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

Christoph Raidl

Bachelorthesis

für den Bachelor of Science Studiengang Bauingenieurwesen

Autor: Christoph Raidl

Matrikelnummer: 

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

Ausgabedatum: 15. Januar 2018

Abgabedatum: 15. Juni 2018
Inhaltsverzeichnis

Zusammenfassung ................................................................................................... 3
Abstract .................................................................................................................... 3
Danksagung ............................................................................................................. 4

1 Einführung 5

2 Grundlagen 8
2.1 Trasse ................................................................. 8
2.1.1 Achse ..................................................................... 8
2.1.2 Gradiente ........................................................... 10
2.2 Brücken ................................................................. 11
2.3 IFC4x1 ..................................................................... 12
2.3.1 IfcAlignment ....................................................... 12
2.3.2 IfcLinearPlacement .............................................. 13
2.3.3 IfcSectionedSolidHorizontal .................................. 14
2.3.4 IfcRelContainedInSpatialStructure ......................... 17
2.4 Programmierung ..................................................... 17
2.4.1 Regular Expression ................................................ 18

3 Methodik 19
3.1 Prozess ................................................................. 19
3.2 Modellierung von Achse und Gradiente ......................................................... 20
3.3 Modellierung von Deck und Kappen ............................................................... 23
3.4 Modellierung der Stützen ................................................................................ 28

4 Fusion 33
4.1 Python ...................................................................... 33
4.2 Skripten ..................................................................... 33

5 Diskussion 40

6 Fazit 42
<table>
<thead>
<tr>
<th></th>
<th>Verzeichnisse</th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Tabellenverzeichnis</td>
<td>43</td>
</tr>
<tr>
<td>7.2</td>
<td>Abbildungsverzeichnis</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>Literatur</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>Anhang A</td>
<td>48</td>
</tr>
<tr>
<td>10</td>
<td>Anhang B</td>
<td>49</td>
</tr>
<tr>
<td>11</td>
<td>Anhang C</td>
<td>66</td>
</tr>
</tbody>
</table>
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

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:


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 Referenzier rung 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:
  - Zeichenkettdarstellung
  - Querschnittsdarstellung
  - Oberflächendarstellung
  - Festkörperdarstellung
  - Gelände

Um mein Thema bearbeiten zu können musste ich mich als Erstes dafür entscheiden, welche Brücke ich nachmodellieren möchte. Ich entschied mich für die Donnersbergbrücke, 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 selbst-entwickeltes 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 Kilometerierungswerten oder Parametern beschriftet. Kilometerierungen beginnen immer am Anfang der Achse und enden am Ende und werden durch einen Beschriftungspfeil mit der genauen Kilometerierung markiert. Dazwischen werden die Kilometerierungswerte 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]


2.1.2 Gradiente


Abbildung 3: Gradiente mit Beschriftung [7]
2.2 Brücken

„Brücken sind Überführungen eines Verkehrsweges über einen anderen Verkehrswege, 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.

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]


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

2.3.1 IfcAlignment


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.

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.


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.
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. IfcOffsetCurveByDistance 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.
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 IfcRelContainedInSpatialStructure 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 gehört, weil sie zwei Ebenen miteinander verbindet. [14]

2.4 Programmierung

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.


Nun gehe ich näher auf die oben genannten Schritte ein:
3.2 Modellierung von 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.

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]*

<table>
<thead>
<tr>
<th>Station</th>
<th>Elementtyp</th>
<th>Radius</th>
<th>Klothoidenparameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1053.164</td>
<td>Gerade</td>
<td>0 m</td>
<td></td>
</tr>
<tr>
<td>1238.382</td>
<td>Bogen rechts</td>
<td>1500 m</td>
<td></td>
</tr>
<tr>
<td>1455.054</td>
<td>Gerade</td>
<td>0 m</td>
<td></td>
</tr>
<tr>
<td>1774.665</td>
<td>Übergangsbo  gen</td>
<td>120 m</td>
<td></td>
</tr>
<tr>
<td>1818.55</td>
<td>Bogen rechts</td>
<td>325 m</td>
<td></td>
</tr>
<tr>
<td>1922.585</td>
<td>Übergangsbo  gen</td>
<td>120 m</td>
<td></td>
</tr>
<tr>
<td>2013.666</td>
<td>Bogen links</td>
<td>325 m</td>
<td></td>
</tr>
<tr>
<td>2138.158</td>
<td>Übergangsbo  gen</td>
<td>110 m</td>
<td></td>
</tr>
<tr>
<td>2175.675</td>
<td>Gerade</td>
<td>0 m</td>
<td></td>
</tr>
</tbody>
</table>
In Abbildung 13 ist der Ausschnitt der Dialogbox von ProVI abgebildet.

<table>
<thead>
<tr>
<th>Station</th>
<th>Typ</th>
<th>V</th>
<th>Radius</th>
<th>Gesamtlänge</th>
<th>Eingabeel.</th>
<th>Eingabeel.</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1053.164</td>
<td>Fest</td>
<td>40</td>
<td>0.0</td>
<td>185.218</td>
<td>0.0</td>
<td>185.218</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1238.382</td>
<td>Koppel</td>
<td>40</td>
<td>150.0</td>
<td>216.672</td>
<td>216.672</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1455.054</td>
<td>Koppel</td>
<td>40</td>
<td>0.0</td>
<td>319.611</td>
<td>319.611</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1774.085</td>
<td>Übergang</td>
<td>40</td>
<td>0.0</td>
<td>43.885</td>
<td>43.885</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1875.55</td>
<td>Koppel</td>
<td>40</td>
<td>325.0</td>
<td>104.035</td>
<td>104.035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2022.555</td>
<td>Übergang</td>
<td>40</td>
<td>0.0</td>
<td>51.861</td>
<td>43.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2031.666</td>
<td>Koppel</td>
<td>40</td>
<td>-325.0</td>
<td>135.457</td>
<td>124.452</td>
<td>124.452</td>
<td>47.201</td>
<td></td>
</tr>
<tr>
<td>2138.158</td>
<td>Übergang</td>
<td>40</td>
<td>0.0</td>
<td>37.517</td>
<td>37.517</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2175.875</td>
<td>Koppel</td>
<td>40</td>
<td>0.0</td>
<td>50.0</td>
<td>50.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2225.075</td>
<td>Koppel</td>
<td>40</td>
<td>0.0</td>
<td>50.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.


3.3 Modellierung von 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.

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.


Abbildung 17: Profilansicht eines Decks in Inventor [22]

Abbildung 18: Parameter des ursprünglichen Profils [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 ent-hält (wie bereits in 2.3.3 IfcSectionedSolidHorizontal erwähnt), habe ich die IFC-Da-teien der Decks noch, durch Hinzufügen von den in Abbildung 21 gezeigten Entitäten und das richtige Verknüpfen der Parameter, händisch in Objekte mit semantischer Be-deutung verwandelt.

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 ver-bindet 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.

Abbildung 22: IFC-Datei eines Decks mit IFC4x1-Entitäten
3.4 Modellierung der Stützen


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 ungefährrer 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 „Loften“ 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.


*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.

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


![Abbildung 26: Rohform aus Polylinien](image1)

![Abbildung 27: Gefüllter Körper nach Loft](image2)

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](image3)

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

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 31 zeigt einen Ausschnitt aus dem Hauptteil 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.

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

ImportColumns.py (siehe Anhang B):


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):**


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):**


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):`


**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-Datenmodellversion 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.


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 Detailierungsgrad 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


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

Tabelle 1: Parameter der finalen Alignmentdatei .................................................. 21
Tabelle 2: Modellierte Regelquerschnitte .............................................................. 24

7.2 Abbildungsverzeichnis

Abbildung 1: Achse aus Radien und Klothoiden mit Kilometrierung (Freudenstein, Verkehrswegebau Grundkurs 2016) ................................................................................................................................. 9
Abbildung 2: Skizze: Positionierung achsbaasierte Objekte (Freudenstein, Verkehrswegebau Grundkurs 2016) ................................................................. 9
Abbildung 3: Gradienten mit Beschriftung (Freudenstein, Verkehrswegebau Grundkurs 2016) ................................................................. 10
Abbildung 4: Blick über die Donnersbergerbrücke (Dörrbecker 2007) ................................................................. 11
Abbildung 5: Geplanter Weg zu IFC5 (Liebich 2016) ................................................................. 12
Abbildung 6: Verknüpfung von LinearPlacement mit IfcAlignmentCurve (BuildingSMART e. V. 2018) ................................................................. 13
Abbildung 7: IfcSectionedSolidHorizontal und IfcArbitraryClosedProfileDef (BuildingSMART e. V. 2017) ................................................................. 15
Abbildung 8: Klassendiagramm IfcSectionedSolidHorizontal (P6B Project Team (Project Lead, Thomas Liebich) 2017) ................................................................. 16
Abbildung 9: IfcRelContainedInSpatialStructure (BuildingSMART e. V. 2018)......................................................................................................................... 17
Abbildung 10: Prozess Nummerierungsänderung mit Regular Expression ................................................................. 18
Abbildung 11: Prozess Modellierung der Achse und Gradienten ................................................................. 20
Abbildung 12: Ausschnitt der Ansicht in ProVi ...................................................................................... 20
Abbildung 13: ProVI Dialog........................................................................................................ 22
Abbildung 14: Prozess Modellierung Deck und Kappen .......................................................................... 23
Abbildung 15: Regelquerschnitt Stahlüberbau (Planungsbüro Obermeyer 1996) ................................................................. 24
Abbildung 16: Ersatzregelquerschnitt für den Stahlüberbau (Planungsbüro Obermeyer 1996) ................................................................. 24
Abbildung 17: Profilansicht eines Decks in Inventor (Fengler 2018) ...................................................................................... 25
Abbildung 18: Parameter des ursprünglichen Profils (Fengler 2018) ...................................................................................... 25
Abbildung 19: Profil mit neuen Parametern in Inventor vgl. (Fengler 2018) ...................................................................................... 26
Abbildung 20: Geänderte Parameter des Profils vgl. (Fengler 2018) ...................................................................................... 26
Abbildung 21: Händisch hinzugefügte Entitäten zur Darstellung der Decks ................................................................. 27
Abbildung 22: IFC-Datei eines Decks mit IFC4x1-Entitäten...................................................................................... 27
Abbildung 23: Prozess Modellierung der Stützen...................................................................................... 28
Abbildung 24: Dokumentationsausschnitt aus den Plänen (Planungsbüro Obermeyer 1986) ................................................................. 28
Abbildung 25: Ebenenansicht in Revit...................................................................................... 29
8 Literatur


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

<table>
<thead>
<tr>
<th>Name</th>
<th>Dateityp</th>
<th>Pfeiler</th>
<th>Beinummer</th>
<th>Richtung</th>
<th>UK Pfeiler</th>
<th>OK Pfeiler</th>
<th>2. Stütze</th>
<th>Höhe Fundament</th>
<th>Höhe Stütze</th>
<th>Vertikale Position</th>
<th>Entfernung zur Achse</th>
</tr>
</thead>
<tbody>
<tr>
<td>fundament_10K_20W_Mass</td>
<td>.doc</td>
<td>1</td>
<td>0</td>
<td>O</td>
<td>522</td>
<td>528.250</td>
<td>1.6</td>
<td>4.650</td>
<td>-7.700</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_10K_20W_Mass_2</td>
<td>.doc</td>
<td>1</td>
<td>1</td>
<td>W</td>
<td>522</td>
<td>528.300</td>
<td>1.6</td>
<td>4.650</td>
<td>-7.800</td>
<td>-1.375</td>
<td></td>
</tr>
<tr>
<td>fundament_10K_20W_Mass_3</td>
<td>.doc</td>
<td>2</td>
<td>0</td>
<td>O</td>
<td>521.7</td>
<td>528.090</td>
<td>1.6</td>
<td>5.780</td>
<td>-5.930</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_10K_20W_Mass_4</td>
<td>.doc</td>
<td>2</td>
<td>1</td>
<td>W</td>
<td>521.7</td>
<td>528.220</td>
<td>1.6</td>
<td>5.920</td>
<td>-9.050</td>
<td>-1.375</td>
<td></td>
</tr>
<tr>
<td>fundament_30K_Mass</td>
<td>.doc</td>
<td>3</td>
<td>0</td>
<td>O</td>
<td>521</td>
<td>528.970</td>
<td>3</td>
<td>4.870</td>
<td>-3.400</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_30K_Mass_2</td>
<td>.doc</td>
<td>3</td>
<td>1</td>
<td>W</td>
<td>521</td>
<td>528.960</td>
<td>3</td>
<td>4.980</td>
<td>-3.510</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fundament_40K_Mass</td>
<td>.doc</td>
<td>3</td>
<td>0</td>
<td>O</td>
<td>518.7</td>
<td>528.330</td>
<td>5</td>
<td>5.230</td>
<td>-11.760</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_40K_Mass_2</td>
<td>.doc</td>
<td>3</td>
<td>1</td>
<td>W</td>
<td>518.7</td>
<td>528.960</td>
<td>5</td>
<td>5.260</td>
<td>-11.790</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fundament_50K_Mass</td>
<td>.doc</td>
<td>5</td>
<td>0</td>
<td>O</td>
<td>522</td>
<td>528.562</td>
<td>1.5</td>
<td>5.062</td>
<td>-9.032</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_50K_Mass_2</td>
<td>.doc</td>
<td>5</td>
<td>2</td>
<td>W</td>
<td>522</td>
<td>528.169</td>
<td>1.5</td>
<td>5.020</td>
<td>-9.250</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_58100 Mass</td>
<td>.doc</td>
<td>6</td>
<td>1</td>
<td>O</td>
<td>520.3</td>
<td>528.020</td>
<td>1.5</td>
<td>6.120</td>
<td>-9.320</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_61000 Mass</td>
<td>.doc</td>
<td>6</td>
<td>1</td>
<td>W</td>
<td>520.6</td>
<td>528.330</td>
<td>1.5</td>
<td>5.320</td>
<td>-9.500</td>
<td>-7</td>
<td></td>
</tr>
<tr>
<td>fundament_62200 Mass</td>
<td>.doc</td>
<td>6</td>
<td>2</td>
<td>O</td>
<td>520.3</td>
<td>528.169</td>
<td>1.5</td>
<td>6.030</td>
<td>-9.300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fundament_62200 Mass_2</td>
<td>.doc</td>
<td>6</td>
<td>2</td>
<td>W</td>
<td>520.6</td>
<td>528.230</td>
<td>1.5</td>
<td>5.950</td>
<td>-9.020</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fundament_7150 Mass</td>
<td>.doc</td>
<td>7</td>
<td>0</td>
<td>O</td>
<td>520.1</td>
<td>527.650</td>
<td>2</td>
<td>5.550</td>
<td>-9.030</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_7150 Mass_2</td>
<td>.doc</td>
<td>7</td>
<td>1</td>
<td>W</td>
<td>520.3</td>
<td>527.794</td>
<td>1.5</td>
<td>5.994</td>
<td>-9.024</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>fundament_72100 Mass</td>
<td>.doc</td>
<td>7</td>
<td>2</td>
<td>W</td>
<td>520.3</td>
<td>527.683</td>
<td>1.5</td>
<td>6.030</td>
<td>-9.123</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>fundament_73100 Mass</td>
<td>.doc</td>
<td>7</td>
<td>3</td>
<td>O</td>
<td>520.1</td>
<td>525.700</td>
<td>1.5</td>
<td>4.100</td>
<td>-7.130</td>
<td>26.1</td>
<td></td>
</tr>
<tr>
<td>fundament_81600 Mass</td>
<td>.doc</td>
<td>8</td>
<td>1</td>
<td>O</td>
<td>518.9</td>
<td>527.430</td>
<td>2</td>
<td>6.530</td>
<td>-10.060</td>
<td>14.625</td>
<td></td>
</tr>
<tr>
<td>fundament_81600 Mass_2</td>
<td>.doc</td>
<td>8</td>
<td>1</td>
<td>W</td>
<td>516.4</td>
<td>526.895</td>
<td>2.5</td>
<td>7.999</td>
<td>-12.029</td>
<td>26.1</td>
<td></td>
</tr>
<tr>
<td>fundament_915 Mass</td>
<td>.doc</td>
<td>8</td>
<td>1</td>
<td>O</td>
<td>520.1</td>
<td>527.613</td>
<td>1.5</td>
<td>6.013</td>
<td>-9.043</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.0</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass_2</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.0</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass_3</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.5</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass_4</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.5</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass_5</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.5</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reiter_5-15 Mass_6</td>
<td>.doc</td>
<td>9</td>
<td>1</td>
<td>520.5</td>
<td>515.65</td>
<td>0.000</td>
<td>11.935</td>
<td>-13.468</td>
<td>12.25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GUI:

```python
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_highestIfcNumber)

    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_endingProject)
        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, numbers, typeIFC, parametersIFC)

ImportOptions

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-Files', '.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:**

```python
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, ColPositioning, 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 verknüpften 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)
```

54
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, ColOrientation, 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[0].split("(",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 %s" %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:
                    # Finden aller 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 vorkommt
            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)

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

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

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

    #absteigend sortiert
    matchObjOld.sort(reverse=True)
    matchObjOld = []
    #die Zahlen vor der Erhöhung nun wieder in einer Liste gesichert, damit mit der
    aktuellen die Berechnung durchgeführt werden kann
    for z in matchObjOld:
        xy = '#' + str(z)
        matchObjOld.append(xy)

    xx = 0
    for z in matchObjOld:
        z_new = z + number

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

    yy = 0
    for z in matchObjOld:
        pattern = z + '\b'

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

    return string
IFC4x1_LinearPlacementAdder:

```python
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
```

60
IFC4x1_DistanceOrientationAdder:

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(positioning[VarsCounter]) + ", " + str(lateral[VarsCounter]) + ", " + str(vertical[VarsCounter]) + ", $, $)"
    data.append("##" + str(orientationNumber) + " = IFCORIENTATIONEXPRESSION(#" + str(directionNumber) + ", $)"
    data.append("##" + str(directionNumber) + " = IFCDIRECTION(" + str(orientation[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(lateral[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(orientationNumber), subDis)
            parametersIFC[index] = subDisOri
            continue

    return numbers, typeIFC, parametersIFC
def importIFCDeck(numbers, typeIFC, parametersIFC, highestIFCNumber):
    path = "D:/Programme/Google Drive/Studium/Bachelorarbeit/Donnersbergerbruecke/In-invinventor/TEST/"

    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)
import re

def stringmanipulationDeck(data, numbers, typeIFC, parametersIFC):
    # teilt die IFC-Daten in einzelne Listen auf
    for z in range(0, len(data)):
        # trennt Nummern von Typen + Parametern
        splitcontent = data[z]
        splitcontent_new = splitcontent.split("=")
        typeparameters = []
        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-
                parametersIFC[index])

    return (numbers, typeIFC, parametersIFC, highestIFCNumber,)
Reunion:

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

RelConAdder:

def RelConAdder(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\可以更好([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 gefundenen 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 gefundenen 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 gefundenen 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 BuildingElementProxys
finalIfcData[index_relCon] = re.sub(r', ', '\([#0-9]+', string, finalIfcData[index_relCon])

return finalIfcData
11 Anhang C

Externen 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


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

München, 15. Juni 2018

______________________________
Vorname Nachname