Technische Universität München

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Computergestützte Modellierung und Simulation

3D-Modellierung von Brückenbauwerken mit Autodesk Revit und Erweiterung um IFC4x1 Entitäten

Christoph Raidl

Bachelorthesis

für den Bachelor of Science Studiengang Bauingenieurwesen

Autor: Christoph Raidl

Matrikelnummer: [redacted]

1. Betreuer: Prof. Dr.-Ing. André Borrmann

2. Betreuer: Štefan Markič, M.Sc.

Ausgabedatum: 15. Januar 2018

Abgabedatum: 15. Juni 2018

Inhaltsverzeichnis

Zusammenfassung

Das IFC-Dateiformat ist zu einem Standard in der Welt der Modellierung geworden. Um neue Einsatzmöglichkeiten zu schaffen und neue Projekte mit diesem Dateiformat auszutauschen, wird es von der buildingSMART e.V. stetig weiterentwickelt. Die letzte veröffentlichte Entwicklung, das IFC4x1 enthält neue Entitäten, die es ermöglichen sollen Brückenbauwerke darzustellen.

In dieser Arbeit habe ich geprüft, ob die neu hinzugefügten Entitäten der IFC4x1 Version auch bei modellierten Brückenprojekten ausreichend sind, um alle wichtigen Eigenschaften und Informationen zu vermitteln oder ob weitere Aufwendungen notwendig sind.

Ich modellierte die Pfeiler der Donnersbergerbrücke und exportierte sie als einzelne IFC2x3-Dateien. Die Achse der Brücke modellierte ich in ProVI und exportierte sie als IFC4x1-Datei. Deck und Kappen der Brücken habe ich noch mit den richtigen Parametern versehen und die Geometrie exportiert. All diese Dateien habe ich durch ein selbstentwickeltes Python-Programm zusammengefasst und um zusätzliche IFC4x1 Entitäten erweitert.

Abstract

The IFC file format has become a standard in the world of modeling. In order to create new application possibilities and exchange new projects with this data format, buildingSMART e.V. is constantly developing it further. The latest published development, the IFC4x1, contains new entities that should make it possible to represent bridge structures.

In this thesis I checked if the newly added entities of the IFC4x1 version are also sufficient for modeled bridge construction projects to convey all important properties and information or if further expenditures are necessary.

I modeled the pillars of Donnersbergerbrücke and exported them as IFC2x3 files. I modeled the alignment of the bridge in ProVI and exported it as IFC4x1 file. I added the right parameters to the decks and caps of the bridges and exported the geometry. All these files have been combined by a self-developed Python program and extended by additional IFC4x1 entities.

Danksagung

An dieser Stelle will ich mich bei allen Personen bedanken, die mir bei der Bewältigung dieser Arbeit geholfen und mich unterstützt haben. Besonders danke ich Štefan Markič, der mir als Betreuer meiner Arbeit immer mit Hilfe und Rat zur Seite stand.

1 Einführung

Im Studiengang Bauingenieurswesen an der Technischen Universität München wurden wir schon ab dem ersten Semester in den Bereich der Bauinformatik eingearbeitet. Angefangen mit dem Lernen der Grundlagen für die mathematische Programmiersprache für das Programm Matlab und dem Bedienen von CAD-Programmen, wurden im weiteren Verlauf des Studiums die Vorteile und Anwendungsbereiche von Building Information Modeling (BIM) gelehrt.

BIM ist wie folgt beschrieben:

> „Building Information Modeling (BIM) ist ein intelligenter, auf einem 3D-Modell basierender Prozess, der Architekten, Ingenieuren und Bauunternehmern Informationen und Werkzeuge für effiziente Planung, Entwurf, Konstruktion und Verwaltung von Gebäuden und Infrastruktur bereitstellt.“ [1]

Da diese Bereitstellung an Gebäudeinformationen als zukünftiger Standard in der Baubranche gilt, hat sich eine neue Schnittstelle zur fehlerfreien Übertragung des Modells zwischen verschiedenen Programme etabliert. Das Format dieser Schnittstelle nennt sich Industry Foundation Classes (IFC) und wurde von buildingSMART international kontinuierlich entwickelt.

Die IFC sind wie folgt definiert:

> Das buildingSMART Datenmodell, auch bekannt unter der Bezeichnung Industry Foundation Classes (IFC), stellt ein allgemeines Datenschema dar, das einen Austausch von Daten zwischen verschiedenen proprietären Software-Anwendungen ermöglicht. Dieses Datenschema umfasst Informationen aller am Bauprojekt mitwirkender Disziplinen über dessen gesamten Lebenszyklus. [2]

Von buildingSMART e.V. wurde im Sommer 2017 eine neue Version des Datenmodells veröffentlicht, das die Darstellung von Trassenmodellen wie Brücken ermöglichen soll.

Dadurch wurden durch das „IfcAlignment project“ Entitäten eingebracht, die folgende Möglichkeiten bieten [3]:

- Trassierungsinformationen über Bauphasen hinweg auszutauschen
- Trassierungsinformationen mit anderen Projektinformationen wie Querschnitten und vollständiger 3D-Geometrie von Bauelementen zu verknüpfen
- Trassierungsinformationen abzufragen, die Daten wie z.B. lineare Referenzierung für die Positionierung bereitstellen
- offener Datenzugriff auf Trassierungsinformationen aus Bestandsverwaltungsdatenbanken
- Darstellung von IFC-Ausrichtungsmodellen auf InfraGML und LandXML“

Der jüngste erfolgreiche Abschluss von den Projekten IFC-Alignment und der IFC-Infra Overall Architecture, deren Ergebnisse zur Veröffentlichung von IFC4x1 führten und gute Grundlagen für die Beschreibung von linearen Ingenieurbauwerken lieferten. [4]

Das IFC-Alignment und das IFC-Overall Architecture Projekt haben ebenfalls Datenstrukturen und Implementierungsrichtlinien entwickelt, einschließlich [5]:

- Ausrichtung und Positionierung (in Zusammenarbeit mit dem IFC-Alignment Projekt);
- Geometrische Darstellungen:
    - Zeichenkettendarstellung
    - Querschnittsdarstellung
    - Oberflächendarstellung
    - Festkörperdarstellung
    - Gelände

Die Möglichkeit des IFC-Datenmodells modellierte Elemente als Textform mit allen wichtigen Informationen programmübergreifend zu teilen, ist, für die Art und Weise zu bauen, wegweisend. Durch diese Arbeit will ich der Entwicklung des IFC-Datenmodells helfen, indem ich eine Machbarkeitsstudie anhand eines Brückenmodells durchführe.

Hierbei werde ich klären, ob es möglich ist dieses Modell im IFC4x1-Datenformat richtig anzeigen zu lassen.

Um mein Thema bearbeiten zu können musste ich mich als Erstes dafür entscheiden, welche Brücke ich nachmodellieren möchte. Ich entschied mich für die Donnersbergerbrücke, da sie zum einen ein Herzstück der Stadt München und zum anderen Teil des Mittleren Rings ist, der den Hauptverkehrsweg in München darstellt.

Bei der Bearbeitung meines Projektes zur Generierung einer IFC4x1-Datei, gab es zwei Möglichkeiten. Zum einen alle Elemente bis auf die Trasse in Autodesk Revit zu modellieren und anschließend die große Datei zu bearbeiten, zum anderen die Elemente einzeln zu modellieren und zu exportieren und anschließend durch ein selbstentwickeltes Programm die einzelnen Dateien zusammenzuführen und so umzuschreiben, dass es dem IFC4x1-Schema entspricht. Mit kleinen IFC-Dateien kann man leichter arbeiten und Fehler können leichter gefunden werden, somit fiel meine Entscheidung über die Bearbeitung des Projektes auf diese Möglichkeit.

2 Grundlagen

2.1 Trasse

Eine Trasse ist eine „geplante, im Gelände abgesteckte Linienführung eines Verkehrsweges, einer Versorgungsleitung o. Ä.“ [6].

Die Linienführung besteht aus drei Bestandteilen. Die Achse, die Gradiente und der Querschnitt. Die Achse ist die Draufsicht auf die Linienführung und die Gradiente der Längsschnitt (Höhenprofil). Der Querschnitt zeigt die Neigung des Verkehrsweges an [7].

2.1.1 Achse

Die Achse ist aus drei möglichen Trassierungselementen zusammengesetzt: Gerade, Kreisbogen und Klothoide. Diese Trassierungselemente werden aufeinander abgestimmt und können nicht für sich alleine betrachtet werden. Die Abstimmung lässt sich durch Aneinanderreihen und gegenseitige Zuordnung der Trassierungselemente erreichen [7].

In Abbildung 1 ist aufgezeigt, wie eine beispielhafte Achse aus Radien und Klothoiden aussieht. Dieses Bild zeigt nur Anfang und Ende der kompletten Achse, der Mittelteil fehlt in der Anschauung.

Die Pfeile markieren einen bestimmten Punkt auf der Achse und sind mit Kilometrierungswerten oder Parametern beschriftet. Kilometrierungen beginnen immer am Anfang der Achse und enden am Ende und werden durch einen Beschriftungspfeil mit der genauen Kilometrierung markiert. Dazwischen werden die Kilometrierungswerte alle 100 Meter durch einen Strich und alle 500 Meter durch einen Beschriftungspfeil angezeichnet. Hier ist zu sehen, dass die Achse mit einer Linkskurve (Radius R = 300m) beginnt und dann durch die Klothoide (A=150) zu einer Rechtskurve (R = 300m) übergeht. Der Unendlichkeitspunkt zeigt hierbei den Punkt an, an dem die Linkskurve in die Rechtskurve übergeht.

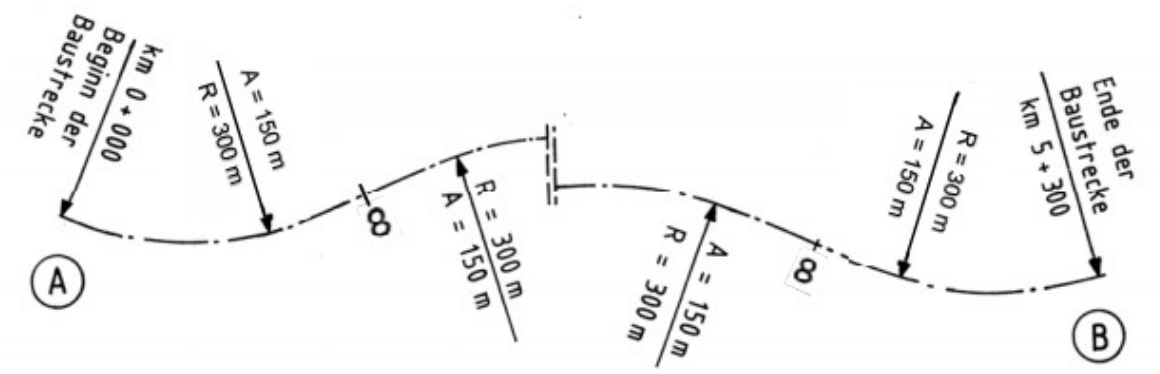


Abbildung 1: Achse aus Radien und Klothoiden mit Kilometrierung [7]

Die Achse ist für die richtige horizontale Positionierung wichtig. Dazu zählt die Stationierung an der Achse über die Kilometrierung und der rechtwinklige horizontale Abstand zu der Achse an dieser Stationierung. Die Abbildung 2 zeigt die obige Achse mit einem Fundament (orange). Diese Skizze veranschaulicht die achsbasierte Positionierung von Elementen. Das Fundament wird hierbei auf der kürzesten Strecke zur Achse verbunden, sodass der horizontale Abstandspfeil (blau) rechtwinklig auf der Achse steht. Der Wert der Stationierung bestimmt sich aus der Strecke (rot) vom Anfang der Achse bis hin zum Abstandspfeil. [8]

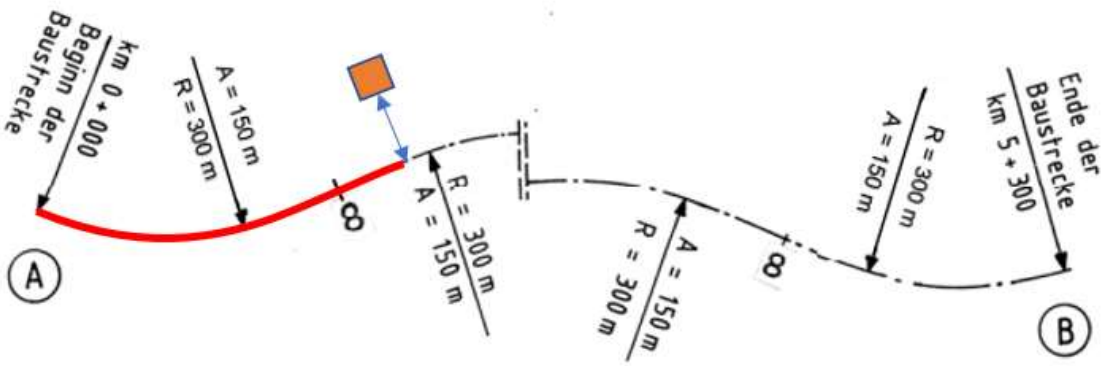


Abbildung 2:Skizze: Positionierung achsbasierte Objekte [7]

2.1.2 Gradiente

Die Gradiente stellt die Höhenabwicklung der Linienführung dar. In der Abbildung 3 ist eine Beispielgradiente abgebildet. Die Gradiente soll hierbei möglichst nahe am Gelände verlaufen, um möglichst wirtschaftlich zu bauen. Die Stellen, an denen die Gradiente über der Erde verläuft, sind hellgrau und die Bereiche, an denen sie unterhalb der Oberfläche verläuft, sind schwarz dargestellt. Auch soll die Gradiente eine gute Ausrundung an Hoch- und Tiefpunkten haben, um ein sicheres Fahren auf dieser Straße zu gewährleisten. [8]

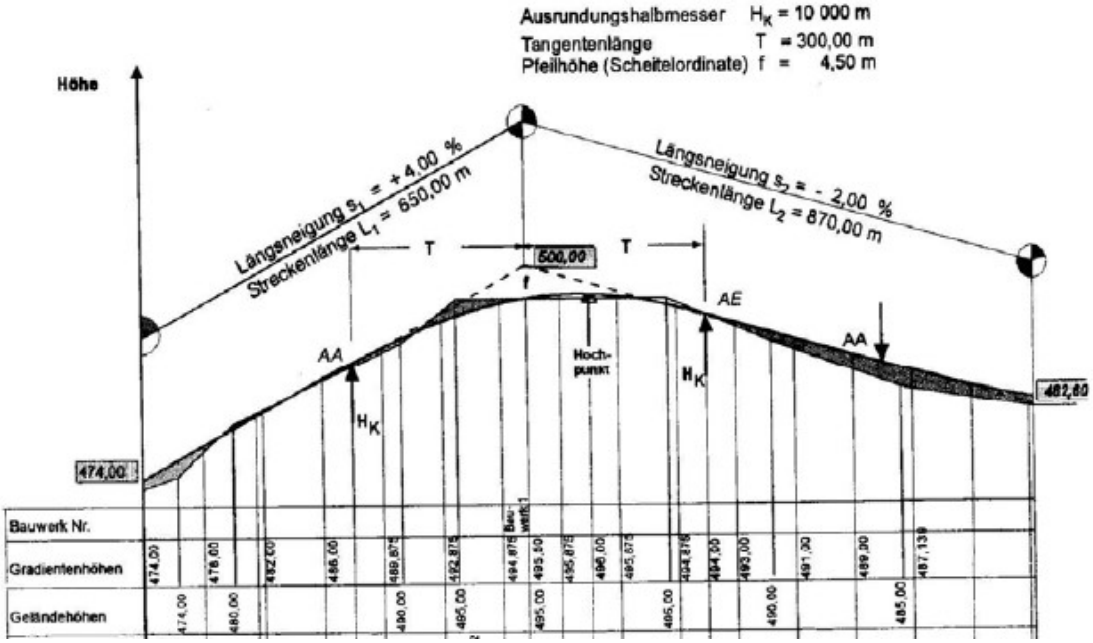


Abbildung 3: Gradiente mit Beschriftung [7]

2.2 Brücken

> „Brücken sind Überführungen eines Verkehrsweges über einen anderen Verkehrsweg, über ein Gewässer oder tiefer liegendes Gelände, wenn ihre lichte Weite rechtwinklig zwischen den Widerlagern gemessen 2,00 m oder mehr beträgt.“

Definition nach DIN 1076 aus Verkehrsblatt-Dokument Nr. B 5276 Vers. 07/97

In meiner Arbeit verwende ich eine Brücke als Modell, um damit meine Überprüfungen des IFC-Datenmodells zu beweisen. Die gewählte Brücke (siehe Abbildung 4) ist eine der Hauptverkehrsbrücken in München. Die Donnersbergerbrücke.

> „Bereits 1900 wurde die erste Brücke gebaut, welche die Donnersbergerstraße mit der südlich gelegenen Trappentreustraße verband. In den 1930er Jahren wurde sie erstmals erneuert und wesentlich breiter ausgeführt. In den 1960er Jahren begann die Planung und der Bau des Mittleren Ringes, einer mehrspurigen Schnellstraße rund um München, ganz im Zeichen der "autogerechten Stadt". Zwischen 1969 und 1971 erfolgte der Umbau der Brücke und die gut dreifache Verbreiterung der ursprünglichen Brücke. Außerdem wurde die Fahrbahn verschwenkt von der Donnersbergerstraße in die Landshuter Allee, da sich diese aufgrund ihrer Breite und ihres Verlaufs besser für den Mittleren Ring eignete.“ [9]

Abbildung 4: Blick über die Donnersbergerbrücke [10]

2.3 IFC4x1

Das IfcAlignment project ist die erste Stufe der Entwicklung von IFC5. IfcRail, IfcRoad und IfcBridge bauen auf die Grundlage von den Projekten IFC Alignment 1.0 / 1.1 und IFC Overall Architecture auf (siehe Abbildung 5).

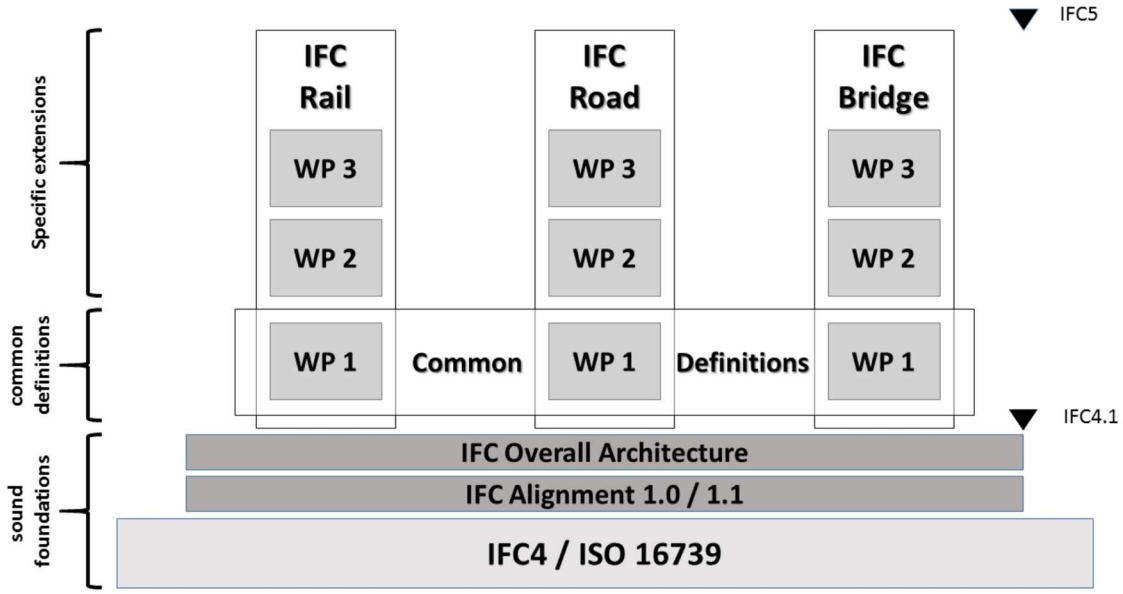


Abbildung 5: geplanter Weg zu IFC5 [11]

Diese Projekte sind bereits abgeschlossen und die Entitäten wurden bereits in das Datenmodell unter der Version IFC4x1 eingebracht. [5] & [12]

Die neuen Entitäten werde ich im Folgenden näher beleuchten, da sie essentiell für meine Arbeit sind.

2.3.1 IfcAlignment

IfcAlignment ist die Entität im neuen Dateiformat, die meine Arbeit erst möglich macht.

Diese Entität wurde entwickelt und eingebunden, um Straßen- und Brückenprojekte mit IFC darstellen zu können. Es soll nun möglich sein, achsbasierte Objekte an eine Trasse anzuhängen. Somit ist der Körper nicht mehr im Raum mit Hilfe von Koordinaten in IfcLocalPlacement platziert, sondern in einer gewissen Position zur Achse. Diese Position bestimmt sich über mehrere Entitäten, die zum einen Verschiebungen zur Achse und zum anderen Verdrehungen des Körpers berücksichtigen.

Die IfcAlignmentCurve stellt hierbei die Entität dar, die die Achse der Brücke in der IFC-Datei abbildet. [13]

2.3.2 IfcLinearPlacement

Diese Achse wird mit jedem einzelnen Objekt mit Hilfe der Entität IfcLinearPlacement verknüpft. Durch diese Verknüpfung steht das Objekt nicht einfach in einem Raum verknüpft mit Koordinaten, sondern ist an eine Achse angehängt. Verändert man die Achse, so verändert sich gleichzeitig die Position des Objektes im Raum, aber nicht der Abstand zur Achse.

Die Verschiebungen relativ zur Achse werden von der Entität IfcDistanceExpression festgelegt. IfcDistanceExpression gibt dem achsbezogenen Element Angaben über seine Position an der Achse (Kilometrierungspunkt an der Achse), seinen lateralen Abstand (horizontale Verschiebung) und seine vertikale Verschiebung zur Achse. Durch diese drei Parameter ist die Position des Körpers nun völlig bestimmbar.

Die Ausrichtung dieses Körpers zur Achse (Verdrehung) an dieser Position ist über die Entität IfcOrientationExpression festgehalten. In IfcOrientationExpression sind IfcDirection-Entitäten verknüpft. Diese verknüpften Entitäten beinhalten einen Vektor zur Ausrichtung des Körpers. All diese Verknüpfungsverbindungen sind in Abbildung 6 aufgezeigt. [13]

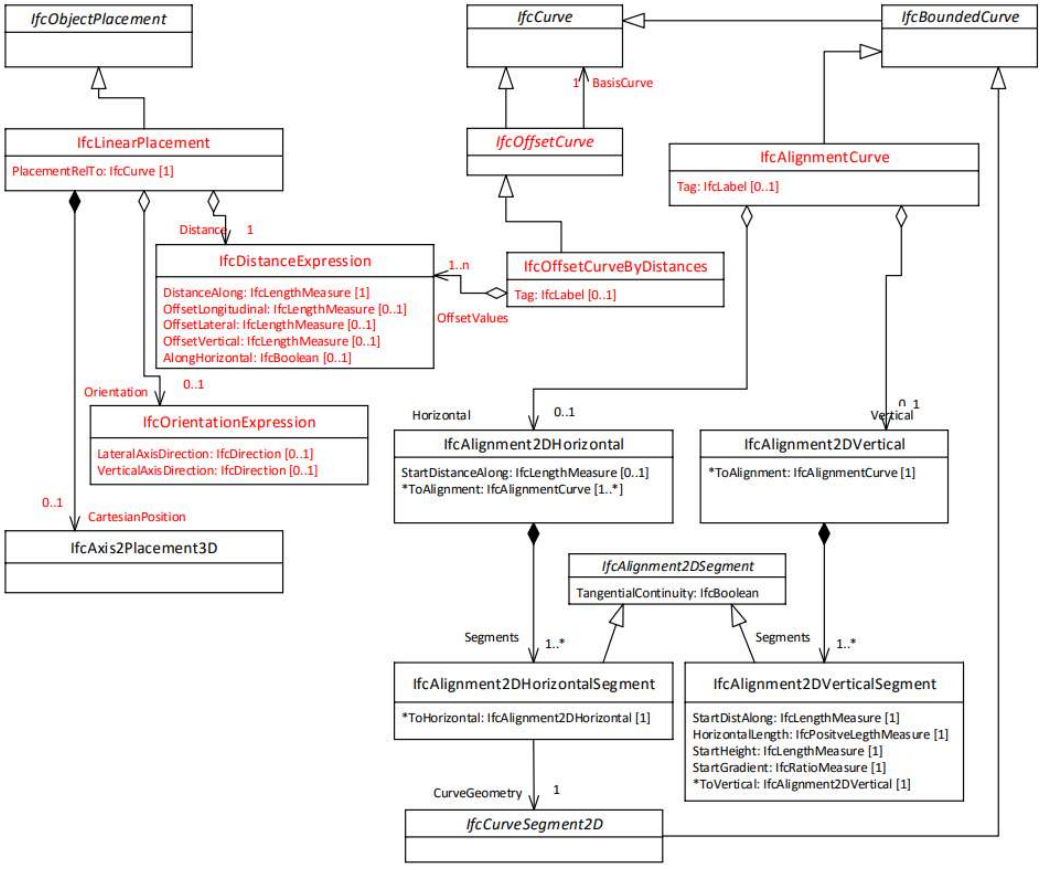


Abbildung 6: Verknüpfung von LinearPlacement mit IfcAlignmentCurve [13]

2.3.3 IfcSectionedSolidHorizontal

Die Möglichkeit Profildarstellungen zu definieren bestand auch schon vor IFC4x1 in Form der Boundary Representation (Brep). Da IFC5 aber darauf abzielt Projekte abbilden zu können, die eine Trasse beinhalten, war Brep nicht mehr flexibel genug für diesen Anwendungsbereich. Brep definiert einen festen Volumenkörper, der sich aber bei Änderung der Achse nicht mitverändern würde. Diese Eigenschaft ist besonders bei Brückenprojekten notwendig. IfcSectionedSolidHorizontal nimmt ein Profil und zieht es entlang einer Achse zum nächsten Profil. Dies verschafft uns die Möglichkeit durch Ändern der Achse auch die Darstellung des Decks zu verändern.

IfcSectionedSolidHorizontal ermöglicht es eine geschlossene Polylinie entlang einer Achse zu ziehen. In Abbildung 7 wird die Funktionsweise von dieser Entität näher dargestellt. Polylinien sind als IfcPolyline in meiner IFC-Datei festgehalten. Polylinien sind eine Aneinanderreihung von mehreren Linien und Linien sind nichts anderes als eine Strecke zwischen zwei Punkten/Koordinaten (IfcCartesianPoints).

Die Entität IfcArbitraryClosedProfileDef verwandelt diese Polylinien in ein Profil und vergibt ihm noch einen Namen. Damit diese Profile an den richtigen Stellen eingesetzt werden, ist IfcSectionedSolidHorizontal mit IfcDistanceExpression Entitäten verbunden. Jedes Profil, das in IfcSectionedSolidHorizontal verknüpft ist, bekommt dadurch noch eine Zuweisung über die Position an der Achse und laterale (horizontale) und vertikale Verschiebung zur Achse.

IfcSectionedSolidHorizontal verbindet alle verknüpften IfcArbitraryClosedProfileDef und bildet daraus einen Festkörper. IfcSectionedSolidHorizontal enthält nur die reine Geometrie des Elementes.

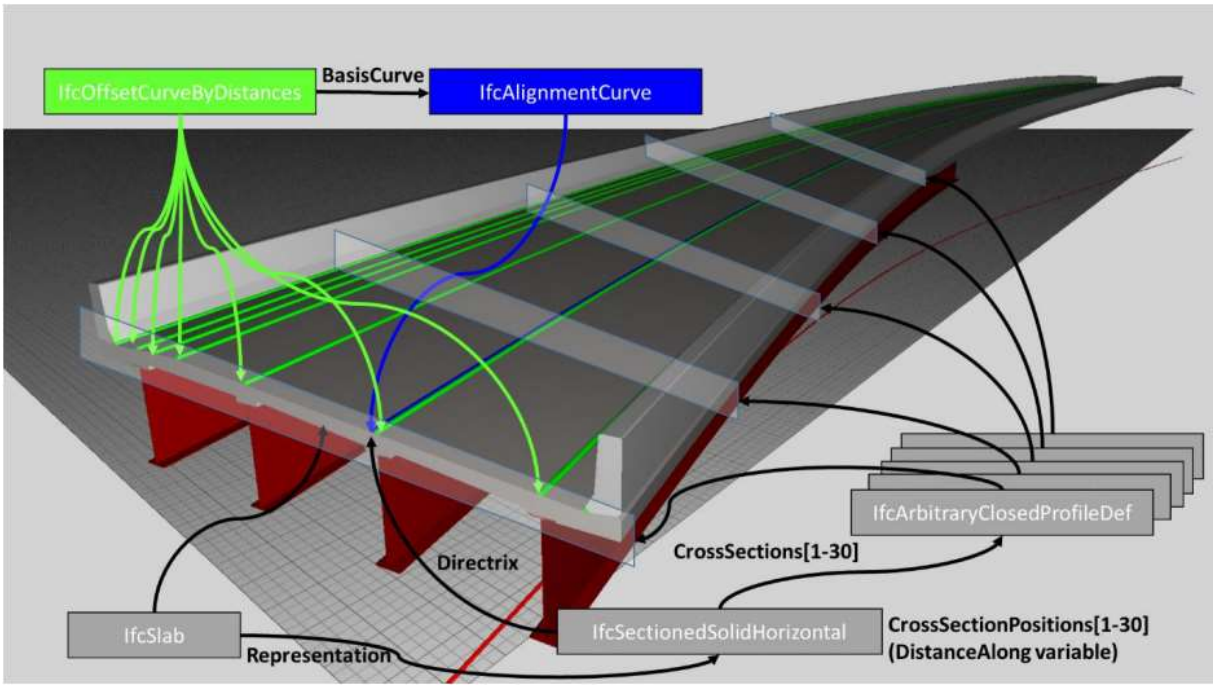


Abbildung 7: IfcSectionedSolidHorizontal und IfcArbitraryColsedProfileDef [5]

Die Abbildung zeigt die verschiedenen Profile (IfcArbitraryClosedProfileDef) an einer Achse (IfcAlignmentCurve) und die Verbindung der Objekte (IfcSlab) durch Extrusion des Profils entlang der Achse mit IfcSectionedSolidHorizontal. IfcOffsetCurveBy-Distance ist eine Entität, die es erlaubt parallele Achsen zur Hauptachse zu erzeugen (zum Beispiel wie im Bild für Stahlträger). [13]

Abbildung 8 zeigt die Beziehungen der Entitäten rund um IfcSectionedSolidHorizontal auf.

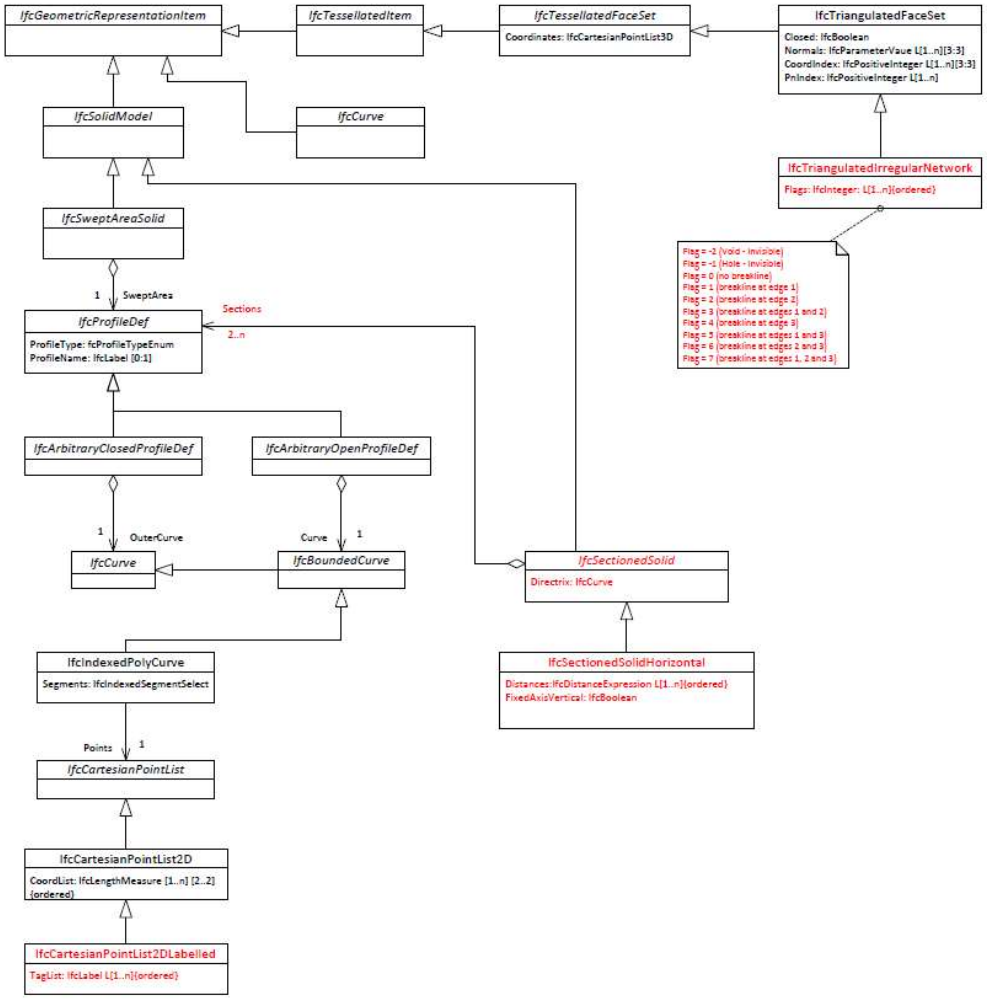


Abbildung 8: Klassendiagramm IfcSectionedSolidHorizontal [12]

2.3.4 IfcRelContainedInSpatialStructure

IfcRelContainedInSpatialStructure, wird verwendet, um Elemente einer bestimmten Ebene der räumlichen Projektstruktur zuzuordnen. Jedes Element kann nur einmal einer bestimmten Ebene der Raumstruktur zugeordnet werden. Damit alle Elemente, die ich im Laufe der Modellierung erstelle, am Schluss in der IFC-Datei auch erkannt werden, muss die Entität IfcRelContainedInSpatialStruture diese Elemente beinhalten.

In Abbildung 9 ist veranschaulicht dargelegt, welche Entitäten durch IfcRelContainedInSpatialStructure verbunden werden. Hierbei ist zu sehen, dass die Treppe, im Gegensatz zu den Wänden, keiner bestimmten Ebene angehört, weil sie zwei Ebenen miteinander verbindet. [14]

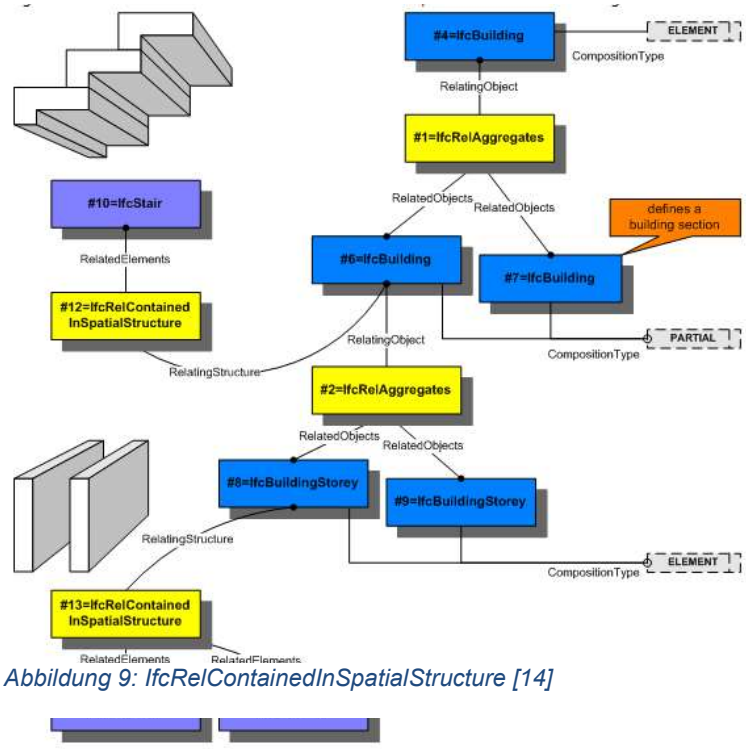


Abbildung 9: IfcRelContainedInSpatialStructure [14]

2.4 Programmierung

Datenverarbeitung: Alle behandelten IFC-Dateien meines Modells wurden einzeln eingelesen und verarbeitet. Diese Daten wurden durch Bearbeiten der Strings verändert. Ich habe mich bewusst für diese Wahl entschieden, da ich ein Programm entwickeln wollte, das möglichst einfach meine Dateien bearbeitet. Die Verwendung von Klassen oder der Möglichkeit Modelle einzulesen habe ich ausgelassen.

2.4.1 Regular Expression

„Regular Expressions (genannt REs, oder regexes, oder regex patterns) sind im Wesentlichen eine kleine, hochspezialisierte Programmiersprache, die in Python eingebettet und über das „re-Modul“ verfügbar gemacht wird. Mit dieser kleinen Sprache geben Sie die Regeln für den Datensatz an, den Sie anpassen möchten; dieser Datensatz kann Sätze, E-Mail-Adressen, TeX-Befehle oder irgendetwas anderes enthalten. Sie können dann Fragen stellen wie "Stimmt diese Zeichenkette mit dem Muster überein", oder "Gibt es eine Übereinstimmung mit dem Muster irgendwo in dieser Zeichenkette?“ Res können auch verwendet werden, um einen String zu modifizieren oder ihn auf verschiedene Arten aufzuteilen.“

Aus dem Englischen übersetzt: [15]

In meinem Code habe ich die Regular Expressions verwendet, um:

- Die Nummerierung der Zeilen zu finden und auszutauschen
- Alle relevanten Einträge aus den IFC-Dateien zu erkennen
- Platzhalter zu finden und mit Werten zu füllen
- Objekte zu suchen und IfcRelContainedInSpatialStructure zu befüllen

Anhand der ersten Verwendung (Die Nummerierung zu finden und auszutauschen) will ich so einen Prozess kurz erklären. Abbildung 10 zeigt den Vorgang und die dafür verwendeten Muster. Zuerst werden alle „#“ gesucht die 1-x Zahlen von 0-9 dahinter haben ('#[0-9]+'). Danach werden aus diesen Ergebnissen alle 1-x Zahlen von 0-9 herausgefiltert ('[0-9]+'). Jetzt wird die Berechnung durchgeführt und alle Zahlen werden vergrößert. Anschließend werden die alten Zahlen durch die neuen ausgetauscht. Das Muster ist an dieser Stelle durch %s variabel gehalten. Dort wird immer die aktuelle Zahl eingesetzt, die ersetzt werden soll ('%s\b' %Zahl).

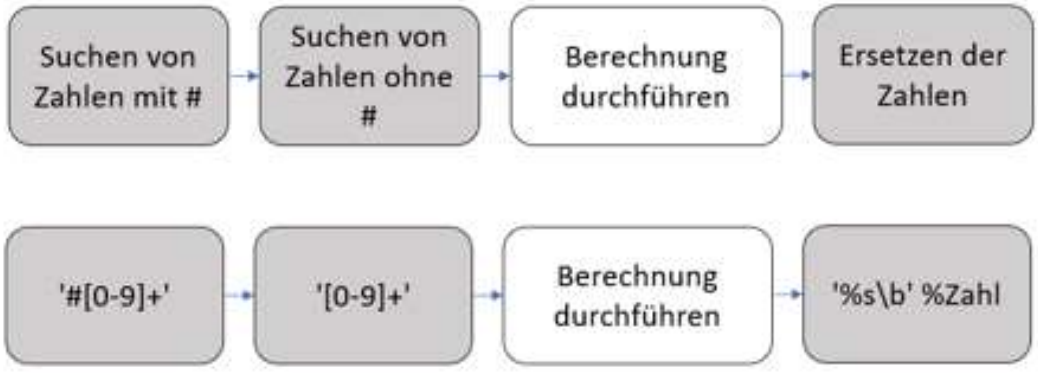


Abbildung 10: Prozess Nummerierungsänderung mit Regular Expression

3 Methodik

3.1 Prozess

Momentan gibt es keine Software auf dem Markt, mit der Brücken geplant werden können und das Modell anschließend als IFC4x1 exportiert werden kann. Deswegen musste ich für meine Arbeit anders vorgehen, um eine IFC4x1 Brückendatei zu erstellen.

Damit der Fokus der Arbeit auf den IFC4x1 Entitäten bleibt, habe ich nur Achse, Pfeiler, Decks und Kappen als wesentlichen Teile einer Brücke modelliert.

Da die Achse der Brücke noch nicht vorhanden war, habe ich diese mit ProVI nachmodelliert, damit die Pfeiler, Deck und Kappen richtig im Projekt positioniert werden können.

Mit Hilfe von Autodesk Revit habe ich die Stützen und Fundamente der Pfeiler nachmodelliert und anschließend wurden die konstruierten Modelle einzeln als IFC2x3-Dateien exportiert.

Die Decks und Kappen habe ich in Inventor nach den Bemaßungen in den Plänen richtig parametrisiert und die Geometrie exportiert. Diese Dateien wurden durch Hinzufügen von drei weiteren Entitäten zu IFC4x1-Dateien.

Nun gehe ich näher auf die oben genannten Schritte ein:

3.2 Modellierung von Achse und Gradiente

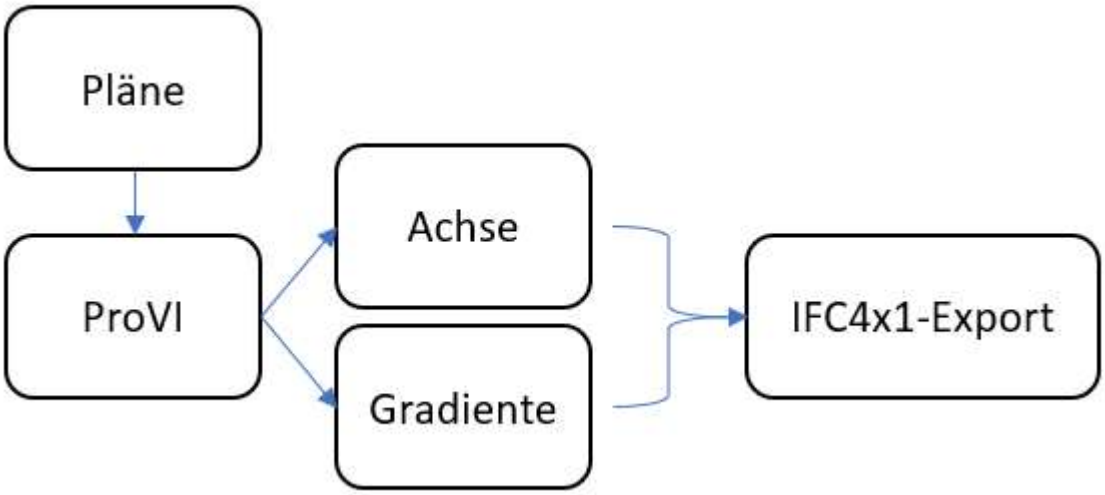


Abbildung 11: Prozess Modellierung der Achse und Gradiente

Für die Modellierung der Achse, an der ich alle zukünftig hinzugefügten Element positionieren werden, habe ich das Programm ProVI der Firma OBERMEYER Planen + Beraten GmbH benutzt.

Um die Achse der Brücke nachzumodellieren, habe ich zuerst die nötigen digitalen Pläne ([16], [17] & [18]) in ProVI als Hintergrund eingelesen und mit Hilfe von Kilometrierungsangaben auf dem Plan richtig skaliert. Die Kilometrierungsangaben sowie die Achsenbezeichungen in den Plänen wie Radius- und Klothoidenparameter und Unendlichkeitspunkte. Abbildung 12 zeigt einen kleinen Ausschnitt der Achse mit Hintergrund in ProVI.

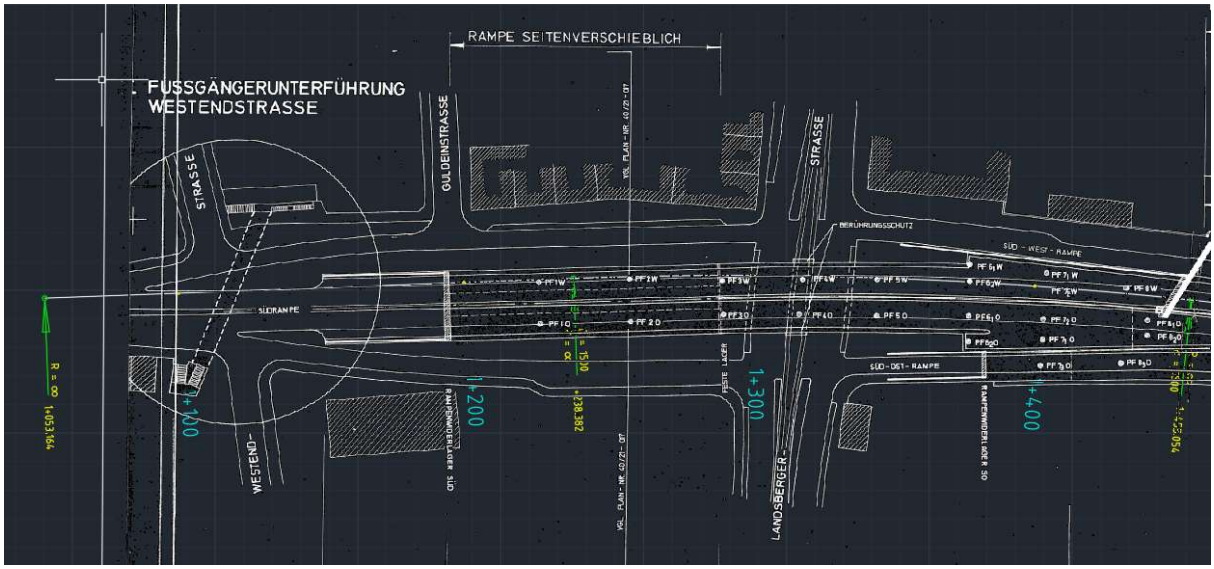


Abbildung 12: Ausschnitt der Ansicht in ProVI

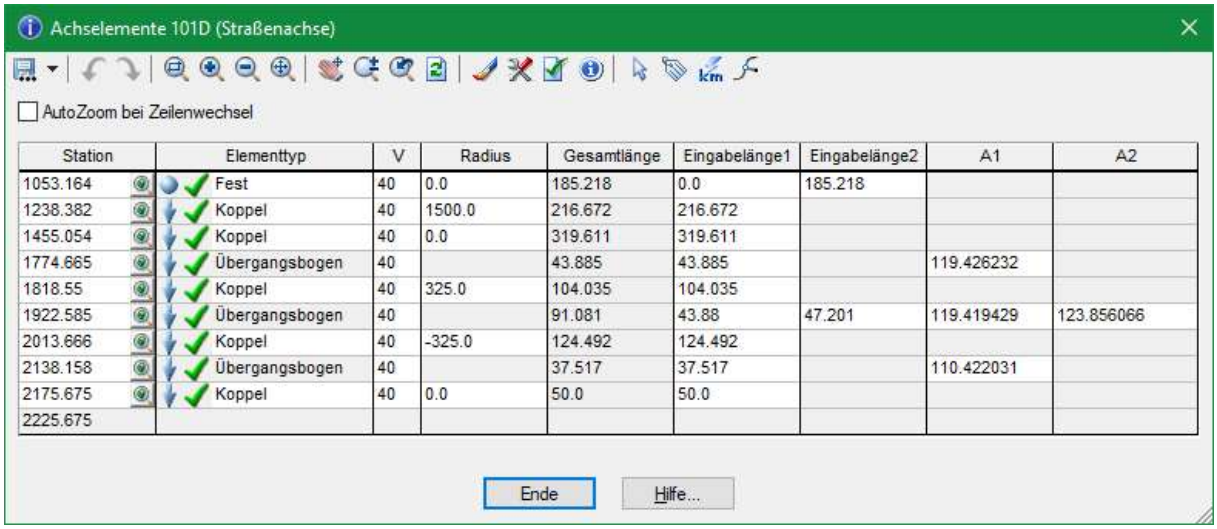
Der kleine Abschnitt in der Mitte der Brücke ist nur auf der Gesamtskizze zu sehen und es gibt keine spezielle Detailansicht. Da hierbei die Skalierung sehr schwierig war, habe ich den Verlauf realitätsnah in unserer Planung angenommen.

In Tabelle 1 sind die Parameter und Daten aufgelistet, die ich aus den Plänen ([16] & [19]) übernommen habe. Die Station beschreibt den Kilometrierungspunkt, an dem der Elementtyp anfängt. Dieser Elementtyp verläuft immer bis zur nächsten Station und verbindet somit das Element in der Tabelle darüber mit dem darunter.

Tabelle 1: Parameter der finalen Alignmentdatei [16] & [19]

Station	Elementtyp	Radius	Klothoidenparameter
1053.164	Gerade	0 m	
1238.382	Bogen rechts	1500 m	
1455.054	Gerade	0 m	
1774.665	Übergangsbogen		120 m
1818.55	Bogen rechts	325 m	
1922.585	Übergangsbogen		120 m
2013.666	Bogen links	325 m	
2138.158	Übergangsbogen		110 m
2175.675	Gerade	0 m	

In Abbildung 13 ist der Ausschnitt der Dialogbox von ProVI abgebildet.



Station	Elementtyp	V	Radius	Gesamtlänge	Eingabelänge1	Eingabelänge2	A1	A2
1053.164	Fest	40	0.0	185.218	0.0	185.218		
1238.382	Koppel	40	1500.0	216.672	216.672			
1455.054	Koppel	40	0.0	319.611	319.611			
1774.665	Übergangsbogen	40		43.885	43.885		119.426232	
1818.55	Koppel	40	325.0	104.035	104.035			
1922.585	Übergangsbogen	40		91.081	43.88	47.201	119.419429	123.856066
2013.666	Koppel	40	-325.0	124.492	124.492			
2138.158	Übergangsbogen	40		37.517	37.517		110.422031	
2175.675	Koppel	40	0.0	50.0	50.0			
2225.675								

Abbildung 13: ProVI Dialog

Hierbei ist zu sehen, dass die von mir modellierte Trasse nahezu mit den auf den Plänen angegebenen Parametern übereinstimmt. Lediglich die Klothoidenparameter weichen ein wenig ab. Da ich jedoch die Werte der Kilometrierung als genauer angesehen habe als die Werte der Klothoidenparameter, wurden diese von mir angepasst, sodass die Klothoidenparameter nun ein wenig abweichen, aber alle anderen Parameter übereinstimmen.

Das gleiche Vorgehen habe ich auch bei der Modellierung der Gradiente genutzt, die mit der Entität IfcVerticalAlignment beschrieben wird. [16] Ich habe die Gradiente mit der mir möglichsten Genauigkeit umgesetzt, denn die Gradiente stellt die richtige vertikale Positionierung der Brückenelemente sicher.

Diese Achse mit Gradiente habe ich nach der Modellierung als IFC-Datei exportiert.

Diese Datei bildet die Grundlage meiner späteren finalen Datei. Anfang und Ende der finalen IFC-Datei entsprechen dem Anfang und dem Ende dieser Datei.

3.3 Modellierung von Deck und Kappen

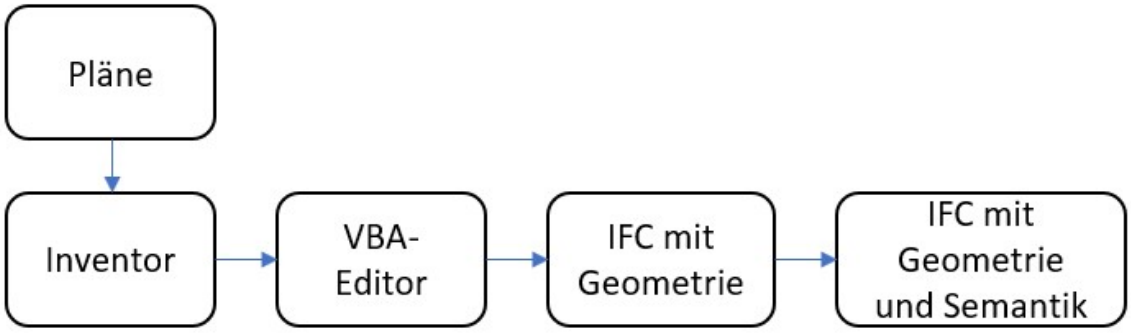


Abbildung 14: Prozess Modellierung Deck und Kappen

Die Donnersbergerbrücke besteht aus vielen unterschiedlichen Regelquerschnitten, die teilweise auch variable Parameter beinhalten, die eine detailgenaue Umsetzung erschwerten. Diese Regelquerschnitte bestehen zum einen aus dem Fahrbahnprofil mit Unterbau und Hohlräumen und zum anderen aus den Kappen links und rechts an den Regelquerschnitten, die den Gehweg und die seitliche Absperrung darstellen.

Ich habe bei der Nachmodellierung der Regelquerschnitte einige Details vernachlässigt und einige Änderungen vorgenommen. Eine Änderung ist die Geometrie des Stahlüberbaus aus Abbildung 15. Diese Geometrie ist sehr komplex und ich habe ihn für mein Projekt durch den Regelquerschnitt in Abbildung 16 ersetzt. Der Regelquerschnitt in Abbildung 16 kommt auf den Plänen im Anschluss zu den Regelquerschnitten mit Stahlüberbau. Ich habe somit den Beginn des Decks in Abbildung 16 vorgezogen auf den Beginn des Regelquerschnitts mit Stahlüberbau. In Tabelle 2 ist eine Übersicht über meine umgesetzten Regelquerschnitte aufgelistet. „RW_WLSued_bis_Pfeiler17“ ist der in Abbildung 16 gezeigte Regelquerschnitt. [20] & [21]

Tabelle 2: Modellierte Regelquerschnitte mit Daten aus [20]

Regelquerschnitt	Von:	Bis:
RQ_RWLSued_bis_Pfeiler3	1136.74 m	1240.75 m
RQ_Pfeiler3_bis_Pfeiler6	1240.75 m	1332.42 m
RQ_Pfeiler6_bis_WLSued	1332.42 m	1410.01 m
Vernachlässigter Stahlüberbau	-	-
RW_WLSued_bis_Pfeiler17	1410.01 m	1706.08 m
RQ_Pfeiler17_bis_WLNord	1706.08 m	2225.675 m

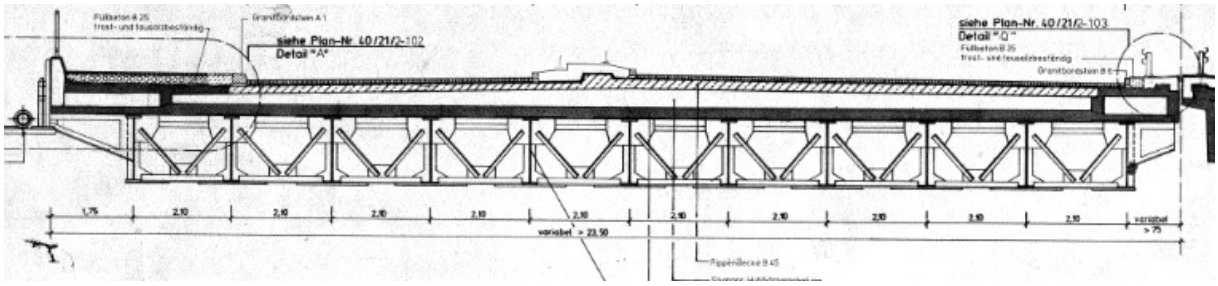


Abbildung 15: Regelquerschnitt Stahlüberbau [20]

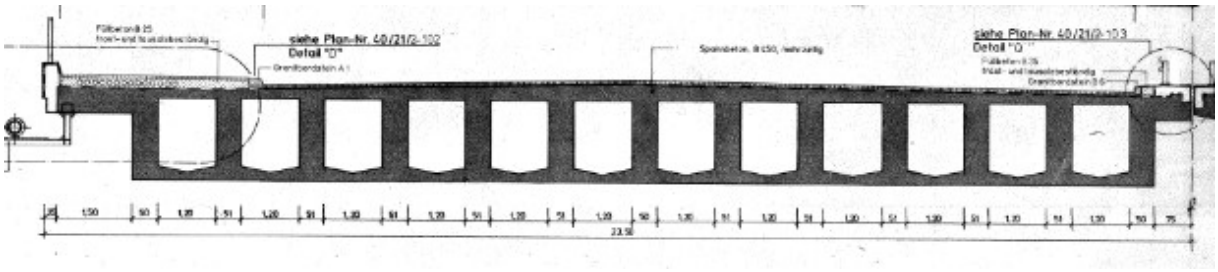


Abbildung 16: Ersatzregelquerschnitt für den Stahlüberbau [20]

Zur Vereinfachung der Konstruktion habe ich auch die variablen Anteile wie statische Parameter behandelt und ihnen einen festen Wert zugeteilt, sowie die Straßenausstattung nicht berücksichtigt.

Bei der Modellierung von Deck und Kappen für mein Projekt hat mir die Arbeit von Herrn Philipp Fengler am Lehrstuhl für Computergestützte Modellierung und Simulation geholfen. Er hat parametrisierbare Polylinien in Inventor erstellt und durch einen VBA-Code die Möglichkeit geschaffen, diese Geometrien im IFC-Format zu exportieren. [22]

Am Beispiel des Regelquerschnittes in Abbildung 16 will ich zeigen, wie ich die Decks erstellt habe.

In Abbildung 17 ist eine Polylinie in Inventor eines Querschnitts dargestellt. Dieser Querschnitt ist in der Mitte gespiegelt und somit kann er nur für symmetrische Querschnittsprofile verwendet werden. In diesem Bild sieht man die Bemaßungen, die den Linien zugewiesen wurden. Zuerst habe ich die Parameterfenster des Profils geöffnet (siehe Abbildung 18) und die benötigten Parameter (siehe Abbildung 20) dort verändert. Nach der Anpassung der Werte in dem Dialogfenster, habe ich den in Abbildung 19 gezeigten Querschnitt erhalten.

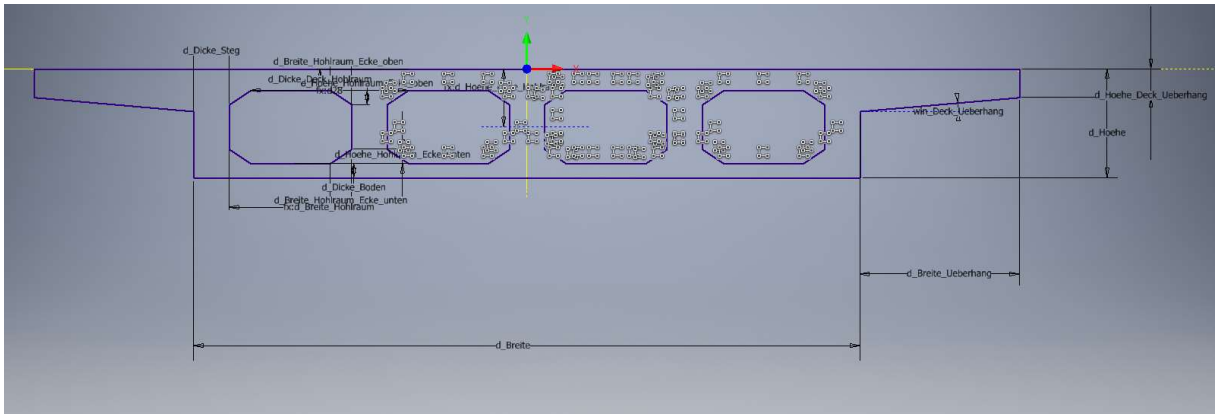


Abbildung 17: Profilansicht eines Decks in Inventor [22]

Parameter

Parametername	Einbezogen von	Einheit	Gleichung	Nennwert	Tol.	Modell	Schlüss	Kommentar
Modellparameter								
win_Steigung	win_BKS1_3	grd	func_Umrechnung_prozent_Steigung_win_Steigung	0,000000		0,00...		
d_BKS1_x	BKS1	m	d_Referenzpunkt_A_x	0,000000		0,00...		
d_BKS1_y	BKS1	m	d_Referenzpunkt_A_y	0,000000		0,00...		
d_BKS1_z	BKS1	m	0,000 m	0,000000		0,00...		
win_BKS1_1	BKS1	grd	0,00 grd	0,000000		0,00...		
win_BKS1_2	BKS1	grd	0,00 grd	0,000000		0,00...		
win_BKS1_3	BKS1	grd	-win_Steigung	-0,000000		-0,0...		
d_Breite	d_Offset_Deck_Kappen_x, ...	m	9,4 m	9,400000		9,40...		
d_Hoehe	d_Hoehe_Mitte_Hohlraum, ...	m	1,53 m	1,530000		1,53...		
d_Breite_Ueberhang	d_Offset_Deck_Kappen_x, ...	m	2,25 m	2,250000		2,25...		
d_Hoehe_Deck_Ueberhang	Skizze_Profil	m	0,4 m	0,400000		0,40...		
win_Deck_Ueberhang	Skizze_Profil	grd	5 grd	5,000000		5,00...		
d_Dicke_Boden	d_Hoehe_Mitte_Hohlraum, ...	m	0,2 m	0,200000		0,20...		
d_Dicke_Steg	d_Hohlraum_Abstand, func...	m	0,5 m	0,500000		0,50...		
d_Dicke_Deck_Hohlraum	d_Hoehe_Mitte_Hohlraum, ...	m	0,3 m	0,300000		0,30...		
d_Breite_Hohlraum_Ecke_oben	Skizze_Hohlraum	m	0,3 m	0,300000		0,30...		
d_Hoehe_Hohlraum_Ecke_oben	Skizze_Hohlraum	m	0,2 m	0,200000		0,20...		
d_Breite_Hohlraum_Ecke_unten	Skizze_Hohlraum	m	0,3 m	0,300000		0,30...		
d_Hoehe_Hohlraum_Ecke_unten	Skizze_Hohlraum	m	0,2 m	0,200000		0,20...		
d_Breite_Hohlraum	d_Hohlraum_Abstand, func...	m	func_Breite_Hohlraum	1,725000		1,72...		
d_Hoehe_Mitte_Hohlraum	Skizze_Hohlraum	m	d_Dicke_Deck_Hohlraum + 0,5 oE * (d_Hoehe - d_Dicke_Deck_Hohlraum - d_Dicke_Boden)	0,815000		0,81...		
d26	Skizze_Hohlraum	oE	n_Hohlraum_Anzahl	4,000000		4,00...		
d28	Skizze_Hohlraum	m	d_Hohlraum_Abstand	2,225000		2,22...		
d_Referenzpunkt_A_x	d_BKS1_x	m	0 m	0,000000		0,00...		
d_Referenzpunkt_A_y	d_BKS1_y	m	0 m	0,000000		0,00...		
Benutzerparameter								
prozent_Steigung	func_Umrechnung_prozent...	oE	0 oE	0,000000		0,00...		
func_Umrechnung_prozent_Steigung_...	win_Steigung	grd	atan(prozent_Steigung / 100 oE)	0,000000		0,00...		
n_Hohlraum_Anzahl	d26, func_Breite, func_Brei...	oE	4 oE	4,000000		4,00...		
			(d_Breite - (					

Numerischen Parameter hinzufügen | Aktualisieren | Nicht verwendete bereinigen | Toleranz zurücksetzen | << Weniger

Verknüpfen | Sofort aktualisieren | Fertig

Abbildung 18: Parameter des ursprünglichen Profils [22]

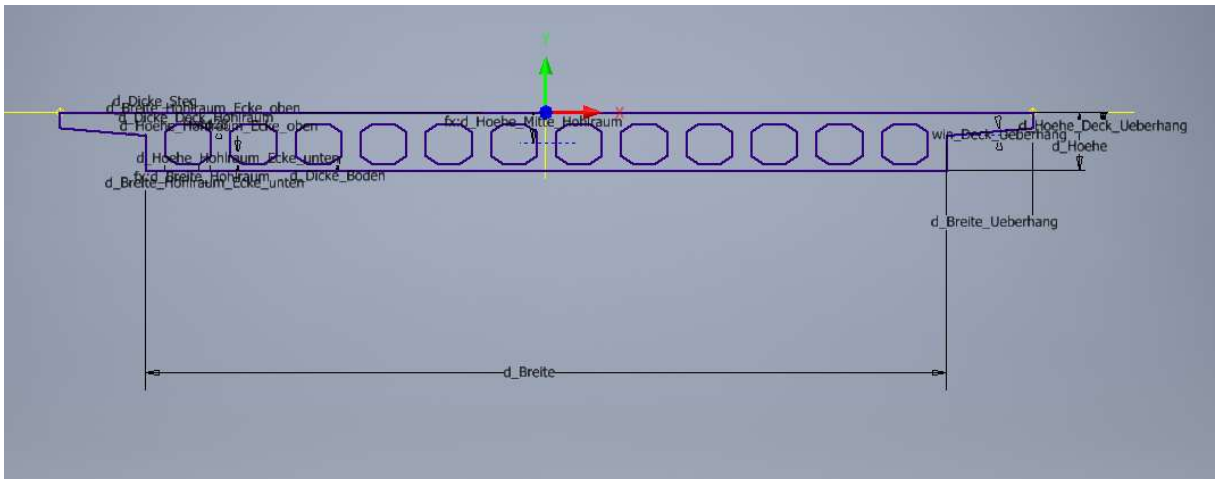


Abbildung 19: Profil mit neuen Parametern in Inventor vgl. [22]

Parameter

Parametername	Einbezogen von	Einheit	Gleichung	Nennwert	Tol.	Modellv	Schlüss	E	Kommentar
Modellparameter									
win_Steigung	win_BKS1_3	grd	func_Umrechnung_prozent_Steigung_win_Steigung	0,000000		0,00...			
d_BKS1_x	BKS1	m	d_Referenzpunkt_A_x	0,000000		0,00...			
d_BKS1_y	BKS1	m	d_Referenzpunkt_A_y	0,000000		0,00...			
d_BKS1_z	BKS1	m	0,000 m	0,000000		0,00...			
win_BKS1_1	BKS1	grd	0,00 grd	0,000000		0,00...			
win_BKS1_2	BKS1	grd	0,00 grd	0,000000		0,00...			
win_BKS1_3	BKS1	grd	-win_Steigung	-0,000000		-0,0...			
d_Breite	d_Offset_Deck_Kappen_x, ...	m	21 m	21,000000		21,0...			
d_Hoehe	d_Hoehe_Mitte_Hohlraum, ...	m	1,53 m	1,530000		1,53...			
d_Breite_Ueberhang	d_Offset_Deck_Kappen_x, ...	m	2,25 m	2,250000		2,25...			
d_Hoehe_Deck_Ueberhang	Skizze_Profil	m	0,4 m	0,400000		0,40...			
win_Deck_Ueberhang	Skizze_Profil	grd	5 grd	5,000000		5,00...			
d_Dicke_Boden	d_Hoehe_Mitte_Hohlraum, ...	m	0,2 m	0,200000		0,20...			
d_Dicke_Steg	d_Hohlraum_Abstand, func...	m	0,51 m	0,510000		0,51...			
d_Dicke_Deck_Hohlraum	d_Hoehe_Mitte_Hohlraum, ...	m	0,3 m	0,300000		0,30...			
d_Breite_Hohlraum_Ecke_oben	Skizze_Hohlraum	m	0,3 m	0,300000		0,30...			
d_Hoehe_Hohlraum_Ecke_oben	Skizze_Hohlraum	m	0,2 m	0,200000		0,20...			
d_Breite_Hohlraum_Ecke_unten	Skizze_Hohlraum	m	0,3 m	0,300000		0,30...			
d_Hoehe_Hohlraum_Ecke_unten	Skizze_Hohlraum	m	0,2 m	0,200000		0,20...			
d_Breite_Hohlraum	d_Hohlraum_Abstand, func...	m	func_Breite_Hohlraum	1,197500		1,19...			
d_Hoehe_Mitte_Hohlraum	Skizze_Hohlraum	m	d_Dicke_Deck_Hohlraum + 0,5 oE * (d_Hoehe - d_Dicke_Deck_Hohlraum - d_Dicke_Boden)	0,815000		0,81...			
d26	Skizze_Hohlraum	oE	n_Hohlraum_Anzahl	12,000000		12,0...			
d28	Skizze_Hohlraum	m	d_Hohlraum_Abstand	1,707500		1,70...			
d_Referenzpunkt_A_x	d_BKS1_x	m	0 m	0,000000		0,00...			
d_Referenzpunkt_A_y	d_BKS1_y	m	0 m	0,000000		0,00...			
Benutzerparameter									
prozent_Steigung	func_Umrechnung_prozent...	oE	0 oE	0,000000		0,00...			
func_Umrechnung_prozent_Steigung_...	win_Steigung	grd	atan(prozent_Steigung / 100 oE)	0,000000		0,00...			
n_Hohlraum_Anzahl	d26, func_Breite, func_Brei...	oE	12 oE	12,000000		12,0...			
			(d_Breite - (						

Numerischen Parameter hinzufügen | Aktualisieren | Nicht verwendete bereinigen | Toleranz zurücksetzen | << Weniger

Verknüpfen | Sofort aktualisieren | Fertig

Abbildung 20: Geänderte Parameter des Profils vgl. [22]

Um die Geometrie des Querschnitts zu exportieren, wurde von Philipp Fengler ein Code im VBA-Editor entwickelt, um die Koordinaten im IFC-Schema zu exportieren und die Entität des IfcSectionedSolidHorizontal einzubinden. [22]

Da die IFC-Datei nach dem Export nur die reine Geometrie der Regelquerschnitte enthält (wie bereits in 2.3.3 IfcSectionedSolidHorizontal erwähnt), habe ich die IFC-Dateien der Decks noch, durch Hinzufügen von den in Abbildung 21 gezeigten Entitäten und das richtige Verknüpfen der Parameter, händisch in Objekte mit semantischer Bedeutung verwandelt.

```
#95 = IFCSLAB('0oo95mAXjAB8ZG7EUGVO20',$,'Deck A',$,$,#38,#96,$,.NOTDEFINED.);
#96 = IFCPRODUCTDEFINITIONSHAPE($,$,(#97));
#97 = IFCSHAPEREPRESENTATION(#28,'Body','AdvancedSweptSolid',(#94));
```

Abbildung 21: Händisch hinzugefügte Entitäten zur Darstellung der Decks

Die von mir hinzugefügten Entitäten beinhalten Informationen über das Objekt (IfcSlab) und seine Profildarstellung (IfcShapeRepresentation). IfcProductDefinitionShape verbindet die Profildarstellung mit dem Objekt. Die Art und Weise wie das Deck aussieht wird in IFC4x1 durch IfcSectionedSolidHorizontal beschrieben und auf diese Entität verweist die IfcShapeRepresentation des Decks. In Abbildung 22 ist ein Ausschnitt der IFC-Datei eines Decks zu sehen.

```
#54=IFCCARTESIANPOINT((6.77, -0.17));
#55=IFCCARTESIANPOINT((6.5, -0.16));
#56=IFCCARTESIANPOINT((6.5, 0.));
#57=IFCPOLYLINE(( #33, #34, #35, #36, #37, #38, #39, #40, #41, #42, #43, #44, #33));
#58=IFCARBITRARYCLOSEDPROFILEDEF(.AREA., 'Kappe1' , #57);
#59=IFCPOLYLINE(( #45, #46, #47, #48, #49, #50, #51, #52, #53, #54, #55, #56, #45));
#60=IFCARBITRARYCLOSEDPROFILEDEF(.AREA., 'Kappe2' , #59);
#61=IFCDISTANCEEXPRESSION(1240.75, 14.85, 0., 0.,.T.);
#62=IFCDISTANCEEXPRESSION(1332.42, 14.85, 0., 0.,.T.);
#63=IFCSECTIONEDSOLIDHORIZONTAL(HIERKOMMTDIEALIGNMENTCURVE,( #58, #58), (#61, #62),.T.);
#64=IFCDISTANCEEXPRESSION(1240.75, 14.85, 0., 0.,.T.);
#65=IFCDISTANCEEXPRESSION(1332.42, 14.85, 0., 0.,.T.);
#66=IFCSECTIONEDSOLIDHORIZONTAL(HIERKOMMTDIEALIGNMENTCURVE,( #60, #60), (#64, #65),.T.);
#67 = IFCSLAB('0oxzUittjAB5zG7EUGVO20',$,'RqBereichPfeiler3Bis6',$,$,#38,#68,$,.NOTDEFINED.);
#68 = IFCPRODUCTDEFINITIONSHAPE($,$,(#69));
#69 = IFCSHAPEREPRESENTATION(#28,'Body','AdvancedSweptSolid',(#32));
#70 = IFCSLAB('0oPoitzio5z8ZG7EUGVO20',$,'Kappe11',$,$,#38,#71,$,.NOTDEFINED.);
#71 = IFCPRODUCTDEFINITIONSHAPE($,$,(#72));
#72 = IFCSHAPEREPRESENTATION(#28,'Body','AdvancedSweptSolid',(#63));
#73 = IFCSLAB('0ooPloTXj5zrut7EUGVO20',$,'Kappe12',$,$,#38,#74,$,.NOTDEFINED.);
#74 = IFCPRODUCTDEFINITIONSHAPE($,$,(#75));
#75 = IFCSHAPEREPRESENTATION(#28,'Body','AdvancedSweptSolid',(#66));
```

Abbildung 22: IFC-Datei eines Decks mit IFC4x1-Entitäten

3.4 Modellierung der Stützen

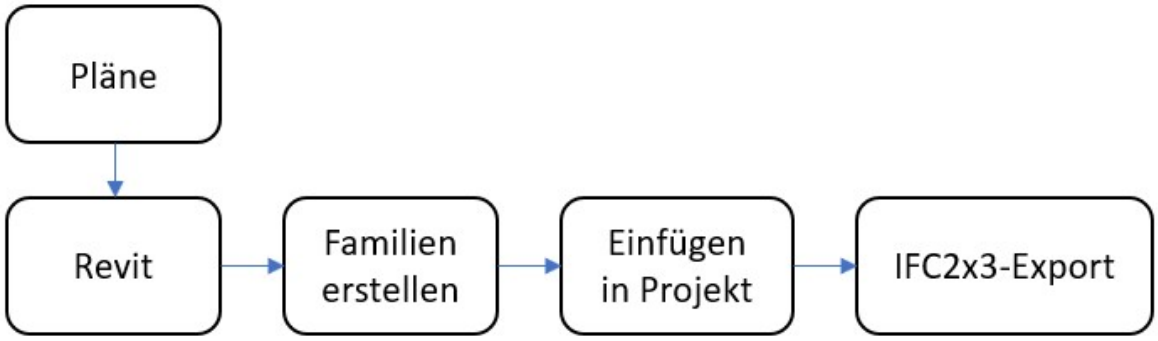


Abbildung 23: Prozess Modellierung der Stützen

Autodesk Revit unterstützt einen IFC2x3 Export, der für meine Arbeit ein wichtiges Kriterium darstellt. Wie in Abbildung 24 zu sehen ist, sind in den Bauplänen ([23] und [24]) der Donnersbergerbrücke die Pfeiler sehr genau dokumentiert.

Es gibt von jeder Stütze eine Seitenansicht und eine Draufsicht mit Bemaßung. Die

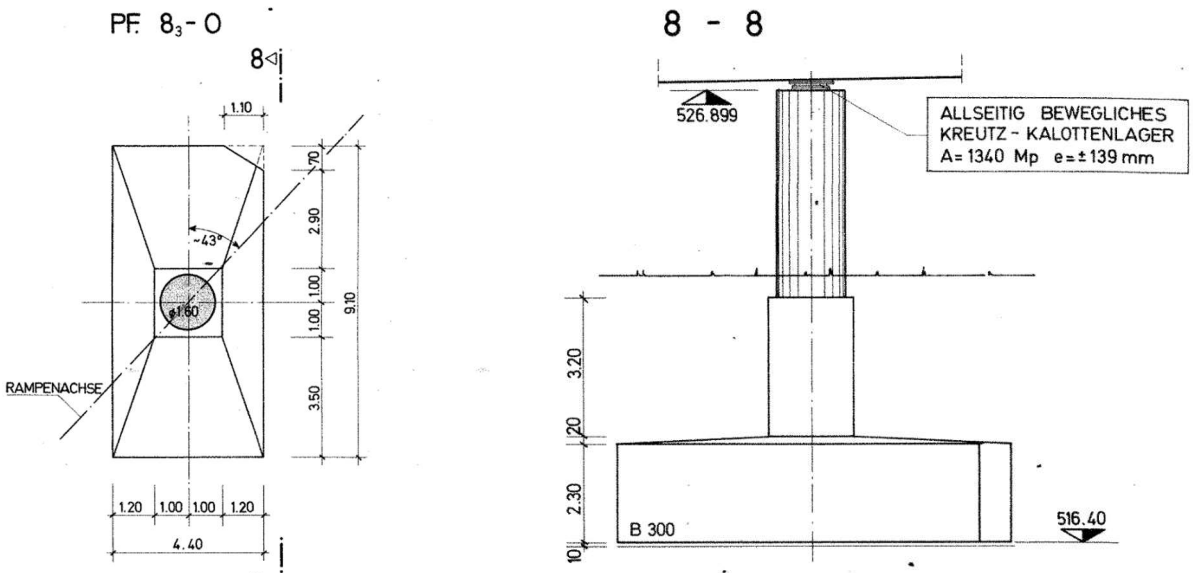


Abbildung 24: Dokumentationsausschnitt aus den Plänen [23]

Pfeiler bestehen aus einer Säule und einem Fundament. Diese sind mit Bezeichnungen versehen, da sie teilweise mehrfach mit den gleichen Bemaßungen verwendet wurden.

Um einzelne IFC-Dateien zu erhalten, die jeweils nur einen Teil der Stütze enthalten, habe ich jedes Fundament, getrennt von dem Teil der Säule, in einer eigenen Familie modelliert. Da in fast allen Pfeilern die gleiche Säule verwendet wurde und sie sich lediglich in der Länge unterscheiden, habe ich in Revit nur eine Säule modelliert. Diese

können wir anschließend in der Programmierung in der Länge ändern und auf alle Pfeiler anpassen.

Ich habe die Modellierung in der Detailreiche von LoD200 konstruiert. Franziska Mini hat dies in ihrer Masterarbeit LoD200 anhand einer Stahlstütze erklärt.

„LoD200: Die Stahlstütze wird typgerecht mit ungefährer Menge, Abmessung, Form, Lage und Orientierung dargestellt.“ [25]

Nun zum Konstruktionsvorgang der einzelnen Fundamente.

Die Fundamente habe ich mittels „Form erstellen“ (englisch: Loft) modelliert. Das „Lofften“ ist eine Funktion, die es ermöglicht aus Rahmenlinien einen Körper zu erstellen. Um diese Funktion zu verwenden, war es notwendig die Fundamente in der Familienvorlage M_Körper zu erstellen, weil sie nur dort zur Verfügung steht. Somit heißen die Fundamente in der IFC-Datei nicht IfcFootage sondern IfcBuildingElementProxy.

Als aller erstes habe ich in Revit mehrere Ebenen festgelegt, die ich zum Arbeiten mit dem „Loft“ brauche. Ein Beispiel dieser Ebenen ist in Abbildung 25 zu sehen.

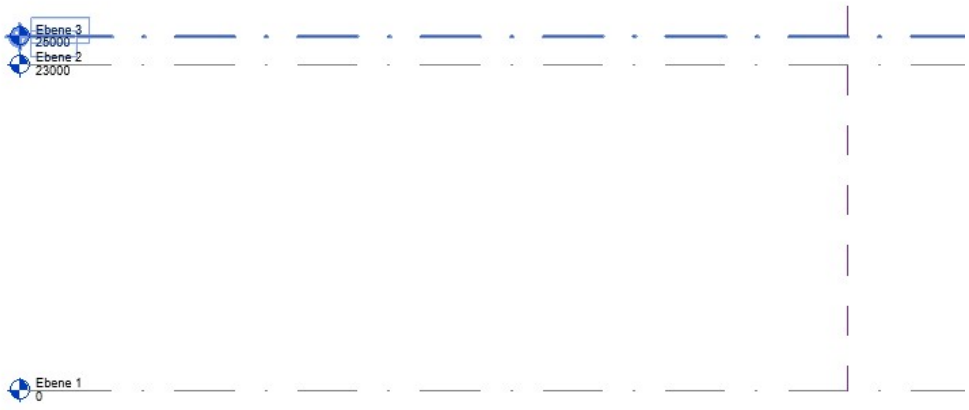


Abbildung 25: Ebenenansicht in Revit

Die Abstände dieser Ebenen waren die Höhen der einzelnen Abschnitte des Fundamentes an denen sich Änderungen im Querschnitt befinden. Die Loft-Funktion verbindet eine Polylinie mit einer anderen und erstellt dadurch einen massiven Körper.

Da nahezu alle Fundamente aus einem viereckigen Rumpf bestehen, habe ich auf der untersten Ebene den Grundriss dieses Fundamentes als Linie gezeichnet. Das Beispiel aus der Abbildung 24 zeigt ein Fundament mit einem fehlenden Eck. Diese Einzelheit wird erst später betrachtet.

In Abbildung 26 sieht man das Grundgerüst des Fundamentes aus einzelnen Polylinien.

Nun kann mittels der Funktion Loft (im deutschen Revit: Form erstellen) diese Polylinien zum Körper verbunden werden, der in Abbildung 27 zu sehen ist. Da dies leider noch nicht dem eigentlichen Fundament aus dem Plan entspricht, musste ich noch die Ecke mit Hilfe eines Abzugskörpers entfernen.

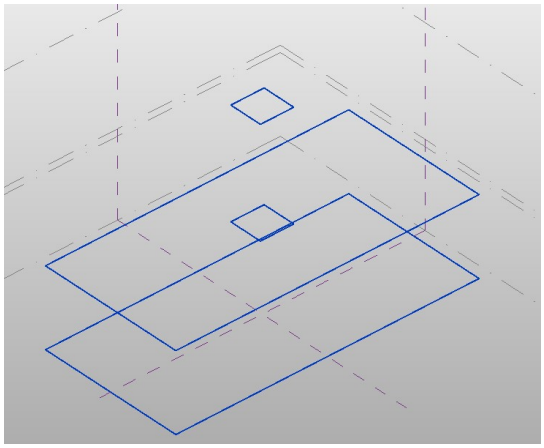


Abbildung 26: Rohform aus Polylinien

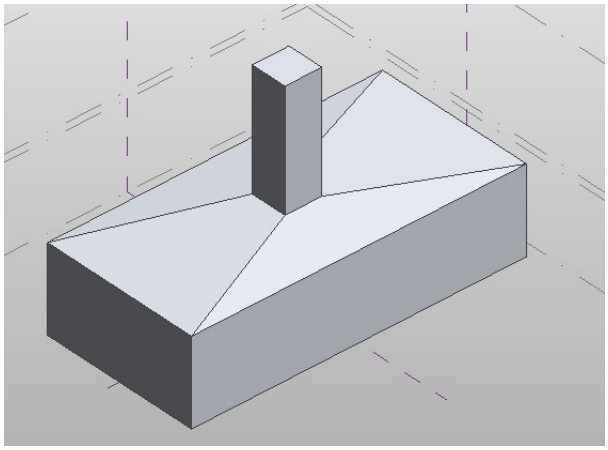


Abbildung 27: Gefüllter Körper nach Loft

In Abbildung 28 sehen Sie die finale Umsetzung eines Stützfundamentes in Revit. Dieses Fundament wird nun in ein leeres Projekt platziert und dort als IFC-Datei exportiert.

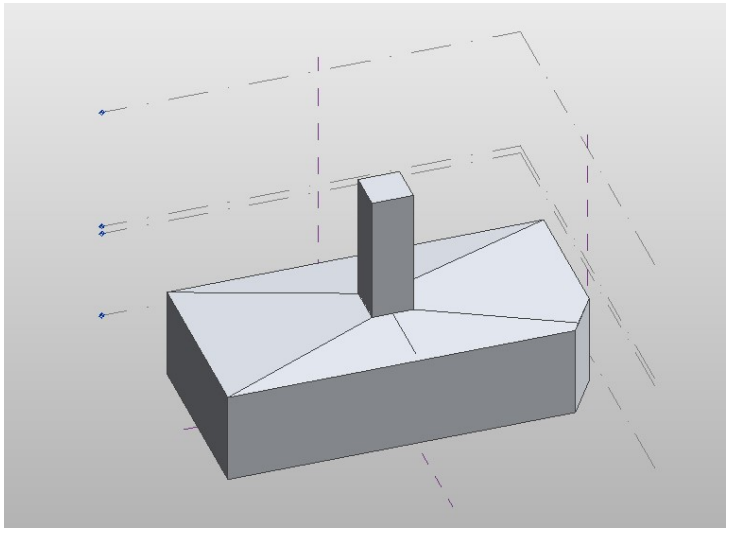


Abbildung 28: Finales Fundament mit abgeschnittener Ecke

In den Plänen sind einige Fundamente und Stützen mit Details versehen, wie zum Beispiel Auflager oder Gebäudeausstattung wie Lüftungskanäle und Abwasserleitungen. Diese habe ich im Hinblick auf das Level of Detail 200 vernachlässigt.

In meinem Projekt gibt es jedoch ein paar Pfeiler, die nicht diesem Aufbau von oben entsprechen. Diese Pfeiler stützen die Donnersbergerbrücke im Bereich der Bahnschienen.

Die Abbildung 30 zeigt einen dieser Pfeiler. Der Pfeiler wurde mit derselben Methodik konstruiert, wie die Fundamente. Das Grundgerüst zum „Loften“ bildet Abbildung 29 ab.

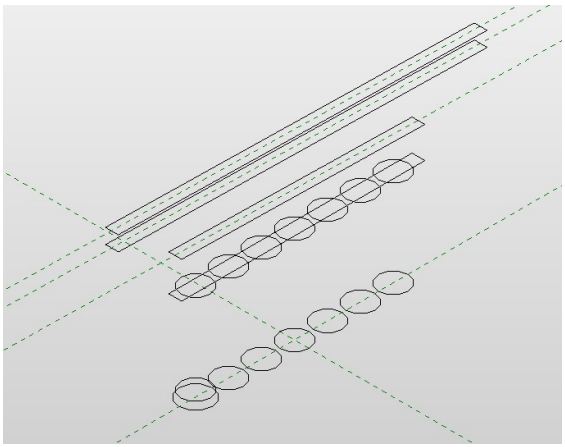


Abbildung 29: Rohform aus Polylinien

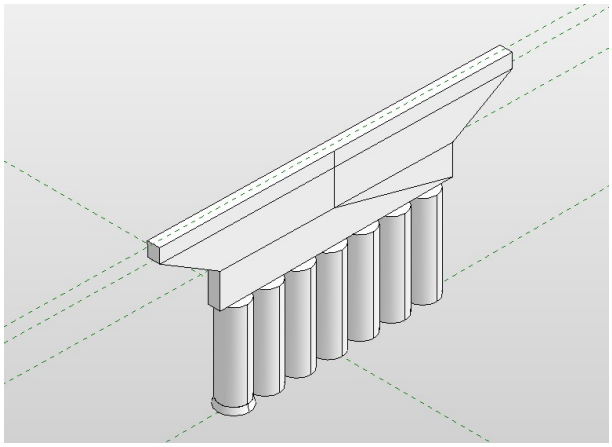


Abbildung 30: Gefüllter Körper nach Loft

Da sich Pfeiler 9 – 14a und I – VII bis auf Pfeiler 11 nur in kleinen Details unterscheiden, habe ich nur Pfeiler 10 modelliert und diesen Pfeiler an alle Stellen dieser Pfeilerart eingefügt. Die Sonderform des Pfeilers 11 habe ich nicht separat modelliert. Ich habe Pfeiler 10 nachmodelliert und diesen mit den richtigen Einsetzpunkten für alle oben genannten Pfeiler in mein Projekt eingefügt. [26]

Abbildung 31 zeigt einen Ausschnitt aus dem Hauptteil einer IFC-Datei.

```
#232= IFCBUILDINGELEMENTPROXYTYPE('3d85ZB$BzDd9ssCTTd6wkn',#41,'Fundament_1OW_2OW_Mass',$,$,$,(#230),'4562',$,.NOTDEFINED.);
#235= IFCCLASSIFICATION('http://www.csiorg.net/uniformat','1998',$,'Uniformat');
#237= IFCCARTESIANTRANSFORMATIONOPERATOR3D($,$,#6,1.,$);
#238= IFCMAPPEDITEM(#230,#237);
#240= IFCSHAPEREPRESENTATION(#102,'Body','MappedRepresentation',(#238));
#242= IFCPRODUCTDEFINITIONSHAPE($,$,(#240));
#246= IFCCARTESIANPOINT((-3000.,1572.,0.));
#248= IFCAXIS2PLACEMENT3D(#246,$,$);
#249= IFCLOCALPLACEMENT(#122,#248);
#251= IFCBUILDINGELEMENTPROXY('3d85ZB$BzDd9ssETTd6wkn',#41,'Fundament_1OW_2OW_Mass:Fundament_1OW_2OW_Mass:4592',$,'Fundament_1OW_2OW_Mass'
#260= IFCPROPERTYSINGLEVALUE('Reference',$,IFCIDENTIFIER('Fundament_1OW_2OW_Mass'),$);
#261= IFCPROPERTYSET('2SlZ3z0O978OK8LkTp6cY9',#41,'Pset_BuildingElementProxyCommon',$,(#260));
#266= IFCRELDEFINESBYPROPERTIES('17fc_kmuz4m8Q5$KPKxCc0',#41,$,$,(#251),#261);
#270= IFCAXIS2PLACEMENT3D(#6,$,$);
#271= IFCLOCALPLACEMENT($,#270);
#272= IFCSITE('03Gqh9apv2xOH7hzOXZ7Lp',#41,'Default',$,'',#271,$,$,.ELEMENT.,(42,24,53,508911),(-71,-15,-29,-58837),0.,$,$);
#277= IFCPROPERTYSINGLEVALUE('AboveGround',$,IFCLOGICAL(.U.),$);
#278= IFCPROPERTYSET('3d85ZB$BzDd9ssCZjd6xmB',#41,'Pset_BuildingStoreyCommon',$,(#277));
#280= IFCRELDEFINESBYPROPERTIES('0tRgFoisj6uOSw3kqLhVFk',#41,$,$,(#124),#278);
#284= IFCRELCONTAINEDINSPATIALSTRUCTURE('3d85ZB$BzDd9ssETPd6xmB',#41,$,$,(#251),#124);
#288= IFCRELAGGREGATES('0TfMFkHsL92hEYrt14jUsA',#41,$,$,#105,(#272));
#292= IFCRELAGGREGATES('1opKjYcwj5tPK38Ry37$lj',#41,$,$,#272,(#115));
#296= IFCRELAGGREGATES('3d85ZB$BzDd9ssETHd6xyK',#41,$,$,#115,(#124));
#300= IFCPROPERTYSINGLEVALUE('NumberOfStoreys',$,IFCINTEGER(1),$);
#301= IFCPROPERTYSET('3d85ZB$BzDd9ssCZnd6xyK',#41,'Pset_BuildingCommon',$,(#300));
#303= IFCRELDEFINESBYPROPERTIES('3OYLDYzQ1DjheQ_ZPC3fnK',#41,$,$,(#115),#301);
#310= IFCPRESENTATIONLAYERASSIGNMENT('A-MASS-____-OTLN',$,(#224,#240),$);
ENDSEC;
```

Abbildung 31: Beispiel einer IFC-Datei

Die Nummerierung der IFC-Typen ist nicht immer direkt aufeinanderfolgend, jedoch ist die nächste Zahl größer als die zuvor. Dieses Kriterium habe ich auch in meinem Programm beachtet, die im Folgenden näher erörtert wird. Der Eintrag #251 ist hierbei das Fundament als IfcBuildingElementProxy und vergibt der Geometrie eine semantische Bedeutung.

4 Fusion

Um die modellierten Stützen am Ende zu einer gesamten IFC-Datei zusammenzuführen gab es zwei Möglichkeiten. Die erste Möglichkeit war es, die exportierten kleinen IFC-Dateien per Texteditor händisch zu editieren. Da dies aber sehr aufwändig und fehleranfällig ist, entschied ich mich für die zweite Option:

Die Entwicklung eines Programms, das diese Arbeit für mich automatisiert erledigt. Dadurch habe ich die Fehleranfälligkeit der finalen IFC-Datei reduziert.

Für die Programmierung des Programms habe ich die Programmiersprache Python 3.6.5 und Visual Studio 2017 gewählt.

4.1 Python

Python ist eine freie, plattformübergreifende, Open-Source-Programmiersprache, die leistungsstark und einfach zu erlernen ist. Sie ist weit verbreitet und wird weitgehend unterstützt. [27]

4.2 Skripten

Abbildung 32 enthält eine Übersichtsskizze über den Prozess des Programms.

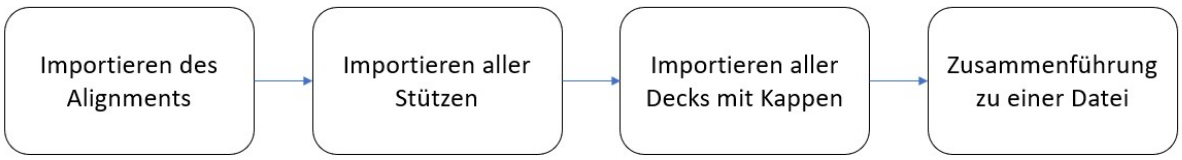


Abbildung 32: Prozess des Python-Programms als Skizze

Allgemeines:

GUI.py (siehe Anhang B):

Dieses Modul erstellt eine grafische Oberfläche mit drei Schaltflächen, die erscheint, sobald der Benutzer das Programm startet. Diese Schaltflächen sind mit Funktionen aus dem Modul MainIFC.py hinterlegt, die ausgeführt werden, wenn der Benutzer diese Flächen anklickt.

MainIFC.py (siehe Anhang B):

Da bei Python die globalen Variablen nur innerhalb eines Moduls funktionsübergreifend funktionieren, beinhaltet dieses Modul die Funktionen hinter den Schaltflächen der GUI. Somit können Variablen zwischen allen Funktionen des Programms übergeben werden.

Schaltfläche „Import Alignment“:

ImportAlignment.py (siehe Anhang B):

Dem Benutzer wird ein Explorer-Fenster geöffnet, in dem er die Alignment-Datei auswählen soll. Nachdem die Datei ausgewählt wurde, wird sie im Hintergrund geöffnet und an die StringManipulation.py Funktion übergeben.

Stringmanipulation.py (siehe Anhang B):

Die übergebene Datei wird hierbei zeilenweise eingelesen und abgespeichert. Hierbei wird der Hauptteil von Anfang und Ende getrennt und in separate Variablen abgespeichert.

Der Hauptteil wird nun Zeile für Zeile in drei Bereiche aufgespaltet und an eine Liste angehängt. Der Erste Teil an die Nummernliste, der zweite an die Typenliste und der dritte an die Parameterliste. Da der letzte hinzugefügte Eintrag an die Nummernliste gleichzeitig die höchste Nummer ist, wird zur weiteren Verwendung diese Zahl extra abgespeichert.

Anfang, Hauptteil, Ende und die höchste Zahl der Alignment-Datei werden nun an das Hauptmodul zurückgegeben.

Schaltfläche „Import Columns“:

ImportOptions.py (siehe Anhang B):

Als Erstes nach dem Auswählen der zweiten Schaltfläche für den Import der Stützen wird die Funktion importOptions() des oben genannten Moduls ausgeführt.

Diese Funktion öffnet den im Code hinterlegten Link zur Options-Datei. Die Datei beinhaltet Informationen über alle einzufügenden Stützen (siehe Abbildung 33). Ihren Dateinamen, ihre jeweilige Position (Y- und Z-Koordinaten), Position und Ausrichtung an der Achse und die individuelle Höhe der Stütze auf dem Fundament.

Diese Informationen werden jeweils als eigene Variable abgespeichert und an die ImportIFC_v2() Funktion des Moduls Import Columns.py übergeben.

ImportColumns.py (siehe Anhang B):

Die Funktion in diesem Modul wird für alle Stützen in der Options-Datei jeweils aufgerufen. Zuerst wird die erste Stütze im Hintergrund geöffnet und zeilenweise in eine Liste eingelesen. Danach wird die Funktion searchIndices_v1() des Moduls SearchIndices.py aufgerufen. Die veränderten Daten werden dann in eine zugeschnittene Form der stringManipulation-Funktion *(siehe Anhang 9.3.7)* übergeben. Diese trennt die Textdaten in die Nummern-, Typen- und Parameterliste auf.

Nach dem Hinzufügen der Entität müssen noch die beiden anderen Entitäten eingefügt und befüllt werden. Hierfür werden der Funktion IFC4x1_DistanceOrientationAdder() im Modul IFC4x1_DistanceOrientationAdder.py der Text, sowie alle Informationen aus der Options-Datei und die drei oben genannten Listen, übergeben.

SearchIndices.py (siehe Anhang B):

Da nicht alle Zeilen der IFC-Datei hinzugefügt werden müssen, weil sie bereits in der ersten IFC-Alignment-Datei vorhanden sind, werden hier nun alle wichtigen Zeilen ermittelt. Wichtig sind alle Zeilen, die IfcBuildingElementProxy oder die Zeilennummer davon beinhalten, sowie alle damit verknüpften Zeilen und deren verknüpften Zeilen.

Um die Zeilennummer und die Parameter von IfcBuildingElementProxy herauszufinden, wird die Datei zeilenweise nach dem Namen durchsucht. Bei einem Treffer wird

sein Index in der Indexliste abgespeichert, und die Zeile nach ihrer Zeilennummer und allen verknüpften Zeilennummern gefiltert. Zuerst wird jetzt die Datei nach der Zeilennummer von IfcBuildingElementProxy durchsucht, danach nach den Zeilennummern der verknüpften Zeilen von IfcBuildingElementProxy. Bei jedem Treffer, außer wenn es sich um die Zeile von IfcRelContainedInSpatialStructure handelt (diese Zeile darf in der finalen IFC-Datei nur einmal vorhanden sein), wird sein Index ausgelesen. Falls dieser Index noch nicht in der Indexliste vorhanden ist, wird er dieser angehängt.

Diese Indexliste wird nun mit dem Text an die Funktion searchIndices_v2 übergeben, die nun alle verknüpften Zeilennummern in den bereits als Index gespeicherten Zeilen sucht und wieder als Indices an die Liste anhängt. Diese Funktion ruft sich so lange selbst auf bis sie alle verknüpften Zeilen als Index in der Indexliste abgespeichert sind.

Die vollständige Liste aller wichtigen Zeilen wird anschließend an das ImportColumns.py-Modul zurückgegeben

Da alle IFC-Dateien mit der Nummerierung von eins beginnen und es keine doppelten Einträge geben darf, müssen alle hinzugefügten IFC-Dateien an die Alignment-Datei mit der Nummerierung dort beginnen, bei dem die zuvor hinzugefügte Datei aufgehört hat. Hierzu die Zeilen, die in der Indexliste sind, und die höchste IFC-Zahl zeilenweise an die Funktion regularExpression() im Modul RegularExpression übergeben.

RegularExpression.py (siehe Anhang B):

Zuerst sucht die Funktion mit Hilfe von Regular Expression alle „#“ mit einer Zahl dahinter und die Ergebnisse werden zwischengespeichert. Aus allen Ergebnissen werden die Zahlen nun vom „#“ getrennt und als Liste gespeichert und in Integer umformatiert, sowie absteigend sortiert.

Die aktuellen Nummern werden in einer separaten Liste gesichert, damit mit der anderen Liste die Berechnung durchgeführt werden kann.

Die Nummern werden um die höchste IFC-Zahl erhöht und mit dem „#“ wieder zu Zeilennummern und Strings gewandelt.

Die alten Nummern werden durch die neuen ersetzt und der String wieder zurückgegeben.

Nachdem alle Zeilen umgewandelt worden sind, sind für die IFC 4x1-Datei noch Entitäten hinzuzufügen. Da ich die Stützen anhand einer Achse ausrichten will muss ich IfcLocalPlacement durch IfcLinearPlacement ersetzen und danach IfcDistanceExpression und IfcOrientationExpression hinzufügen. Für den ersten Teil wird der Text der Datei nun in die Funktion IFC4x1_LinearPlacementAdder() im Modul IFC4x1_LinearPlacementAdder.py übergeben.

IFC4x1_LinearPlacementAdder.py (siehe Anhang B):

Der übergebene Text wird zeilenweise nach dem Wort „IfcLocalPlacement“ durchsucht. Bei allen Treffern dieser Suche merkt sich die Funktion seine Zeilennummer. Alle Treffer werden nun aus dem Text entfernt, außer der Erste, denn diesen String merkt sich die Funktion, um ihn anschließend durch die neue Entität auszutauschen. Diese Entität ist mit Platzhaltern ausgestattet, sodass nach dem hinzufügen von IfcDistanceExpression und IfcOrientationExpression, die Zeilennummern dort eingefügt werden können. Der Text wird wieder zurückgegeben.

Die veränderten Daten werden nun in eine zugeschnittene Form der stringManipulation-Funktion übergeben. Diese trennt die Textdaten in die Nummern-, Typen- und Parameterliste auf.

Nach dem Hinzufügen der Entität müssen noch die beiden anderen Entitäten eingefügt und befüllt werden. Hierfür werden der Funktion IFC4x1_DistanceOrientationAdder() im Modul IFC4x1_DistanceOrientationAdder.py der Text, sowie alle Informationen aus der Options-Datei und die drei oben genannten Listen, übergeben.

IFC4x1_DistanceOrientationAdder.py (siehe Anhang B):

In diesem Modul werden die zwei Entitäten unten an die momentane Datei angehängt und mit zugehörigen Werten befüllt. Zudem werden die Einträge auch getrennt an die Listen angehängt. Die Platzhalter werden zur Verknüpfung mit IfcLinearPlacement durch die Zeilennummern ersetzt. Der Text und die drei Listen werden wieder zurückgegeben.

Schlussendlich werden die neue höchste IFC-Nummer, die Listen, sowie der Text an das MainIFC-Modul zurückgegeben.

Schaltfläche „Import all Decks“:

importIFCDeck (siehe Anhang B):

Dieser Funktion werden die einzelnen Datenlisten der Nummern, Parametern und Typen und die höchste Nummer meiner bisherigen IFC-Datei übergeben. Alle IFC-Dateien der Decks werden nacheinander aufgerufen, eingelesen und mit dem bereits erwähnten regularExpression.py-Modul richtig nummeriert. Anschließend werden sie durch eine abgewandelte Form des stringManipulation.py-Moduls wieder den Datenlisten angehängt. Dieses abgewandelte Model ist stringManipulationDeck.

StringManipulationDeck.py (siehe Anhang B):

Aufgabe dieses Moduls ist es, zusätzlich zum Anhängen an die Datenlisten, die Nummer von der IfcAlignmentCurve an die richtige Stelle anzuhängen. Danach werden alle Datenlisten wieder zurück an das MainIFC-Modul gegeben.

Schaltfläche „Merge files“:

Reunion.py (siehe Anhang B):

Die Funktion reunion() führt die bearbeiteten Daten am Schluss zusammen. Der Funktion werden beim Auswählen der Schaltfläche die Variablen übergeben, die den Anfang und das Ende der Alignment-Datei beinhalten und auch die Listen sowie den veränderten Hauptteil mit allen hinzugefügten IFC-Dateien. Nun werden alle Teile wieder zeilenweise zusammengeführt.

Bevor unsere IFC-Datei jedoch richtig gelesen werden kann muss die Zeilennummer jedes IFC-Elementes noch in die Entität IfcRelContainedInSpatialStructure eingetragen werden und an die Funktion RelContAdder() im Modul RelConAdder.py übergeben.

RelContAdder.py (siehe Anhang B):

Zuallererst wird die Zeile der Entität IfcRelContainedInSpatialStructure in der alle Elemente eingetragen werden sollen gesucht. Wenn die Zeile gefunden wurde, werden alle IfcBuildingElementProxy's, IfcColumn's und IfcSlab's aus der Datei gesucht und die Zeilennummer in die Entität übertragen.

Der gesamte Text ist nun fertig formatiert und wird wieder zum Speichern zurückgegeben.

Es öffnet sich ein Explorer-Fenster und der Benutzer kann den Speicherort der finalen Datei auswählen. Die Datei wird dann am ausgewählten Speicherort als gültige IFC4x1-Datei abgespeichert.

5 Diskussion

Die Implementierung von neu entwickelten IFC-Datenmodellsversion in Programme ist langwierig. Die neuen Versionen wurden noch nicht ausreichend getestet und die Möglichkeit besteht, dass noch nachgebessert werden muss, bevor es zur Implementierung kommt. In meiner Arbeit habe ich die neue IFC4x1-Version mit Hilfe eines Brückenmodells getestet.

Meine Arbeit kann als Machbarkeitsstudie für die Entwickler des IFC-Datenmodells angesehen werden. Diese Machbarkeitsstudie habe ich zwar nur anhand einer Brücke bewiesen, jedoch kann daraus geschlussfolgert werden, dass das Datenmodell alle Brückenprojekte mit allen wichtigen Informationen darstellen kann.

Die Erkenntnis, dass das jetzige Datenmodell ausreicht, um Bauwerke darzustellen, hilft in der weiteren Entwicklung dabei den Fokus auf andere Teilbereiche zu legen und in der CAD-Branche ist dies der erste Meilenstein zur Integration in die vorhandenen Systeme. Die neu hinzugefügte Entität IfcAlignment ist im Stande die Trasse der Brücke mit allen Details darzustellen und an andere Programme weiterzugeben. Das Anhängen von achsenbezogenen Objekten funktionierte problemlos und erlaubte mir auch alle Informationen, die ich über den Standpunkt eines Elements zur Achse habe, zu implementieren. Darüber hinaus ist diese Entität die wichtigste Grundlage für weitere Entwicklungen im Bereich der Infrastruktur.

Die Visualisierung in Open Infra Platform konnte ich nicht umsetzen, da das Programm, die IFC4x1-Version nicht komplett unterstützt. Jedoch habe ich die Decks und die Achse in Constructivity visualisiert. Contructivity liest jedoch nicht mehr als einen Pfeiler ein. Die einzelnen eingelesenen Pfeiler enthalten alle wichtigen Informationen. In der Zukunft wird Open Infra Platform aber weiterentwickelt und es wird diese Version unterstützen.

Das in diesem Projekt entwickelte Brückenmodell stellt den groben Aufbau der Brücke in der Realität dar. Es war Ziel dieser Arbeit ein Brückenbauwerk zu modellieren, das so detailgetreu wie möglich ist, aber der Fokus der Arbeit sollte immer sein die Entwicklung des Datenmodells zu unterstützen.

In der Zukunft kann meine bestehende Modellierung als Grundlage dazu verwendet werden, ein Modell der Donnersbergerbrücke zu modellieren, das einen höheren Detaillierungsgrad besitzt und das Modell dann für zum Beispiel Wartungsarbeiten eingesetzt werden kann. Die fehlenden komplexen Decks, die Auflager und die Rampen können noch nachmodelliert werden.

Die modellierte Achse, die ich in meinem Projekt verwendet habe, kann auch in der Zukunft als Grundlage für eine maßstabsgetreue Achse der Donnersbergerbrücke verwendet werden.

Die Pläne, die mir zur Verfügung standen, habe keine genauere Modellierung möglich gemacht und der nächste Schritt für eine detailgetreuere Achse wäre die erneute Vermessung der Brücke. Die neu gewonnenen Informationen können dann für die Trassenmodellierung verwendet werden.

Mein Programm, welches die Generierung der finalen IFC4x1-Datei erst möglich machte, ist auf meinen Anwendungsbereich angepasst und funktioniert nur mit meinem Projekt fehlerfrei. Jegliche Änderung an den Ausgangsdateien könnte dazu führen, dass sich die Datei nicht mehr mit meinem Code erstellen lässt.

Die Programmierung eines allgemeinen Programms zur Transponierung von Brückenbauwerken zur neuesten Version hätte den Rahmen dieser Arbeit gesprengt.

Mein Code kann aber in der Zukunft als Grundlage verwendet werden, um ein Programm zu kreieren, dass alle bestehenden Brückenprojekte in das neue Datenformat umschreibt.

Die Resultate meiner Arbeit stellen eine gute Grundlage dafür dar, dass das „IfcAlignment project“ eine gute Basis für weiterführende Entwicklungen in diese Richtung ist.

6 Fazit

Für diese Arbeit habe ich ein Brückenmodell anhand der Donnersbergerbrücke nachgebildet. Die Fundamente und Stützen habe ich ohne Auflager und Gebäudetechnik in Autodesk Revit modelliert und als einzelne IFC2x3-Dateien exportiert. Die Achse der Brücke habe ich mit ProVI von der Firma OBERMEYER Planen + Beraten GmbH von den Plänen nachmodelliert und diese mit dem programmeigenen Export im IFC4x1-Datenmodell exportiert. Die Geometrie der Decks und Kappen, bestehend aus Polylinien mit Abhängigkeiten und einem Export, hat Herr Philipp Fengler mir zur Verfügung gestellt. Diese Geometrien habe ich mit den Parametern aus den Plänen befüllt und exportiert. Die exportierten Decks wurden von mir durch Hinzufügen der IFC4x1-Entitäten und dem Verknüpfen der Geometrie mit diesen Entitäten, zu einer visuell darstellbaren Datei umgewandelt.

All diese einzelnen IFC-Dateien habe ich durch meinen entwickelten Programmcode richtig nummeriert und die entsprechenden IFC4x1-Entitäten eingebunden. Jede Datei wurde mit dem IfcAlignment verknüpft, sodass die Positionierung an der Achse und der Abstand zur Achse den Abständen auf den Plänen entsprach.

Die abschließende Darstellung in Construtivity bezeugt, dass alle wichtigen Informationen der konstruierten Brücke in dieser Datei vorhanden sind.

Diese finale Datei belegt, dass es mit der neuen veröffentlichten IFC-Version möglich ist ein Brückenbauwerk ausreichend darzustellen und alle Informationen zu übermitteln, ohne dass es einer Nachbesserung an dem Datenmodell bedarf.

Somit habe ich eine gültige IFC4x1 Datei am Beispiel einer existierenden Brücke erstellt.

7 Verzeichnisse

7.1 Tabellenverzeichnis

7.2 Abbildungsverzeichnis

8 Literatur

[1] Autodesk GmbH, „www.autodesk.de," 07 05 2018. [Online]. Available: https://www.autodesk.de/solutions/bim/overview.

[2] buildingSMART e.V., „https://www.buildingsmart.de," 07 05 2018. [Online]. Available: https://www.buildingsmart.de/bim-knowhow/standards.

[3] buildingSMART e.V., „http://www.buildingsmart-tech.org," 06 06 2018. [Online]. Available: http://www.buildingsmart-tech.org/infrastructure/projects/alignment. [Zugriff am 06 06 2018].

[4] Š. Markič, „IFC-Bridge: Previous Initiatives and Their Proposals," 2017.

[5] buildingSMART e.V., „IFC Infra Overall Architecture Project / Documentation and Guidelines," 2017.

[6] „https://www.duden.de/," 2018. [Online]. Available: https://www.duden.de/rechtschreibung/Trasse. [Zugriff am 30 06 2018].

[7] U.-P. D.-I. S. Freudenstein, „Verkehrswegebau Grundkurs," München, 2016.

[8] U.-P. D.-I. S. Freudenstein, „Verkehrswegebau Grundkurs Übung," 2016.

[9] „http://www.doku-des-alltags.de/," 2014. [Online]. Available: http://www.doku-des-alltags.de/BDMuenchen/MuenchenDoBr/Mue%20Dobr%200.html. [Zugriff am 30 06 2018].

[10] M. Dörrbecker, „München - Donnersbergerbrücke (Panorama)," 2007.

[11] T. LIEBICH, „IFC5 Development plan.," 2016.

[12] P6b Project Team (Project Lead, Thomas Liebich), „IFC Alignment 1.1 project, IFC Schema extension proposal," 2017.

[13] buildingSMART e.V., „http://www.buildingsmart-tech.org," 2018. [Online]. Available: http://www.buildingsmart-tech.org/ifc/IFC4x1/final/html/.

[14] buildingSMART e.V., „http://www.buildingsmart-tech.org," 2018. [Online]. Available: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/schema/ifcproductextension/lexical/ifcrelcontainedinspatialstructure.htm. [Zugriff am 30 06 2018].

[15] A. Kuchling, „https://docs.python.org/," 29 06 2018. [Online]. Available: https://docs.python.org/3/howto/regex.html. [Zugriff am 29 06 2018].

[16] Planungsbüro Obermeyer, „Bestandsplan Lageplan,Längsschnitt 40/21 - 01," 1973.

[17] Planungsbüro Obermeyer, „Bestandsplan Gesamtübersicht 40/21 - 016," München, 1974.

[18] Planungsbüro Obermeyer, „Bestandsplan Gesamtübersicht 40/21 - 016 a," München, 1986.

[19] Planungsbüro Obermeyer, „Bestandsplan Lageplan,Längsschnitt, Südrampe 40/21 - 011," München, 1973.

[20] Planungsbüro Obermeyer, „Bestandsplan Lageplan Schnitte 1 bis 6 - 40/21/1-101," München, 1996.

[21] Planungsbüro Obermeyer, „Bestandsplan Regelquerschnitte SO- und SW-Rampe," München, 1986.

[22] P. Fengler, „Erstellung eines Brückenquerschnittprofils in IFC mithilfe eines Autodesk Inventor Modells," München, 2018.

[23] Planungsbüro Obermeyer, „Bestandsplan Pfeiler Rampenwiderlager Südost 40/21 - 13 a,“ München, 1986.

[24] Planungsbüro Obermeyer, „Bestandsplan Pfeiler 40/21 - 04,“ München, 1973.

[25] F. Mini, „Entwicklung eines LoD Konzepts für digitale Bauwerksmodelle von Brücken und dessen Implementierung,“ München, 2016.

[26] Planungsbüro Obermeyer, „Bestandsplan Pfeiler 9, 10, 11, 12 - 40/21 - 08,“ München, 1973.

[27] ESRI Inc., „ArcGIS Desktop,“ 03 06 2018. [Online]. Available: https://desktop.arcgis.com/de/arcmap/10.3/analyze/python/what-is-python-.htm.

9 Anhang A

Parametertabelle der Stützen:

Name	Dateityp	Pfeiler	Beinummer	Richtung	UK Pfeiler	OK Pfeiler	2.Stütze	Höhe Fundament	Höhe Stütze	Vertikale Position	Entfernung zur Achse
Fundament_1OW_2OW_Mass	.ifc	1		O	522	528.250		1.6	4.650	-7.780	14.625
Fundament_1OW_2OW_Mass_2	.ifc	1		W	522	528.300		1.6	4.700	-7.830	-1.375
Fundament_1OW_2OW_Mass_3	.ifc	2		O	521.7	529.090		1.6	5.790	-8.920	14.625
Fundament_1OW_2OW_Mass_4	.ifc	2		W	521.7	529.220		1.6	5.920	-9.050	-1.375
Fundament_3OW_Mass	.ifc	3		O	521	528.870		3	4.870	-9.400	14.625
Fundament_3OW_Mass_2	.ifc	3		W	521	528.980		3	4.980	-9.510	0
Fundament_4OW_Mass	.ifc	4		O	518.7	528.930		5	5.230	-11.760	14.625
Fundament_4OW_Mass	.ifc	4		W	518.7	528.960		5	5.260	-11.790	0
Fundament_5OW_Mass	.ifc	5		O	522	528.562		1.5	5.062	-8.092	14.625
Fundament_5OW_Mass	.ifc	5		W	522	528.169		1.5	4.669	-7.699	0
Fundament_6(1)O_Mass	.ifc	6	1	O	520.3	528.020		1.5	6.220	-9.250	14.625
Fundament_6(1)W_Mass	.ifc	6	1	W	520.8	528.136		1.5	5.836	-8.866	-7
Fundament_6(2)O_Mass	.ifc	6	2	O	520.3	528.140		1.5	6.340	-9.370	20
Fundament_6(2)W_Mass	.ifc	6	2	W	520.8	528.296		1.5	5.396	-9.026	0
Fundament_7(1)O_7(2)O_Mass	.ifc	7	1	O	520.1	527.650	527.752	2	5.550	-9.080	14.625
Fundament_7(1)W_Mass	.ifc	7	1	W	520.3	527.794		1.5	5.394	-9.024	-3
Fundament_7(2)W_Mass	.ifc	7	2	W	520.3	527.893		1.5	6.093	-9.123	3
Fundament_7(3)O_Mass	.ifc	7	3	O	520.1	525.700		1.5	4.100	-7.130	26.1
Fundament_8(1)O_8(2)O_Mass	.ifc	8	1	O	518.3	527.438	527.603	2	6.538	-10.068	14.625
Fundament_8(3)O_Mass	.ifc	8	3	O	516.4	526.839		2.5	7.939	-12.029	26.1
Fundament_8W_Mass	.ifc	8		W	520.1	527.613		1.5	6.013	-9.043	0
Pfeiler_9-15_Mass	.ifc	9			515.85	0.000			11.935	-13.465	13.25
Pfeiler_9-15_Mass	.ifc	9	1		520.05	0.000			11.935	-13.465	0
Pfeiler_9-15_Mass	.ifc	10			516	0.000			11.935	-13.465	13.25
Pfeiler_9-15_Mass	.ifc	10	1		520.5	0.000			11.935	-13.465	0
Pfeiler_9-15_Mass	.ifc	11			516	0.000			11.935	-13.465	13.25
Pfeiler_9-15_Mass	.ifc	11	1		520.5	0.000			11.935	-13.465	0
Pfeiler_9-15_Mass	.ifc	12			516	0.000			11.935	-13.465	13.25
Pfeiler_9-15_Mass	.ifc	12	1		520.5	0.000			11.935	-13.465	0

Abbildung 33: Excel-Datei für Parameter der Options-Datei

10 Anhang B

GUI:

```
from tkinter import *
from tkinter import messagebox
from re import split
from builtins import print
from tkinter import ttk
from tkinter import filedialog
import re
import io
import tkinter as tk
from MainIFC import ButtonAlignment
from MainIFC import ButtonColumn
from MainIFC import ButtonReunion
from MainIFC import ButtonDeck

#GUI
#erstellt das Fenster
root = Tk()

#benennt den Titel
root.title("IFC4x1-Generierung")

#Hintergrundfarbe des Fensters
root.config(background = "#FFFFFF")


#Hier kommen die Fensterelemente hin:

#initialisiert Frame
mainFrame = Frame(root, width = 500, height = 300)

#rel. Position des Frames

#mainFrame.grid(row = 3, column = 0, padx = 10, pady = 10)

#Button für Main IFC
IFCAlignment = Button(root, text="import IFC Alignment", command= ButtonAlignment)
IFCAlignment.grid(row = 0, column = 0, pady = 10, padx = 10)
IFCAlignment.pack

#Button für Add Column
IfcColumn = Button(root, text="import all Columns", command= ButtonColumn)
IfcColumn.grid(row = 1, column = 0, pady = 10, padx = 10)
IfcColumn.pack

#Button für Add Deck
IfcDeck = Button(root, text="import all Decks", command= ButtonDeck)
IfcDeck.grid(row = 2, column = 0, pady = 10, padx = 10)
IfcDeck.pack

#Button für Zusammenführung
IfcReunion = Button(root, text="Merge files", command= ButtonReunion)
IfcReunion.grid(row = 3, column = 0, pady = 10, padx = 10)
IfcReunion.pack
root.mainloop()
```

MainIFC:

```
from StringManipulation import stringmanipulation
from ImportColumns import importIfc
from ImportAlignment import importIFCAlignment
from Reunion import reunion
from ImportOptions import importOptions
from ImportDeck import  importIFCDeck

def ButtonAlignment():

    GloVars = importIFCAlignment()
    global g_highestIfcNumber
    global g_data
    global g_beginningProject
    global g_endingProject
    global g_numbers
    global g_typeIFC
    global g_parametersIFC

    print ("Import des Alignments abgeschlossen")

    g_beginningProject = GloVars[0]
    g_data = GloVars[1]
    g_endingProject = GloVars[2]
    g_highestIfcNumber = GloVars[3]
    g_numbers = GloVars[4]
    g_typeIFC = GloVars[5]
    g_parametersIFC = GloVars[6]

def ButtonColumn():

    global g_highestIfcNumber
    global g_data
    global g_beginningProject
    global g_endingProject
    global g_numbers
    global g_typeIFC
    global g_parametersIFC


    GloVarsOptions = importOptions()

    ColFilename = GloVarsOptions[0]
    ColPositioning = GloVarsOptions[1]
    ColOrientation = GloVarsOptions[2]
    ColYCoordinate = GloVarsOptions[3]
    ColZCoordinate = GloVarsOptions[4]
    rawPath = GloVarsOptions[5]
    pattern = GloVarsOptions[6]
    height = GloVarsOptions[7]
    counter = 1
    VarsCounter = 0
    for x in ColFilename:
        path = rawPath + x
        GloVarsCol = importIfc(g_highestIfcNumber, g_data, g_beginningProject,
g_endingProject, ColPositioning, ColOrientation, ColYCoordinate, ColZCoordinate, path,
VarsCounter, pattern, height)

        print ("Import der %s. Stütze abgeschlossen." %counter)
        counter = counter + 1
        VarsCounter = VarsCounter + 1
        g_highestIfcNumber = GloVarsCol[0]
```

```
        g_numbers = GloVarsCol[1]
        g_typeIFC = GloVarsCol[2]
        g_parametersIFC = GloVarsCol[3]
        g_data = GloVarsCol[4]
    print ("Import aller Stützen abgeschlossen.")

def ButtonDeck():
    global g_numbers
    global g_typeIFC
    global g_parametersIFC
    global g_highestIfcNumber

    GloVarsDeck = importIFCDeck(g_numbers, g_typeIFC, g_parametersIFC, g_highe-
stIfcNumber)

    g_numbers = GloVarsDeck[0]
    g_typeIFC = GloVarsDeck[1]
    g_parametersIFC = GloVarsDeck[2]

    print('Alle Decks wurden hinzugefügt')




def ButtonReunion():
    reunion(g_numbers, g_typeIFC, g_parametersIFC, g_beginningProject, g_endingPro-
ject)
    print('Die finale IFC-Datei wurde erstellt und abgespeichert.')
```

ImportAlignment:

```
from tkinter import filedialog
from StringManipulation import stringmanipulation

def importIFCAlignment():
    #Einlesen der ersten IFC Datei (mind. zwei Dateitypen benötigt)
    IFC = filedialog.askopenfilename(filetypes=(('IFC-files','.ifc'),("Text File",
"*.txt")))

    #Ausgabe der IFC Datei zur Überprüfung (später nicht mehr gebraucht)

    openedIFC = open(IFC, "r")
    GloVars = stringmanipulation(openedIFC)

    return GloVars
```

Stringmanipulation:

```
def stringmanipulation(openedIFC):

    #liest die Zeilen der IFC-Datei ein
    content = openedIFC.readlines()

    #Ersetzt das neue Zeile Zeichen
    content = [x.strip() for x in content]

    beginningProject = [content[x] for x in range(0 , content.index("DATA;") + 1)]


    beginningIfc = content.index("DATA;")

    data = [content[x] for x in range(beginningIfc , len(content))]
    endIFC = data.index("ENDSEC;")
    endingProject = [data[x] for x in range(endIFC , len(data))]
    data = [data[x] for x in range(1 , endIFC)]

    #ersetzt ; in der IFC-Datei zu ""
    data = [c.replace(';',"") for c in data]
    numbers = []
    typeparameters = []
    typeIFC = []
    parametersIFC = []

    #teilt die IFC-Daten in einzelne Listen auf
    for z in range(0, len(data)):
        splitcontent = data[z]

        #trennt Nummern von Typen + Parametern
        splitcontent_new = splitcontent.split("= ")
        typeparameters = []

        #hängt Nummern an die Nummernliste an
        numbers.append(splitcontent_new[0])
        typeparameters.append(splitcontent_new[1])

        #trennt Typen von den Parametern
        typeparameters_edited = typeparameters[0].split("(",1)

        #hängt Typen an die Typenliste an
        typeIFC.append(typeparameters_edited[0])

        #hängt Parameter an die Parameterliste an
        parametersIFC.append("(" + typeparameters_edited[1])

    highestIfcString = numbers[len(numbers) - 1 ].replace("#", "", 1 )
    highestIfcNumber = int(highestIfcString)

    length_numbers = highestIfcNumber
    print ("#%s ist der höchste Eintrag in der Nummernliste" %length_numbers)

    if numbers is "":
        print("keine Einträge in der Nummernliste")

    else:

        global g_highestIfcNumber
        global g_data
        global g_beginningProject
        global g_endingProject
```

```
        global g_numbers
        global g_typeIFC
        global g_parametersIFC

        g_highestIfcNumber = highestIfcNumber
        g_data = data
        g_beginningProject = beginningProject
        g_endingProject = endingProject


        return (g_beginningProject, g_data, g_endingProject, g_highestIfcNumber, num-
bers, typeIFC, parametersIFC)
```

ImportOptions

```
from tkinter import filedialog

def importOptions():
    filename = []
    positioning = []
    orientation = []
    yCoordinate = []
    zCoordinate = []
    heightColumn = []
    pattern = []

    #Öffnet die Options-Datei der Columns für sich im Hintergrund, um darauf zugreifen
zu können
    print("Bitte wählen Sie die Options Datei aus...")

    path = filedialog.askopenfilename(filetypes=(("Text File", "*.txt"),('IFC-fi-
les','.ifc')))

    rawPath = 'C:/Users/chris/Desktop/PräsentationCode/'
    openedOptions = open(path, "r")

    print("Options Datei ausgewählt.")
    #liest die Zeilen der Option-Datei ein
    content = openedOptions.readlines()

    #Ersetzt das neue Zeile Zeichen
    content = [x.strip() for x in content]
    content = [c.replace('[', '').replace(']', '') for c in content]

    for z in range(0, len(content)):
        splitcontent = content[z]

        splitcontent_new = splitcontent.split("#")

        filename.append(splitcontent_new[0])
        positioning.append(splitcontent_new[1])
        orientation.append(splitcontent_new[2])
        yCoordinate.append(splitcontent_new[3])
        zCoordinate.append(splitcontent_new[4])
        heightColumn.append(splitcontent_new[5])
        pattern.append(splitcontent_new[6])



    return (filename, positioning, orientation, yCoordinate, zCoordinate, rawPath,
pattern, heightColumn)
```

ImportColumns:

```
from tkinter import *
from tkinter import messagebox
from re import split
from builtins import print
from tkinter import ttk
from tkinter import filedialog
import re
import io
import tkinter as tk
from SearchIndices import searchIndices
from RegularExpression import regularExpression
from StringManipulationColumns import StringManipulationColumns
from IFC4x1_LinearPlacementAdder import IFC4x1_LinearPlacementAdder
from IFC4x1_DistanceOrientationAdder import IFC4x1_DistanceOrientationAdder
from ColumnHeightAdder import IFC_ColumnHeightAdder

def importIfc(highestIfcNumber, data_v2, beginningProject, endingProject, ColPosition-
ing, ColOrientation, ColYCoordinate, ColZCoordinate, path, VarsCounter, pattern,
height):
        ExpressionData = []
        data_v3  = []

        #Liest die Zeilen der IFC-Datei ein
        openedIFC = open(path, "r")
        text = openedIFC.readlines()

        #Aufrufen der SearchIndices Funktion zur Bestimmung aller miteinander ver-
knüpften IFC Dateien
        IndexList = searchIndices(text,pattern[VarsCounter])

        IndexListWoDE = IndexList
        IndexListWoDE.sort()


        #Indizes werden wieder zu Text zusammengeführt
        content_column = []
        for x in IndexListWoDE:
            content_column.append(text[x])


        #Ersetzt ; in der IFC-Datei zu ""
        content_column = [c.replace(';',"") for c in content_column]

        #Ersetzt das neue Zeile Zeichen
        data = [x.strip() for x in content_column]

        if pattern[VarsCounter] == "IFCCOLUMN":
            print("hier")
            data = IFC_ColumnHeightAdder(data, VarsCounter, height)


        start = 0
        print("Nummerierung wird angepasst...")
        for z in data:

            #Übergabe in die regularExpression Funktion
            changedString = regularExpression(z, highestIfcNumber)
            ExpressionData.append(changedString)
            start = start + 1

        IFC4x1_data = IFC4x1_LinearPlacementAdder(ExpressionData)
```

```
        for z in IFC4x1_data:
            data_v2.append(z)



        GloVars = StringManipulationColumns(data_v2)
        numbers = GloVars[1]
        typeIFC = GloVars[2]
        parametersIFC = GloVars[3]

        for z in range(0,len(numbers)):

            data_v3.append(numbers[z]+ "= " + typeIFC[z] + parametersIFC[z] + ";")

        #aufrufen der Distance und Orientation Funktion zum Hinzufügen zu IFCLINEAR-
PLACEMENT
        ExpressionVars = IFC4x1_DistanceOrientationAdder(data_v3, ColPositioning, Co-
lOrientation, ColYCoordinate, ColZCoordinate, VarsCounter, GloVars[0], GloVars[1],
GloVars[2], GloVars[3])
        data_v2 = []
        numbers = ExpressionVars[0]
        typeIFC = ExpressionVars[1]
        parametersIFC = ExpressionVars[2]

        for z in range(0,len(numbers)):

            data_v2.append(numbers[z]+ "= " + typeIFC[z] + parametersIFC[z])

        GloVars = list(GloVars)
        GloVars[0] = GloVars[0] + 3
        GloVars = tuple(GloVars)


        return (GloVars[0], ExpressionVars[0], ExpressionVars[1], ExpressionVars[2],
data_v2)
```

StringmanipulationColumns:

```
def StringManipulationColumns(data_v2):

    numbers = []
    typeparameters = []
    typeIFC = []
    parametersIFC = []

    #teilt die IFC-Daten in einzelne Listen auf
    for z in range(0, len(data_v2)):
        splitcontent = data_v2[z]

        #trennt Nummern von Typen + Parametern
        splitcontent_new = splitcontent.split("= ")
        typeparameters = []

        #hängt Nummern an die Nummernliste an
        numbers.append(splitcontent_new[0])
        typeparameters.append(splitcontent_new[1])

        #trennt Typen von den Parametern
        typeparameters_edited = typeparameters[0].split("(",1)


        #hängt Typen an die Typenliste an
        typeIFC.append(typeparameters_edited[0])

        parametersIFC.append("(" + typeparameters_edited[1])

    highestIfcString = numbers[len(numbers) - 1 ].replace("#", "", 1 )
    highestIfcNumber = int(highestIfcString)


    return(highestIfcNumber, numbers, typeIFC, parametersIFC)
```

SearchIndices:

```
import re

def searchIndices_v2(indices, text):

    for x in indices:
        matchNumWH = re.findall('#[0-9]+',text[x])
        linkedNumbers  = []

        for y in range(1,len(matchNumWH)):
            linkedNumbers.append(matchNumWH[y])
        indices_v2 = []

        for y in linkedNumbers:

            for z in text:
                addedIS = y + "="
                match = re.findall(addedIS, z)


                if match == []:
                    pass
                else:
```

```
                    indices_v2.append(text.index(z))

        if indices_v2 == []:
            pass

        else:

            for x in indices_v2:
                if x not in IndexList:

                    IndexList.append(x)

            #wiederaufrufen der Funktion um zu allen unterverlinkten Zeilen zu kommen
            searchIndices_v2(indices_v2, text)

    return IndexList

def searchIndices_v1(text,pattern):
    indices = []
    IndexList = []
    for x in text:


        #Suche nach dem Wort "IFCBUILDINGELEMENTPROXY" bzw. "IFCCOLUMN"
        buildingElementProxy = re.findall(pattern, x)

        #Falls kein ElementProxy gefunden wird, weiter ohne Aktion
        if not buildingElementProxy:
            pass

        #Wenn ein ElementProxy gefunden wird
        else:

            #Finden der verlinkten Parameter in jeder einzelnen Zeile der IFC
            indexBEP = text.index(x)
            matchNumWH = re.findall('#[0-9]+',x)

            linkedNumbers  = []
            linkedNumbersBEP = []

            #Abspeichern der Zeilennummer von Building Element Proxy
            numberIfcBuElPr = matchNumWH[0]


            #Text Zeile für Zeile wieder durchgehen
            for y in text:
                pattern = (r"%s\b" %numberIfcBuElPr)

                #Zeilen nach der BEP-Nummer durchsuchen
                numbersWithBEP = re.findall(pattern,y)

                #Wenn nichts gefunden wird, weiter ohne Aktion
                if not numbersWithBEP:
                    pass

                #Wenn die Nummer gefunden wird:
                else:
                    #Finde alle Nummern in dieser Zeile der Daten
                    matchnumBEP = re.findall('#[0-9]+',y)

                    #Anhängen der Zeilennummern an eine Liste um alle Zeilennummern zu
speichern in denen BEP vorkommt
                    linkedNumbersBEP.append(matchnumBEP[0])
```

```
            #Nun wird mit jeder Zahl, die in der Liste für Zeilennummern mit BEP ist,
einzeln der Text durchgegangen
            for y in linkedNumbersBEP:
                for z in text:

                    #Ein "=" angehängt, damit sie eindeutig zu identifizieren sind
                    addedIS = str(y) + "="

                    #Finden der Indizes der verlinkten Parametern in denen BEP vor-
kommt
                    match = re.findall(addedIS, z)

                    #Solange nicht gefunden wird, weiter ohne Aktion
                    if match == []:
                        pass

                    #Wenn etwas gefunden wird:
                    else:

                        #Suchen ob es sich um ein IFCRELCONTAINEDINSPATIALSTRUCTURE
handelt
                        relcon = re.findall("IFCRELCONTAINEDINSPATIALSTRUCTURE", z)

                        #Wenn nein, dann hier:
                        if relcon == []:
                            indexPar = text.index(z)

                            #Überprüfung ob der Index schon in der Indexliste vorkommt
                            if indexPar not in IndexList:
                                indices.append(indexPar)
                                IndexList.append(indexPar)

                        #Wenn ja -> weiter ohne Aktion
                        else:
                            pass


    #Aufrufen der SearchIndices Funktion zur Bestimmung aller miteinander verknüpften
IFC Dateien


    IndexList = searchIndices_v2(indices, text)
    for x in indices:
        if x not in IndexList:
            IndexList.append(x)


    IndexListWoDE = IndexList
    IndexListWoDE.sort()


    return IndexList

def searchIndices(text, pattern):
    print("Relevate Einträge aus dem IFC-File werden identifiziert...")
    global IndexList
    IndexList = []
    IndexList = searchIndices_v1(text,pattern)


    return IndexList
```

RegularExpression:

```
import re

def regularExpression(string, number):

    #findet alle # mit einer Zahl dahinter und "merkt" sie sich
    matchObj = re.findall('#[0-9]+',string)


    matchObjoH = []

    #aus allen Ergebnissen von oben zieht er nun alle Zahlen raus
    for z in matchObj:
        matchObjoH_new = re.findall('[0-9]+',z)

        #diese Zahlen werden hier in eine eigene Liste abgespeichert
        for x in matchObjoH_new:
            matchObjoH.append(x)

    #hier in Integer umformatiert
    matchObjoH = list(map(int, matchObjoH))

    #absteigend sortiert
    matchObjoH.sort(reverse=True)
    matchObj_old = []

    #die Zahlen vor der Erhöhung nun wieder in einer Liste gesichert, damit mit der
aktuellen die Berechnung durchgeführt werden kann
    for z in matchObjoH:
        xy = '#' + str(z)
        matchObj_old.append(xy)



    #Erhöhung der Zahlen umd die höchste Zahl der vorhergehenden IFC-Datei
    xx= 0
    for z in matchObjoH:

        z_new = z + number

        #Umwandlung wieder in Zeilennummern mit #
        matchObjoH[xx] = '#'+ str(z_new)
        xx = xx+1



    yy = 0
    for z in matchObj_old:

        pattern =  z + "\b"

        #Ersetzen der alten Zeilennummern durch die neuen Zeilennummern
        string = re.sub(r'%s\b' %z,matchObjoH[yy], string, 1)

        yy = yy + 1

    return string
```

IFC4x1_LinearPlacementAdder:

```
import re

def IFC4x1_LinearPlacementAdder(data):
    listPlacements = []
    counter = 0
    for x in data:

        localPlacement = re.findall(r'IFCLOCALPLACEMENT\b', x)



        if localPlacement:

            #findet alle # mit einer Zahl dahinter und "merkt" sie sich
            matchObj = re.findall('#[0-9]+',x)

            listPlacements.append(matchObj[0])

            if counter is 0:
                pattern = x

                counter= counter + 1
                continue

            else:
                #entfernt alle LocalPlacements bis auf das erste
                data.remove(x)

    for x in data:
        index = data.index(x)
        if x is pattern:

            #ersetzen vom ersten LocalPlacement zu LinearPlacement mit Platzhaltern
            index = data.index(x)

            data[index] = str(listPlacements[0]) + "= IFCLINEARPLACEMENT(#40" + "," +
" INSERTDISTANCEEXPRESSION" + "," + " INSERTORIENTATIONEXPRESSION" + ",$)"
            continue

        for y in listPlacements:
            matchedLocPlaIndex = re.findall((y), x)

            if matchedLocPlaIndex:

                data[index] = re.sub((y), listPlacements[0], data[index])


    return data
```

IFC4x1_DistanceOrientationAdder:

```
import re

def IFC4x1_DistanceOrientationAdder(data, positioning, orientation, lateral, vertical,
VarsCounter, highestIfcNumber, numbers, typeIFC, parametersIFC):
    distanceNumber = highestIfcNumber + 1
    orientationNumber = highestIfcNumber + 2
    directionNumber = highestIfcNumber + 3

    #Für die DistanceExpression
    data.append("#" + str(distanceNumber) + "= IFCDISTANCEEXPRESSION(" + str(position-
ing[VarsCounter]) + ", " + str(lateral[VarsCounter]) + ", " + str(vertical[VarsCoun-
ter]) + ", $, $)")
    data.append("#" + str(orientationNumber) + "= IFCORIENTATIONEXPRESSION(#" +
str(directionNumber) + ", $)")
    data.append("#" + str(directionNumber) + "= IFCDIRECTION(" + str(orienta-
tion[VarsCounter]) + ")")
    numbers.append("#" + str(distanceNumber))
    numbers.append("#" + str(orientationNumber))
    numbers.append("#" + str(directionNumber))
    typeIFC.append("IFCDISTANCEEXPRESSION")
    typeIFC.append("IFCORIENTATIONEXPRESSION")
    typeIFC.append("IFCDIRECTION")
    parametersIFC.append("(" + str(positioning[VarsCounter]) + ", " + str(late-
ral[VarsCounter]) + ", " + str(vertical[VarsCounter]) + ", $, $)")
    parametersIFC.append("(#" + str(directionNumber) + ", $)")
    parametersIFC.append("(" + str(orientation[VarsCounter]) + ")")

    for x in parametersIFC:

        Distance = re.findall(r' INSERTDISTANCEEXPRESSION\b', x)

        if Distance:

            index = parametersIFC.index(x)
            subDis = re.sub((r' INSERTDISTANCEEXPRESSION'), "#" + str(distanceNumber),
parametersIFC[index])

            subDisOri = re.sub((r' INSERTORIENTATIONEXPRESSION'), "#" + str(orientati-
onNumber), subDis)

            parametersIFC[index] = subDisOri
            continue

    return numbers, typeIFC, parametersIFC
```

ImportIFCDeck:

```
from tkinter import filedialog
from StringManipulationDeck import stringmanipulationDeck
from RegularExpression import regularExpression



def importIFCDeck(numbers, typeIFC, parametersIFC, highestIFCNumber):

    path = "D:/Programme/Google Drive/Studium/Bachelorarbeit/Donnersbergerbruecke/Inventor/TEST/"
    ListOfDeckNames = ["RQ_RWLSued-Pfeiler3_rechts_mitKappen.txt", "RQ_RWLSued-Pfeiler3_links_mitKappen.txt", "RQ_Pfeiler9-Pfeiler18_rechts.txt", "RQ_Pfeiler9-Pfeiler18_links.txt", "RQ_Bereich_Pfeiler6_rechts.txt", "RQ_Bereich_Pfeiler6_links.txt", "RQ_Bereich_Pfeiler3-Pfeiler6_rechts.txt", "RQ_Bereich_Pfeiler3-Pfeiler6_links.txt" ]

    for x in ListOfDeckNames:
        ExpressionData = []
        itemPath = path + x
        openedIFC = open(itemPath, "r")
        #liest die Zeilen der IFC-Datei ein
        content = openedIFC.readlines()

        #Ersetzt das neue Zeile Zeichen
        content = [x.strip() for x in content]

        #ersetzt ; in der IFC-Datei zu ""
        content = [c.replace(';',"") for c in content]

        data = [content[x] for x in range(0 , len(content))]

        for y in data:
            changedString = regularExpression(y, highestIFCNumber)
            ExpressionData.append(changedString)


        GloVars = stringmanipulationDeck(ExpressionData, numbers, typeIFC, parametersIFC)
        numbers = GloVars [0]

        typeIFC = GloVars [1]
        parametersIFC = GloVars [2]
        highestIFCNumber = GloVars [3]


    return (numbers, typeIFC, parametersIFC)
```

StringmanipulationDeck:

```
import re
def stringmanipulationDeck(data, numbers, typeIFC, parametersIFC):

    #teilt die IFC-Daten in einzelne Listen auf
    for z in range(0, len(data)):
        splitcontent = data[z]

        #trennt Nummern von Typen + Parametern
        splitcontent_new = splitcontent.split("=")

        typeparameters = []



        #hängt Nummern an die Nummernliste an
        numbers.append(splitcontent_new[0])
        typeparameters.append(splitcontent_new[1])

        #trennt Typen von den Parametern
        typeparameters_edited = typeparameters[0].split("(",1)

        #hängt Typen an die Typenliste an
        typeIFC.append(typeparameters_edited[0])

        #hängt Parameter an die Parameterliste an
        parametersIFC.append("(" + typeparameters_edited[1])

        highestIFCString = numbers[len(numbers) - 1 ].replace("#", "", 1 )
        highestIFCNumber = int(highestIFCString)

        for x in parametersIFC:

            Distance = re.findall(r'HIERKOMMTDIEALIGNMENTCURVE\b', x)

            if Distance:
                index = parametersIFC.index(x)
                subDis = re.sub((r'HIERKOMMTDIEALIGNMENTCURVE'), "#40", parameter-
sIFC[index])


                parametersIFC[index] = subDis



    return (numbers, typeIFC, parametersIFC, highestIFCNumber,)
```

Reunion:

```
from RelConAdder import RelContAdder
from tkinter import filedialog

def reunion(numbers, typeIFC, parametersIFC, beginningProject, endingProject):
    finalIfcData = []

    #Hängt den Anfang von IFC Alignment an den Anfang der finalen IFC Datei
    for z in range(0, len(beginningProject)):
        finalIfcData.append(beginningProject[z])

    #Hängt der erstellten Mittelteil bestehend aus allen hinzugefügten IFC-Dateien an
die finale Datei an
    for z in range(0,len(numbers)):
        finalIfcData.append(numbers[z]+ "= " + typeIFC[z] + parametersIFC[z] + ";")

    #Hängt das Ende von IFC Alignment an das Ende der finalen IFC Datei
    for z in range(0, len(endingProject)):
        finalIfcData.append(endingProject[z])



    #Aufrufen der RelConAdder Funktion, um die BuildingElemProxy Nummern anzuhängen
    finalIfcData = RelContAdder(finalIfcData)


    final_textwrapper= filedialog.asksaveasfile(mode='w', defaultextension=".ifc")
    final = final_textwrapper.name
    writeFinalIfc = open(final, 'w')
    for item in finalIfcData:
        writeFinalIfc.write("%s\n" % item)
```

RelConAdder:

```
import re

def RelContAdder (finalIfcData):
    RelCon_List = []
    for x in finalIfcData:
        #Finden der Zeilennummer von IFCRELCONTAINEDINSPATIALSTRUCTURE
        relConInSpa = re.findall("IFCRELCONTAINEDINSPATIALSTRUCTURE", x)

        #Finden der Zeilennummer von IFCBUILDINGELEMENTPROXY
        buildingElementProxy = re.findall(r'IFCBUILDINGELEMENTPROXY\b', x)
        Column = re.findall(r'IFCCOLUMN\b', x)
        slab =  re.findall(r'IFCSLAB\b', x)
        #Solange nichts gefunden wird "pass"
        if not relConInSpa:
                pass
        #Index abspeichern von RelConInSpaStr, zum späteren befüllen
        else:
            index_relCon = finalIfcData.index(x)
            entry_relcon = re.findall(r'(\,\s\(\#[0-9]+)', x)
            number = re.findall(r'#[0-9]+', entry_relcon[0])
            RelCon_List.append(number[0])


        if not buildingElementProxy:
                pass
```

```
        else:
            matchNumWH = re.findall('#[0-9]+',x)

            #Abspeichern der gefundenden BuildingElementProxys
            if matchNumWH[0] is not RelCon_List:
                RelCon_List.append(matchNumWH[0])

        if not Column:
            pass
        else:
            matchNumWH = re.findall('#[0-9]+',x)

            #Abspeichern der gefundenden Columns
            if matchNumWH[0] is not RelCon_List:
                RelCon_List.append(matchNumWH[0])

        if not slab:
                pass
        else:
            matchNumWH = re.findall('#[0-9]+',x)

            #Abspeichern der gefundenden Slabs
            if matchNumWH[0] is not RelCon_List:
                RelCon_List.append(matchNumWH[0])

    counter = 0
    string = ",("

    for x in RelCon_List:
        if counter == 0:
            string = string + x
            counter = counter +1
        else:

            string = string + ", " + x

    #Ersetzen der Parameter von RelCon mit dem oben generierten String aus allen Buil-
dingElementProxys
    finalIfcData[index_relCon] = re.sub(r'\, \(#[0-9]+', string, finalIfcData[in-
dex_relCon])

    return finalIfcData
```

11 Anhang C

Externer Anhang

Auf der beigefügten USB-Stick befindet sich folgender Inhalt:

- Schriftliche Ausarbeitung im PDF-Format
- Programmcode
- Bestandspläne der Brücke
- IFC-Datei für die Visualisierung

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-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, 15. Juni 2018

Vorname Nachname