Software Autodesk Inventor eingelesen und dort angezeigt werden.

Als Grundlage für diese Entwicklung eines Graphersetzungssystems wurde Literatur aus den Bereichen der formalen Graphentheorie, der Theorie von Graphersetzungssystemen und der parametrischen Modellierung herangezogen.

Um festzustellen, welche Anforderungen sich an den Graphen, durch den die Skizzen repräsentiert werden sollen, und an das zugehörige Graphersetzungssystem stellen, wurde ein methodisches Vorgehen entwickelt. Dieses ist im Wesentlichen ein iterativer Ablauf von Arbeitsschritten, in denen die Informationen, die in der graphbasierten Repräsentation enthalten sein müssen, immer detaillierter definiert werden.

Auf Basis dieses Vorgehens war es anschließend möglich, eine umfassende Definition des Metamodells und der Graphersetzungregeln, aus denen sich das Graphersetzungssystem zusammensetzt, zu erarbeiten.

Für das in dieser Arbeit untersuchte Beispiel des Detaillierungsvorgangs eines vereinfachten Tunnelquerschnitts konnte dadurch eine Darstellung und automatisierte Detaillierung durchgeführt werden.

Es kann dementsprechend abschließend festgestellt werden, dass der Ansatz, parametrische Skizzen durch Graphersetzung zu detaillieren, prinzipiell umsetzbar ist.

8.2 Ausblick

Aus den Ergebnissen dieser Arbeit ergeben sich im Wesentlichen die im Folgenden aufgeführten Ansatzpunkte für weitere Foschungsarbeiten.

Obwohl funktionsfähig, ist das in dieser Arbeit entwickelte Graphersetzungsystem in seinen Anwendungsmöglichkeiten eingeschränkt. Es erscheint daher sinnvoll, weitere Graphersetzungsregeln zu erstellen und diese hinsichtlich ihrer Anwendbarkeit und Wiederverwendbarkeit zu überprüfen.
Zusätzlich können weitere Knotentypen in das Metamodell integriert werden, um auch Skizzen repräsentierbar zu machen, die nicht nur aus Punkten, Linien und Kreisen bestehen, sondern auch andere geometrische Elemente beinhalten.

Auch das entwickelte G2I-Tool kann weiter optimiert werden.
Dabei besteht ein möglicher Ansatz darin, neben der API von Autodesk Inventor auch auf die von GrGen.NET zuzugreifen. So könnte einerseits der Importvorgang vereinfacht oder sogar ganz umgangen werden, indem direkt auf die Datenstruktur des Graphen innerhalb von GrGen.NET zugegriffen wird.
Außerdem ist es denkbar, die Funktionen von GrGen.NET über eine intuitive graphische Benutzeroberfläche steuerbar zu machen und so auch Nutzern, die keine Kenntnisse von Graphersetzungssystemen haben, eine Ausführung der Graphersetzungsoperationen zu ermöglichen.

Weiterhin kann untersucht werden, inwiefern das hier erarbeitete Konzept erweitert oder abgeändert werden muss, um praktisch angewendet zu werden.
Einerseits besteht die bereits in der Einleitung beschriebene Möglichkeit, die graphbasierte Repräsentation zur Erstellung von mehrskaligen geometrischen Modellen zu nutzen, bei denen sich parametrische Beziehungen nicht nur auf Elemente innerhalb eines LOD beziehen, sondern auch LOD-übergreifend definiert sind.
Andererseits kann weiter untersucht werden, wie zeitaufwendige manuelle Zeichnungsoperationen, mit denen parametrische Skizzen erstellt werden, auf Basis der Ergebnisse dieser Arbeit in der Praxis erleichtert oder automatisiert werden können. Dazu ist es denkbar, dass

vom Nutzer in der Skizze einzelne Elemente ausgewählt werden, die dann mit Hilfe einer Graphersetzungsregel veränderbar sind.

Anhang A

Compact Disc

Auf der beigefügten CD befindet sich folgender Inhalt:

- Die vorliegende Thesis [PDF]
- Die Dateien des Graphersetzungssystems [GM, GRG, GRS]
- Der Quellcode des G2I-Tools [CS]

Literaturverzeichnis

Blomer, J., Geiss, R. & Jakumeit, E. (2013). The GrGen User Manual.

Borrmann, A., Flurl, M., Jubierre, J. R., Mundani, R.-P. & Rank, E. (2014). Synchronous collaborative tunnel design based on consistency-preserving multi-scale models. *Advanced Engineering Informatics*.

Borrmann, A., Ji, Y. & Jubierre, J. R. (2012). Multi-scale geometry in civil engineering models: Consistency preservation through procedural representations. In: *Proc. of the 14th Int. Conf. on Computing in Civil and Building Engineering*, Moscow and Russia.

Borrmann, A., Kolbe, T., Donaubauer, A., Steuer, H., Jubierre, J. & Flurl, M. (2014). Multi-Scale Geometric-Semantic Modeling of Shield Tunnels for GIS and BIM Applications. *Computer-Aided Civil and Infrastructure Engineering*.

Diestel, R. (1996). *Graphentheorie*. Springer-Lehrbuch. Berlin and New York: Springer.

Fudos, I. (1995). *Constraint solving for computer aided design*. Dissertation, Purdue University.

Fudos, I. & Hoffmann, C. M. (1997). A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics (TOG)* 16(2), S. 179–216.

Geiß, R., Batz, G. V., Grund, D., Hack, S. & Szalkowski, A. (2006). GrGen: A fast SPO-based graph rewriting tool. In: *Graph Transformations*, S. 383–397. Springer.

Heckel, R. (2006). Graph Transformation in a Nutshell. *Electronic Notes in Theoretical Computer Science* 148(1), S. 187–198.

Helms, B. (2013). *Object-Oriented Graph Grammars for Computational Design Synthesis*. Dissertation, Technische Universität München, München.

Hoisl, F. (2012). *Visual, Interactive 3D Spatial Grammars in CAD for Computational Design Synthesis*. Dissertation, Technische Universität München, München.

Ji, Y., Borrmann, A., Beetz, J. & Obergriesser, M. (2013). Exchange of Parametric Bridge Models using a Neutral Data Format. *ASCE Journal of Computing in Civil Engineering* 27(6), S. 593–606.

Jubierre, J. R. (2009). Analysis and coupling of a Geometric Constraint Solver with a CAD application. Masterarbeit, Technische Universität München.

Kniemeyer, O. (2008). *Design and Implementation of a Graph Grammar Based Language for Functional-Structural Plant Modelling.* Dissertation, Brandenburgische Technischen Universität Cottbus.

Kolbe, T. H. (2009). Representing and exchanging 3D city models with CityGML. In: *3D geo-information sciences*, S. 15–31. Springer.

Kroll, M. (2007). GrGen.NET: Portierung und Erweiterung des Graphersetzungssystems GrGen. *Studienarbeit, IPD Goos, Universität Karlsruhe*.

Rozenberg, G. (1997). *Handbook of graph grammars and computing by graph transformation.* Singapore: World Scientific.

Sacks, R., Eastman, C. M. & Lee, G. (2004). Parametric 3D modeling in building construction with examples from precast concrete. *Automation in Construction* 13(3), S. 291–312.

Shah, J. J. (1995). *Parametric and feature-based CAD/CAM: concepts, techniques, and applications.* John Wiley & Sons.

Tittmann, P. (2003). *Graphentheorie.* Mathematik-Studienhilfen. München and Wien: Fachbuchverl. Leipzig im Carl-Hanser-Verl.

Winter, A., Kullbach, B. & Riediger, V. (2002). An overview of the GXL graph exchange language. In: *Software Visualization*, S. 324–336. Springer.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Master-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 28. September 2014

Simon Vilgertshofer

Simon Vilgertshofer
Zielstattstraße 58
D-81379 München
e-Mail: vilgertshofer@tum.de