Abschlussarbeit

"Individualisierbarer Konstruktionsplaner mit automatischer Bauplandeduktion"

Sebastian Rossi

January 2023

1 Motivation

Der stetig steigende Fachkräftemangel trifft auch das Baugwerbe. Laut einer Umfrage des Hauptverbandes der Deutschen Bauindustrie e.V. stufen über 75 Prozent aller befragten Unternehmen sowohl den vorherrschenden Fachkräftemangel, als auch die steigenden Energie- und Rohstoffpreise als Risiko für das eigene wirtschaftliche Wachstum ein [10]. Abbildung 1 illustriert die Entwicklung dieser Sorge über einen Zeitraum von etwas mehr als zwanzig Jahren. Damit ist es wenig überraschend, dass eine Bewegung weg von menschlichen Arbeitskräften hin zur Automatisierung existiert. Neben dem Fachkräftemangel stellt aber auch die geringe Effizienz von Bauvorhaben ein Problem dar, welche sich über den gesamten Planungs- und Bauprozess erstreckt. Diese Ineffizienz entsteht aufgrund der Vielzahl der an Bauprojekten beteiligten Experten und Unternehmen und ist, als *Fragmentierungsproblem der Bauindustrie* bezeichnet, ein bekanntes Problem [23]. Deshalb etablieren sich derzeit Standards, um Bauprojekte digital zu begleiten. Mit diesen soll gleichzeitig die Effizienz gesteigert, die Kommunikation zwischen den einzelnen Expertenteams vereinfacht, der Arbeitsplatz "Baustelle" sicherer gestaltet und ein resourcensparender Bau ermöglicht werden [3] [26]. Gleichzeitig steigen mit der Zunahme an digitalen Informationen zu Bauprojekten, auch die Möglichkeiten diese besser zu analysieren, zu optimieren und an neue Technologien zu knüpfen. Erst dadurch wurde das seit einigen Jahren erforschte Gebiet der Additiven Fertigung von Gebäuden, etwa mit Beton druckenden Roboterarmen oder mobilen Robotern, realisierbar [8]. [TODO nochmal reinschauen ob das einigermaßen als Quelle passt]. Obwohl es mittlerweile viele Projekte zur Additiven Fertigung von Gebäuden gibt, haben diese oft den Nachteil der Nicht-Parallelisierbarkeit der druckenden Roboter, die durch die Höhe der temporären Stützstrukturen (wie Kräne, Gerüste, Aufhängungen) eingeschränkte Bauhöhe und die vergleichsweise lange Bauzeit [TODO Quelle oder lange bauzeit weg]. [Frage wegen Kommentar: Star Wars roboterschwarm/Marsbesiedelung; soll ich hier sagen dass ein derzeit in der SciFi Szene aufgekommener Trend von organisierten Roboterschwärmen, die was bauen gar nicht so blöd ist und wir das ähnlich angehen?] Diesen Einschränkungen soll nun mithilfe eines Schwarmes bodengebundener automoner Roboter, welche gleichzeitig an dem Bauprojekt arbeiten, entgegengewirkt werden. Dabei sollen sich die Roboter auf den Mauern des Gebäudes selbst bewegen können, während sie dieses errichten. In dieser Arbeit liegt der Schwerpunkt allerdings nicht auf dem Entwicklen der Roboter selbst, sondern das Erstellen eines Bauplans für den genannten Roboterschwarm

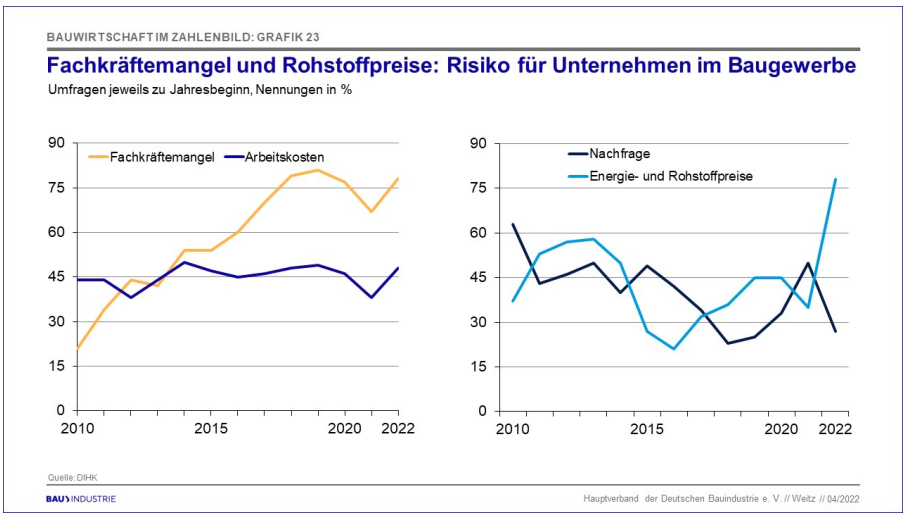


Figure 1: Während wirtschaftliches Risiko durch eine potentiell sinkende Nachfrage nach Bauaufträgen und eventuell steigender Arbeitskosten unverändert blieben oder sogar als weniger relevant bewertet wurden, ist ein deutlicher Anstieg aufgrund des vorherrschenden Fachkräftemangels und der Energie- und Rohstoffpreise zu erkennen.

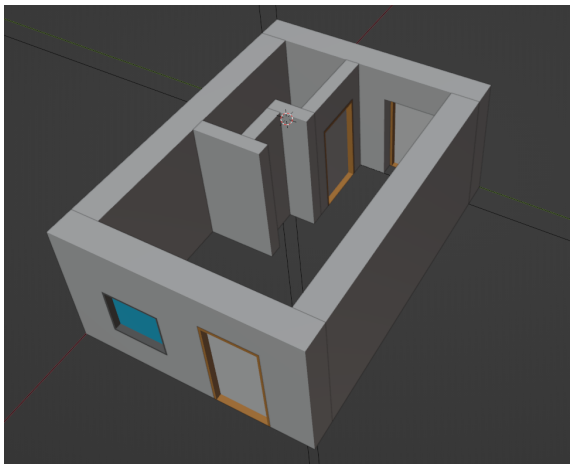
basierend auf einem 3D Modells des Gebäudes. Das Erstellen des Modells innerhalb eines nutzerfreundlichen 3D Editors stellt nicht nur die geometrischen und physikalischen Eigenschaften des Gebäudes digital bereit, sondern verschlankt auch die Kommunikation zwischen Endnutzer und Architekt oder ersetzt letzteren komplett. Gleichzeitig bildet diese Arbeit damit auch den Trend hin zur sogenannten Massenpersonalisierung ab, welcher als Nachfolgetrend zur Massenproduktion und als "heilger Gral" der Fertigung angesehen wird . Dieser Trend ist auch für die Bauindustrie interessant, denn auch hier schafft die Möglichkeit sämtliche Kundenwünsche an ein Produkt (oder Gebäude) umzusetzen, ohne dafür spezielles Werkzeug herstellen zu müssen, neue Gewinnmöglichkeiten. Mit der Option der Modellierung des Gebäudes durch den Kunden selbst, ist die Kommunikation mit den Architekten über dessen Wünsche effizienter, da beide Parteien zusammen an dem Modell arbeiten können. Im Anschluss an den Designprozess des 3D Models des Gebäudes, soll dieses in einen Bauplan übersetzt werden, welcher alle notwendigen Informationen für den oben genannten Roboterschwarm enthält, sodass dieser das Gebäude selbstständig errichten kann [TODO ugs]. Dieses Konzept wird anhand nachfolgender Fallstudie(n) getestet.

2 Fallstudien

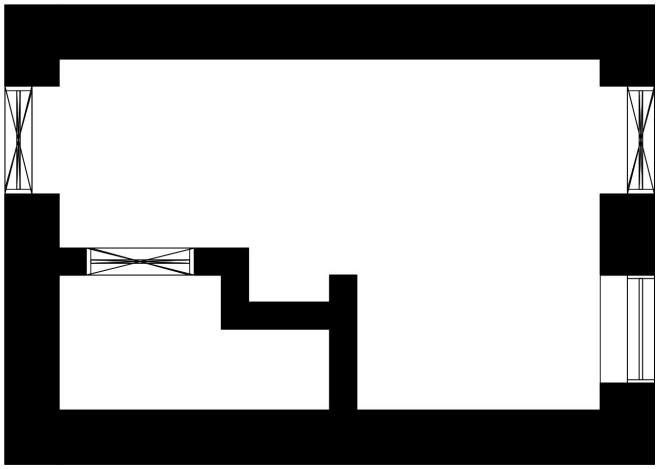
2.1 Bauplandeduktion am Beispiel von LEGO Gebäuden

Frage vorweg: Wenn der Konstruktionsplaner Teil meiner Ergebnisse aus der Masterarbeit ist, wieso macht es dann Sinn, dass ich dann in dem Szenario von einem Modell ausgehe, welches ich aber nur mit eben diesem Konstruktionsplaner erstellen konnte?

Ziel dieses Szenarios ist es, aus dem in Abbildung 2a dargestellten 3D Modell ein Bauplan zu generieren. Zu sehen ist der Plan eines einfachen Hauses mit einem Stockwerk. Dieses besitzt eine Eingangstür, eine Terassentür neben einem Fenster und eine Tür, die das Badezimmer vom Hauptraum trennt. Türen und Fenster stellen eine Herausforderung für



(a) 3D Modell innerhalb von Blender.



(b) Gebäudeplan des 3D Modells.

Figure 2: Modell eines Studentenzimmers (verschiedene Darstellungsformen).

den Planungsalgorithmus dar, da der Verlauf einer ansonsten durchgängigen Wand dadurch unterbrochen wird und Lücken aufweist. Details wie Duschen, Betten, Toiletten und ähnliche Komponenten, sind für dieses Szenario irrelevant, da diese keinen Effekt auf die Struktur der Wände haben und wurden aus diesem Grund bewusst weggelassen. Das Modell wurde mithilfe der in Kapitel 5 näher behandelten Technologien erstellt und enstpricht in seiner Struktur einem verbreiteten Industriestandard. Dafür wurden zwei Wandtypen definiert, die jeweils unterschiedliche Wanddicken vorgeben. So gibt es breite Außen- und dünne Innenwände. Diese entsprechen in ihren Maßen dem Raster, welches das *LEGO System* (siehe Kapitel 5) vorgibt. Für breite Wände gilt, dass diese immer zwei Noppen breit, mindestens eine Noppe lang und einen Stein hoch sein muss. Für dünne Wände hingegen gilt eine feste Breite von einer Noppe, ebenfalls eine Mindestlänge von einer Noppe und eine Mindesthöhe von einem Stein. Beide Wandtypen können nur Höhen beziehungsweise Längen aufweisen, die jeweils einem Vielfachen der Höhe oder Länge des kleinsten Legosteins aufweisen, der zum Bau der Wände verwendet werden soll (hier der 1x1 Stein). Daraus resultiert ein Raster, welches ebenfalls für die Ausmaße und Positionen der Fenster und Türen einzuhalten ist. Dieses Raster gilt es, abhängig des ausgewählten Wandtyps, in den Editor zu integrieren, um das Modellieren solcher Gebäude nutzerfreundlich zu gestalten und nicht durch ständiges Messen und Eintragen genauer Positionen oder Maße zu unterbrechen.

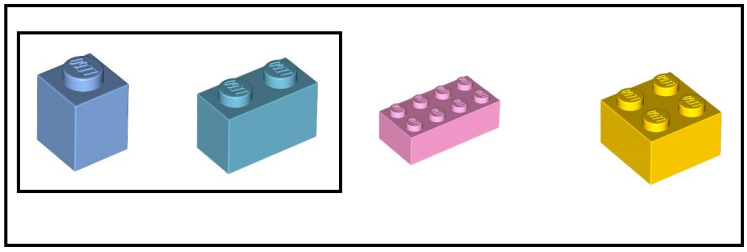


Figure 3: LEGO Steintypen für die Innen- und Außenwände. (TODO Bild ist sehr hässlich)

Nicht nur die Abmessungen der Wände müssen in ein Raster fallen, auch deren Rotation wird in diesem Szenario auf 90° Schritte limitiert. Das stellt in diesem Fall eine vertretbare Enschränkung dar, da es ohnehin dem intuitiven Umgang mit LEGO Steinen und gleichzeitig dem Baustil der meisten einfachen Gebäuden entspricht. Folglich muss ein Format

für die Bausteintypen entwickelt werden, aus welchen all diese Informationen abgeleitet werden können. Dieses Format muss sowohl von dem Editor selbst verwendet, als auch zur Berechnung innerhalb des Planungsalgorithmus herangezogen werden. Außerdem werden weitere Regeln benötigt, um den resultierenden Plan näher an das Vorgehen eines realen Baus zu bringen. So werden beim Errichten von Häusern zuerst die Ecken (der Schnittpunkt zweier Wände) um eine Stufe erhöht, um im Anschluss die geraden Wandabschnitte aufzufüllen. Damit wird vermieden, dass in den ohnehin schon komplexeren Eckbereichen auch noch zugeschnittene Ziegel notwenig werden. Stattdessen schneidet man diese erst zurecht, wenn sich dann eher mittig im Wandabschnitt Lücken ergeben, die kleiner sind als die vorhandenen Ziegel. Zwar wird dies im Fall von Legosteinen nie auftreten, aber es ist übersichtlicher das Problem in dem vorliegenden eingeschränkten Szenario zu beschreiben. Zusätzlich müssen Regeln in den Planungsalgorithmus eingeführt werden, die diverse Mauerwerksverbände (vorgestellt in Kapitel 5) erzwingen zu können, um damit die Stabilität und gleichmäßige Kraftverteilung in einer Wand zu gewährleisten und sich ebenfalls möglichst nah an der Realität des Mauerbaus zu bewegen.

2.2 Szenario mit Läufern und Bindern

Dicke Wände, die mit Läufern und Bindern erstellt werden müssen. Eckfälle sehr komplex (siehe Basics)

2.3 Szenario mit veränderbaren Bausteintypen

Wie können wir die Bausteine veränderbar machen, sprich die Bearbeitungsmöglichkeiten während des Baus (schneiden eines Ziegels) miteinbeziehen in unser Bausteinformat. Erstmal das schneiden nur parallel zu den ebenen eines rechteckigen ziegelsteins betrachten, denn bei "schrägen" schnitten entstehen neue komplexe Formen.. Wie generiert man daraus dann Baupläne -> riesiges aber cooles Optimierungsproblem! Vlt geht das richtung constraint programming? Sprich den Bauplan als Lösung für ein beschränktes Problem ansehen. Total viele Fragen, keine Ahnung wo zu beginnen aber klingt cool

2.4 Sternchenaufgabe: Szenario mit "runden" Wänden und arbiträren (nicht rechteckige) Bausteinformen / schräge schnitte

Was machen wir wenn die Wand nicht perfekt mit den vorgegebenen Bausteintypen gebaut werden kann (Beispiel ein runder Turm) Bausteinverbindungen (wie Mörtel) betrachten und als *Verbindungselement* in das Bausteinformat mit aufnehmen? Wie kann man diesen so einschränken, dass nur physisch machbares ausgrechnet wird. Was wenn die Bausteine arbiträre Formen haben und nur in sehr komplexen Mustern eine "dichte" Wand ergeben -> tiling Probleme.

3 Problemstellung

Ziel der Arbeit ist es einen Workflow zu schaffen, welcher es einem Nutzer ermöglicht ein Gebäude in einem 3D Designer zu planen, das im Anschluss in einen durch einen heterogenen Roboterschwarm ausführbaren Bauplan übersetzt wird. Dies erfolgt in mehreren Schritten, welche sich jeweils mit unterschiedlichen Fragestellungen befassen. Anhand der Teilschritte

lässt sich diese umfangreiche Problemstellung logisch einteilen und die jeweilgen Kernprobleme und Fragestellungen der einzelnen Schritte werden klar.

Roboter auf Mauer -> Test nicht mit Schaumstoffbricks möglich da gewicht Lücken (Fenster, Türen)

3.1 Bausteindefinition

Wie können wir zu einem bestimmten Wandtyp ein Set an Bausteintypen definieren, mit welchen Wände diesen Typs gebaut werden müssen. Dabei sollen die Bausteine nicht auf Quader beschränkt, sondern beliebige Körpern sein können. Ebenfalls relevant sind eventuelle die Bausteinverbindungen, die ebenfalls Einschränkungen haben. Ein Beispiel dafür ist etwa Mörtel bei Ziegelwänden.

3.2 Wall-Detailing und Tiling

Wie kann man algorithmisch mit einem Set an Bausteintypen eine Wand, welche als Mesh vorliegt, vollständig erbauen und daraus einen Bauplan herleiten (Stichwort Abhängigkeitsgraph).

3.3 Definition Bauplan

Was muss ein Bauplan konkret beinhalten?

4 Aufgabenstellung

Nachfolgend werden in chronologischer Reihenfolge alle notwendigen Teilaufgaben aufgezeigt, die gelöst werden müssen, um ein Gebäude zu planen, welches anschließend von Robotern gebaut werden kann.

4.1 Modellierung des 3D Modells

Es muss ein Weg gefunden werden, dem Nutzer das Modellieren von Teilstücken eines Gebäudes zu erleichtern, sodass er keine einzelnen Bauteile in dem Editor verwenden muss, sondern Konglomerate (wie etwa eine Wand oder ein Fenster) zur Verfügung hat, welche als Ganzes bewegt oder verformt werden können. Dabei dürfen aber Informationen über die eigentlich zugrundeliegenden Bauteile (z.B. Ziegel) entweder nicht verloren gehen und implizit vorgegeben sein (z.B. durch das aufzwingen eines Rasters, welches dem Formfaktor der Bauteile entspricht, sodass etwa eine Wand nicht in beliebig kleinen Schritten vergrößert werden kann, sondern immer nur einen Längensprung um die Länge des kleinsten zugrundeliegenden Ziegel macht) oder das Programm versucht, das Gebäude nach beenden des Designprozesses möglichst gut mithilfe des vorgegebenen Sets an Bauteilen abzubilden. Das führt aber dazu einen möglichst guten Kompromiss finden zu müssen, was durch ein Raster von vornherein vermieden werden kann.

Fragestellung: Wie kann das Modellieren eines Gebäudes für Nutzer vereinfacht werden ohne wichtige Informationen zu verlieren?

4.2 Finden eines geeigneten Dateiformats

Wie breits vorweg genommen, muss es möglich sein aus dem Gebäudemodell eine Menge an Bauteilen von vorgegebenem Typen zu berechnen, die das Gebäude komplett (oder möglichst gut) abbildet. Das geht entweder, indem man diese Menge während dem Designprozess impliziert vorliegen hat, da alle größeren Teilstücke (wie etwa eine Wand) schon mithilfe der zugrunde liegenden Bauteile definiert wurden oder durch das einmalige Konvertieren eines von den Bauteilen völlig losgelösten 3D Modells in eben jene Menge an Bauteilen. Im ersten Fall fällt das Konvertieren weg und der Nutzer arbeitet immer direkt auf dem Datenformat, welches anschließend in späteren Schritten verwendet wird. Im zweiten Fall kann das konvertierte Modell vermutlich nicht wieder in den Editor zurückgeladen werden, sodass man ein "Design Speicherformat" und ein "Konvertiertes Datenformat" hätte. Dies ist aber ebenfalls eine gängige Praxis bei z.B. Vektorgraphik-Bearbeitungsprogrammen, die in einem proprietären Format Daten über mathematisch definierte Kurven halten und der Nutzer daraus Bilder in pixelbasierte Formaten wie jpg oder png generieren kann. Dabei gehen viele Informationen verloren und man kann das exportierte Bild nicht in derselben Art wieder in das Programm laden. Mit Blick auf die Domäne des Häuserbauens ist allerdings ein möglichst großer Freiheitsgrad wünschenswert, was für den Konvertierungsansatz spricht. Dabei wird es wie bereits erwähnt herausfordernd eine möglichst gute Abbildung des im Editor designten Gebäudes mithilfe eines limitierten Sets an Bauteiltypen zu finden, welches nach Möglichkeit alle vom Nutzer gewünschten Eigenschaften behält. Als Speicherformat kann ein im Bauingeneursbereich weit etablierter Standard verwendet werden, der neben den bloßen geometrischen Informationen über das Gebäude auch wichtige Details aus anderen Fachbereichen integriert. Dieser Standard wird im weiteren Verlauf dieses Exposés vorgestellt.

Fragestellung: Das Gebäude im Hintergrund immer als Menge von definierten Bauteilen oder als gängiges 3D Modell speichern, welches irgendwann in Bauteile überführt werden muss?

4.3 Finden eines geeigneten Bauplans für Roboterschwärme

Um dem Umfang und die Komplexität diesen Schrittes zu erfassen, wurde Ludwigs Dissertation gelesen [22]. Diese beschäftigt sich mit dem Finden geeigneter Baupläne für beliebige Produkte mit Einbeziehung diverser (etwa geometrischer) Einschränkungen der Einzelteile oder der Monatageroboter. Mit einem solchen Verfahren kann die derzeit teure Individualisierung von Produkten kosteneffizient ermöglicht werden, da sämtliche Produktionsschritte automatisch herausgefunden werden und damit Zeit sowohl in der Planung, als auch beim Einlernen der Montageroboter gespart werden kann. Im Grunde ist die Problemstellung aus dieser Arbeit folgende: Die Menge möglicher Baupläne für ein Produkt steigt exponentiell mit der Menge der verwendeten Bauteile und kann als sehr großer Suchraum für passende Lösungen angesehen werden, welcher z.B. mithilfe bestimmter Anforderungen an den resultierenden Bauplan verkleinert werden kann. Tatsächlich kann diese Fragestellung direkt auf die Domäne des Häuserbauens angewendet werden: Bei Gebäuden handelt es sich um Objekte mit vielen Tausend Bauteilen, was aufgrund des exponentiellen Wachstums des Suchraums zu einer unmöglich großen Menge potentieller Baupläne führt. Um dem entgegenzuwirken, sucht man Constraints, wie etwa die Erkenntnis, dass Ziegel nur von unten nach oben aufgeschichtet werden können. Damit fallen eine Vielzahl an Bauplänen weg und der Suchraum verkleinert sich erheblich. Auch die Beschaffenheit der Bauteile und der Roboterendeffektoren und -körper bringen Constraints mit sich. Allerdings ist der Einsatz von mehreren Robotern ein neuer Aspekt, der in Ludwigs Arbeit nicht berücksichtigt wird. Dennoch liegt es nahe die Ergebnisse aus seiner Arbeit für dieses Projekt zu verwenden

und für die Abarbeitung durch mehrere Agenten anzupassen. Als Input für seinen "Suchalgorithums" wurde in Ludwigs Dissertation ein eigenes xml-artiges Datenformat zur vollständigen Beschreibung des fertigen Bauteils eingeführt auf Basis dessen ein nach einer Heuristik optimierter Bauplan herausgesucht wird. Oftmals hat diese Heuristik zum Ziel, die Montagedauer oder die Kosten der Monatge zu minimieren. Es ergibt Sinn dieses Format als Ergebnis aus dem vorherigen Schritt anzusehen. Das bedeutet, dass das 3D Modell des Gebäudes in das von Ludwig verwendete Format übersetzt werden muss, welches alle für die Suchopertation notwendigen Informationen über die individuellen Bauteile besitzt. Somit wird Ludiwgs Format zum "Konvertierten Datenformat", das aus dem 3D Modell berechnet wird. Das Ergbnis der Suchoperation auf dem Konvertierten Datenformat stellt ein für einen heterogenen Roboterschwarm abarbeitbarer Bauplan dar. Dafür muss die Suchoperation in Teilen angepasst werden.

Fragestellung: Wie wird aus dem Bauteilplan ein für Roboterschwärme abarbeitbarer Plan zur Erstellung des Gebäudes?

4.4 Format zur Beschreibung der Bauteile aus welchen Wände bestehen

Wie beschreibt man Bauteile, sodass neben der geometrischen Struktur z.B. auch die Berarbeitungsmöglichkeiten (wie etwa das Zerschneiden eines Ziegels) angegeben/eingeschränkt werden können. Wie verwendet man diese Informationen, um Wände, die aus einem solchen Baustein errichtet werden sollen schon vorab im 3D Editor in die darin vorgegebenen Einschränlungen zu pressen (z.B. Legosteine sollten nicht zerschnitten werden -> es gibt also einen Stein, welcher als kleinste Größenänderung für die Wand gelten muss. Die Wand wird quasi in ein Grid gepresst. Wie sieht das aber aus, wenn es sich nicht um so einfache geometrische Körper handelt? -> sau rechenaufwendig?)

5 Grundlagen

Wie bereits oben erwähnt, wird für diese Arbeit ein geeignetes Speicherformat für 3D Modelle von Gebäuden benötigt. Um den Bau von Gebäuden zu automatisieren, ist es notwendig die Domäne *Gebäude* vollständig digital abbilden zu können. Hilfreich ist dabei, wichtige Daten über bestimmte Bestandteile des Gebäudes direkt in dem Modell zu integrieren auf Basis derer etwa Kostenberechnungen durchgeführt oder Materialmengen herausgefunden werden können. Da diese oft von verschiedenen Experten vieler Fachbereiche (etwa aus den Bereichen der Architektur, des Bauwesens oder der Statik) stammen, muss das Format sehr flexibel und im besten Fall auch zeitgleich bearbeitbar sein. Dafür werden seit 2000 die *Industry Foundation Classes* (IFC) von buildingsmart entwickelt, deren Anwendung im internationalen Bauwesen mittlerweile weit verbreitet ist [15].

5.1 Industry Foundation Classes

In der Spezifikation des Standards selbst, wird dieser wie folgt beschrieben: "Die Industry Foundation Classes (IFC) sind ein offener internationaler Standard für Daten des Building Information Model (BIM), welche zwischen Softwareanwendungen ausgetauscht, gemeinsam genutzt und von den verschiedenen Akteuren der Bauindustrie und des Gebäudemanagements verwendet werden. Der Standard enthält Definitionen für Daten, die für die Lebenszyklen von Gebäude- und Infrastrukturarbeiten erforderlich sind. Die bis jetzt in die IFC aufgenommenen Infrastrukturtypen umfassen Brücken,

Straßen, Eisenbahnen, Wasserstraßen und Hafenanlagen" (aus dem Englischen) [17]. Eine frühere Version des IFC Standards ist unter der Bezeichnung ISO 16739 [18] registriert. Da die IFC aber nach wie vor kontinuierlich weiterentwickelt werden, wird in dieser Arbeit die derzeit neueste Version verwendet. Diese ist die IFC Spezifikation 4.3.1.0 [16]. Das verbreitetste Austauschformat für IFC ist das Step Physical File Format, welches im ISO 10303 Teil 21 registriert ist [13]. Zudem gibt es speicherreduziertere Formate wie ifcZip oder für Menschen lesbarere Formate wie ifcXML [14] [11].

5.1.1 IFC 4.3.1.0 Aufbau

Im Grunde definieren die *Industry Foundation Classes* eine Vielzahl an Klassen, die in einer komplexen Hierarchie angeordnet den Grundstock des Datenmodells bilden. Diese sind anfangs abstrakte Konzepte, die sich mit zunehmender Tiefe in der Hierarche konkretisieren.

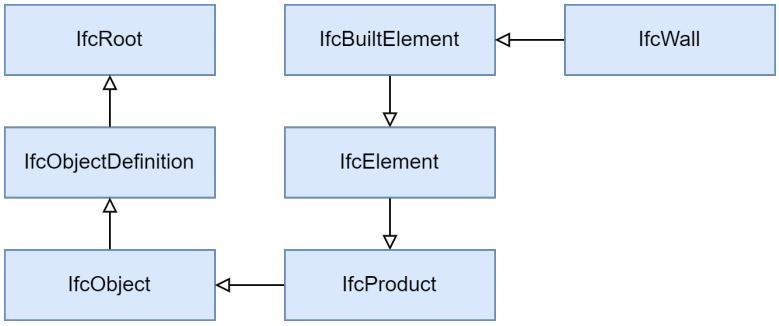


Figure 4: Klassenhierarchie am Beispiel der Klasse *IfcWall*

Da sich diese Arbeit zum größten Teil mit aus Wänden bestehenden Gebäuden befasst, wird nachfolgend die Klasse *IfcWall* wiederholt als Beispiel herangezogen. Der für diese Klasse relevante Ausschnitt aus der Klassenhierarchie ist in Abbildung 4 dargestellt. Objekte werden von dem Standard in Relation zueinander gestellt, um komplexere Zusammenhänge darzustellen. In Abbildung 5 erkennt man den Zusammenhang zwischen einem Objekt des Types *IfcWall*, des Stockerwerks, welches diese Wand referenziert und wiederum selbst Teil eines *IfcBuildings* ist, bis hin zur obersten Komponente eines Ifc Projektes, dem gleichnamigen *IfcProject*.

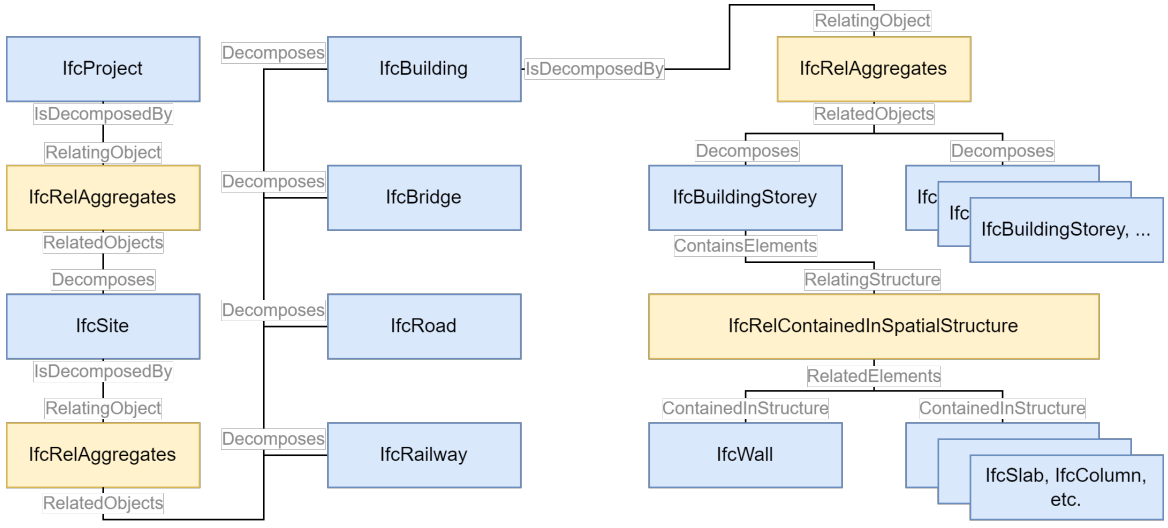


Figure 5: Relation der IfcWall und einem IfcProjekt

5.1.2 IfcPropertySets und IfcQuantitySets

Mit dem Erweitern des Gebäudemodells um möglichst viele Informationen, schafft man einen detailierten digitalen Zwilling, der neben der bloßen Darstellung des Gebäudes als 3D Modell zb. auch eine präzisere Kosten- und Zeitschätzung für den Bau ermöglicht [14]. Dies kann mithilfe der IfcPropertySets und der IfcQuantitySets umgesetzt werden blablabla.

5.1.3 Positionierung von IFCProducts

TODO: Positionierung ist anscheinend bisschen verschachtelt Blablabla so können aus einem IFC Projekt alle relevanten Klassen mit den für diese Arbeit erforderlichen Eigenschaften extrahiert werden.

5.1.4 IfcOpeningElement

Wie funktionieren Löcher in z.B. IfcWall? -> IfcOpeningElement. Am Beispiel eines Fensters erklären. In dessen Doku steht dass Windows auf ein IfcOpeningElement verweisen, welches ein Stück Wand wegmacht und z.b. durch das Fenster aufgefüllt wird. TODO: mit rausparsen und aus wandmeshes rauschneiden vor detailing.

ifcopeningelement um lücken in wände zu machen. "The opening element stands for opening, recess or chase, all reflecting voids. It represents a void within any element that has physical manifestation. Openings can be inserted into walls, slabs, beams, columns, or other elements." Auszug aus Doku. und This also includes IfcOpeningElements, the mechanism used to extract openings for windows and doors, typically from walls.

5.2 IFC for Blender

5.2.1 Blender

Blender ist eines der beliebtesten Open Source Programme zur Modellierung von 3D Modellen und Animationen [4]. Aufgrund dessen existieren auch eine Vielzahl an freien Erweiterungen bzw. Plugins - unter anderem auch eine Integration von IFC Projekten.

5.2.2 blenderbim

Neben kommerziellen Produkten wie etwa revit von autodesk [24] zur Modellierung von IFC Modellen, gibt es auch für Blender ein freies Plugin, um IFC Modelle zu erstellen [5]. Dieses Plugin ermöglicht es neben dem bloßen Designen des Gebäudes in kurzer Zeit z.B. detailierte Zeichnungen verschiedener Perspektiven herauszuarbeiten, die z.B. von Bauingeneuren verwendet werden können, um einzelne Stockwerke oder Verkabelungen zu planen. Blenderbim selbst kapselt unter anderem die Open Source Python Bibliothek *IfcOpenShell*, sodass diese in der Blender Laufzeitumgebung zur Verfügung steht [12].

5.2.3 IfcOpenShell

```python
import ifcopenshell
from ifcopenshell import geom
from stl import mesh, Mode
import numpy as np

settings = ifcopenshell.geom.settings()
settings.set(settings.USE_WORLD_COORDS, True)

ifc_file = ifcopenshell.open("../models/sample_house.ifc")
products = ifc_file.by_type("IfcProduct")
meshes = []

for product in products:
    if product.Representation and product.is_a("IfcWall"):
        shape = ifcopenshell.geom.create_shape(settings, product)
        vertices = np.array(shape.geometry.verts).reshape((-1, 3))
        edges = np.array(shape.geometry.edges)
        faces = np.array(shape.geometry.faces).reshape((-1, 3))

        m = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
        for i, f in enumerate(faces):
            for j in range(3):
                m.vectors[i][j] = vertices[f[j], :]
        meshes.append(m)

# Create the combined mesh
combined = mesh.Mesh(np.concatenate([m.data for m in meshes]))
combined.save('cube.stl', mode=Mode.ASCII)
```

Listing 1: Beispielprogrammcode um bestimmte Daten aus einem IFC File zu laden und daraus ein Mesh zu generieren

TODO Beispielcode zum extrahieren einer Wand und deren geometrischen Eigenschaften

conda create -n masterarbeit conda activate masterarbeit conda install -c conda-forge ifcopenshell conda install -c conda-forge ipykernel python -m ipykernel install –user –name=masterarbeit conda install -c conda-forge pythonocc-core=7.7.0 conda install -c conda-forge meshplot

conda create -n masterarbeit conda activate masterarbeit conda install -c conda-forge ifcopenshell conda install -c conda-forge pythonocc-core=7.7.0 conda install -c anaconda pyqt

the ABS models to solve floor layout problems constraint solver / programming

5.3 Building Information Modeling

Ein weiterer Punkt, der für die Verwendung von IFC spricht ist das sogenannte *Building Information Modeling* (BIM) [6]. Ein Definitionsvorschlag lautet wie folgt: "BIM ist definiert als der Einsatz von Informations- und Kommunikationstechnologien zur Verschlankung der Prozesse im Lebenszyklus von Gebäuden, um eine sicherere und produktivere Umgebung für die Bewohner zu schaffen, die Umwelt so wenig wie möglich zu belasten und die Effizienz der Betriebsabläufe für

die Eigentümer während des gesamten Lebenszyklus des Gebäudes zu erhöhen" (Übersetzt aus dem Englischen) [21]. Zum Lebenszyklus eines Gebäudes gehören etwa anfangs das Planen und Designen, später das Bauen, das Verwenden und Instandhalten und nach eventuellen Renovierungen das Abreißen. BIM kommt in all diesen Phasen zum Tragen und erleichtert diese Prozesse durch Anbieten einer einheitlichen Schnittstelle für alle am Infrastrukturbau und -management beteiligten Personen. Zusätzlich ermöglicht BIM eine exakte Dokumentation des Geschehens in sämtlichen Phasen des Bauwerks, was unter anderem zu einer genaueren Zeit- und Kostenplanung führt. Auch Verantwortlichkeiten sind Teil von BIM, was zu einer erhöhten Produktivität beiträgt. Um nun das Zusammenarbeiten der unterschiedlichen Fachbereiche zu erleichtern, gibt es sogennante BIM-Server auf welchen mehrere Arbeitende synchron an einem Projekt arbeiten können, während sie jeweils die für ihren Aufgabenbereich passende Ansicht vor sich haben. BIM-Server unterstützen zusätzlich eine Versionierung des Fortschritts an einem Projekt.

In einem Gespräch mit einem Ingeneur aus dem Bereich "Energysystemtechnik" kam zur Sprache, dass viele Bereiche von BIM noch nicht ganz Einzug in Deutschland gefunden haben. Eben jene "Kollaboration über einen BIM-Server mit Änderungsmanagement etc. [sei] (noch) nicht üblich, da noch nicht alle Beteiligten dazu in der Lage sind. Vor allem Bauherren, Architekten und Baufirmen können es nicht". Weiter sei "auch unklar, wer für falsche Angaben haftet und wer die Konsistenz aller Daten gewährleistet". Auf der anderen Seite sei "das im BIM festegelegte Datenformat IFC das Maß der Dinge und auch bei uns so in Verwendung". Auch das Einpflegen "ergänzende[r] Bauteilinformationen (z.B. zu Gewicht, Dämmwert, Recyclebarkeit, CO2 Fußabdruck, etc." finden Einsatz und sind Teil seines Alltags. Für ihn wichtig ist ebenfalls der Betrieb des Gebäudes. Hier unterstützt BIM, indem sämtliche Teile der Installationen in einem Gebäude, wie z.B Fensterdichtungen, Kabel, Rohre, Sicherungen oder eine Umwälzpumpe individuelle Teilenummern zugewiesen bekommen, hinter welchen alle Daten wie etwa Hersteller, Bestellnummern, Lebensdauer, Wartungshistorie oder Entsorgungsnachweise vermerkt sind. Dies wurde allerdings "angesichts der Realität der Handwerker und Gebäudenutzer für völlig unrealisitsch und auch etwas over-engineered" eingestuft. Trotzdem sei "BIM [...] das große Ding in der Bauwelt und der einzige echte Standard".

5.4 brick schema

ChatGPT "While IFC is primarily focused on representing building information for interoperability between software applications, BrickSchema focuses on providing a standardized and semantically rich representation of building systems and their components. In practice, BrickSchema can be used in conjunction with IFC to enhance the semantic representation and analysis of building data, particularly when it comes to systems-level information.

By using IFC as a foundation for representing the overall building information and combining it with the more detailed and specialized semantic representation offered by BrickSchema, it becomes possible to achieve a comprehensive and interoperable representation of building information that can support various use cases, including energy modeling, fault detection and diagnosis, and optimization of building performance."

5.5 opensourcebim

Während es vorwiegend kommerzielle Produkte gibt, die Unternehmen das Arbeiten mit BIM ermöglichen, exisitiert auch hier eine Open Source Bewegung. Darin enthalten sind 70 Repositories, unter anderem ein BIM-Server inklusive diverser

Clients für Endanwender und Werkzeuge, um einfacher mit den IFC Files zu agieren [25]. Ein kurzer Test hat gezeigt, dass die in Blender modellierte IFC Files tatsächlich über einen "Anzeige-Client", der mit einer BIM-Server verbunden ist, angezeigt werden können. Obwohl die Verwendung des BIM-Servers für diese Arbeit nicht notwendig ist, besteht die Option diesen künftig mit in den Workflow zu integrieren, da damit auch das simultante Arbeiten an einem IFC File möglich ist, was in Blender nur teilweise und mit dem Einsatz von Plugins ermöglicht wird. Dabei ist fraglich, ob diese Plugins dann ebenfalls die IFC Erweiterung unterstützen. Das stellt einen Praxisbezug zum aktuell verwendeten Stand dieser Technologien her, was in der oftmals konzeptionellen Natur der Forschung nicht immer der Fall ist. Wie auch Blender untertützt BIM-Server das Einbinden von eigenen Plugins, sodass eine Erweiterung um neue Funktionalität möglich ist. Die Plugins werden in Java geschrieben. Der Server bietet aber auch eine REST Schnittstelle an, um Clients in anderen Sprachen anzubinden.

5.6 LEGO

Ein 1x1 LEGO Stein hat eine quadratische Grundfläche von $7.8mm$ x $7.8mm$. Zwischen zwei nebeneinander platzierten Steinen ist ein Abstand von $0.2mm$. Daraus ergibt sich ein Rastermaß von $8mm$ x $8mm$. In Abbildung 6 werden zur Veranschaulichung die Maße des populären 2x4 Steines aufgeschlüsselt. Die für ein dreidimensionales Raster noch fehlende Größe ist die Höhe der Steine. Diese beträgt $9.6mm$. Der Abstand zwischen zwei übereinander gestapelten Steinen hängt von dem Druck ab, der beim Zusammenstecken geleistet wurde. Dennoch kann dieser vernachlässigt, sprich als Abstand von $0.0mm$ gewertet werden.

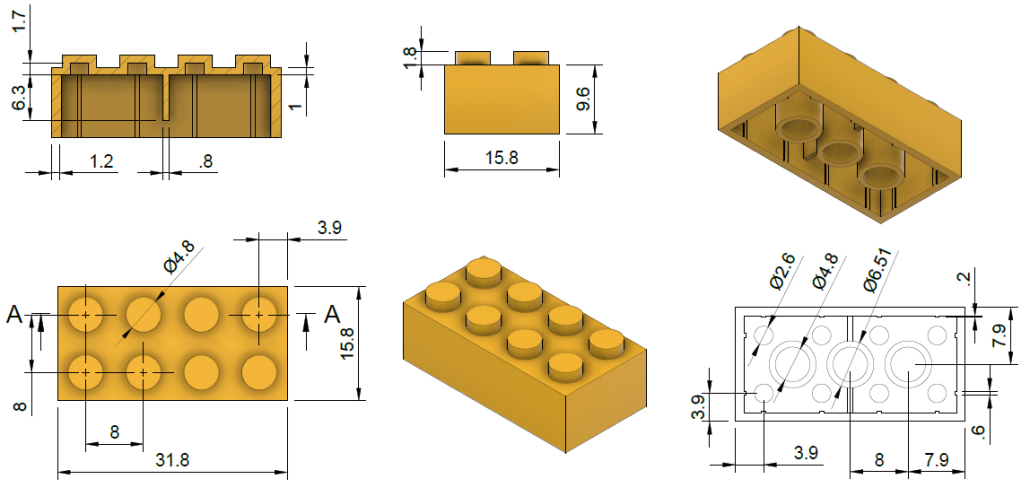


Figure 6: Maße des Standard 2x4 LEGO Steins [19]

5.7 Mauerwerksverband

Als Mauerwerksverband bezeichnet man bestimmte, gleichmäßige Anordnungen von Mauersteinen, um einen homogenen Mauerwerkskörper zu erreichen [20]. Damit kann eine gleichmäßige Kraftverteilung innerhalb der Mauer gewährleistet werden. Eine wichtige Rolle nimmt dabei das Überbindemaß ein, welches die Mindestüberlappung von Mauersteinen aus zwei Schichten der Mauer vorgibt. Für das planmäßige Überbindemaß l_{ol} gilt für übliche Mauersteine mit Schichthöhen

$h_u \leq 249mm$ nach DIN EN 1996-1-1: $l_{ol} \geq 0,4h_u \geq 45mm$[2][7]. Zudem wird darin die Mindestwanddicke für tragendes Mauerwerk, sofern aus Gründen der Standsicherheit, der Bauphysik oder des Brandschutzes nicht größere Dicken erforderlich sind, auf $t_{min} = 115mm$ festgelegt [2][7].

Man unterscheidet zwei Arten von Mauerwerk: dem Einsteinmauerwerk und dem Verbandsmauerwerk. Wie schon dem Namen zu entnehmen, handelt es sich beim Einsteinmauerwerk und ein Mauerwerk, bei welchem die Wanddicke der Steindicke entspricht. Hier muss das Überbindemaß lediglich über die Wandlängsrichtung eingehalten werden. Bei Verbandsmauerwerk gilt dies zusätzlich für die Wandquerrichtung. [1]

TODO Abbildungen von verschiedenen Verbänden TODO Ecklösungen TODO Rund Wände, keine 90 Grad Ecken

DIN 1053-1 (wurde durch DIN EN 1996-1-1 ersetzt) anstoßendes Wandstück: der bereich an dem zwei wandsegmente aneinanderstoßen z.b eine ecke Verzahnung: aus https://baulexikon.beuth.de/VERZAHNUNG.HTM : Verzahnung, im Mauerwerksbau übliche Technik, beim Herstellen einer Wand eine Verbindungsstelle für eine später zu errichtende und in die bereits bestehende einzubindende Wand den Verbandsregeln entsprechend vorzubereiten. Es gibt Lochverzahnung, stehende und liegende Verzahnung. Nur die letztgenannte Verzahnungsart gilt nach DIN 1053-1 als ausreichende Verbindung zwischen tragenden und aussteifenden Mauerwerkswänden Überbindemaß ist wichtig um Mauerwerksverbände zu bewerten

5.8 Definitionen

5.8.1 Bausteintyp

5.8.2 Baustein

5.8.3 Wand

6 Related Work

6.1 3D Druck und Additive Fertigung von Gebäuden

6.2 Legeroboter

6.3 Materialien

6.4 Bausteine

6.5 Mauer detailing und das (3D) Bin Packing Problem

TODO hinführen über Bin Packing hin zu "spezialfall" Wall detailing mit arbiträren Bausteinen und Eigenschaften (wie versetzen der ziegel)

TODO über bin packing schreiben, erklären paper suchen, lösungsansätze zu np hartem problem

Xu Chengran et al. haben in ihrem Paper "Optimal brick layout of masonry walls based on intelligent evolutionary algorithm and building information modeling" verschiedene Optimierungsansätze aus dem Bereich des 2D Packaging Problems getestet [27]. Konkret wurden drei Algorithmen verwendet: Differential Evolution, Particle Swarm Optimization und Neighbourhood Field Optimization. Außerdem wird ein drei-phasiges Vorgehen vorgeschlagen: Data collection, Brick layout und Data Output. Dieses Vorgehen eignet sich auch für das Finden von Bausteinkonfigurationen in dieser Arbeit, da zuerst alle relevanten geometrischen Daten (in diesem Fall Wände, Fenster, Türen usw.) aus dem 3D Modell gesammelt werden müssen, bevor das Detailing stattfinden kann. Nach dem Optimieren der Bausteinkonfiguration muss das Ergebnis ebenfalls in ein Format gebracht werden, das für die folgenden Schritte verwendet und eventuell auch dem Nutzer angezeigt werden kann.

Soft items: https://arxiv.org/abs/2206.15116

Irregular Shaped items: https://link.springer.com/content/pdf/10.1631/FITEE.1400421.pdf

"Parametric Blockwall-Assembly Algorithms for the Automated Generation of Virtual Wall Mockups Using BIM"

7 Vorgehensweise

7.1 Einarbeiten in IFC

Da sich IFC in der Industrie als Standard durchgesetzt hat, ist dessen Verwendung für diese Arbeit sinnvoll. Es existieren Beispielprojekte, die zum Testen herangezogen werden können [9]. Durch die Unterstützung von Blender können nach kurzer Einarbeitung in die Projektstruktur von IFC Modellen ebenfalls schnell eigene Projekte umgesetzt werden, die den Anforderungen besser entsprechen.

7.2 Filtern relevanter Strukturen aus einem IFC Projekt

Wie bereits erwähnt führt IFC Klassen wie Wände, Türen und Fenster ein. Dadurch sollte es möglich sein, das Modell nach Objekten zu durchsuchen, welche mit für diese Arbeit relevante Klassen markiert wurden. Fenster und Türen stellen eine besondere Herausforderung dar, da ein Roboter, welcher auf einer Wand entlangfahren kann, vermutlich nicht in der Lage ist, diese Lücken in der Wand zu überspringen. Somit muss das während dem Finden eines Bauplans beachtet werden.

7.3 Diskretisieren von Strukturen mit vorgegebenen Bauteilklassen

Es muss eine Lösung gefunden werden, beliebige Wände mit einem vorgegebenen, vermutlich stark limitierten Set an Bauteilklassen abzubilden, ohne das Modell zu verändern. Da dies nicht immer möglich ist, wird für diese Arbeit in diesem Schritt davon ausgegangen, dass passende Modelle als Input geliefert werden, da das Einschränken des Nutzers im Modellierungsschritt stark von der Realität des Architekturprozesses abweichen würde. Eine Möglichkeit wäre das "Teildiskretisieren" der Strukturen und ein Ausgabeset an Volumenteilen, für welche keine passende Bauteilklasse gefunden werden konnte (etwa ein Bruchteil eines Ziegels).

7.4 Einarbeiten in und Evaluierung von Ludiwgs Dissertation

Da Ludwigs Programmcode sehr wahrscheinlich in Teilen angepasst werden muss, ist eine Evaluierung des aktuellen Stands von Nöten und in welcher Weise dem Nutzer die darin enthaltenen Prozesse zur Verfügung gestellt werden. Dabei könnte es sich entweder um einen in Blender integrierten Konverter handeln oder um eine Erweiterung des BIM-Servers. Außerdem gilt es herauszufinden, ob das Anpassen seines Algorithmus an für Roboterschwärme optimierte Baupläne den Rahmen dieser Arbeit sprengen würde und wie sich dessen Output dafür ändern müsste. Eventuell ist es ausreichend, den Roboterschwarm nicht bei der Suche nach einem geeigneten Bauplan zu berücksichtigen und diesen sogar dahingehend zu optimieren, sondern im Anschluss den Bauplan nach Möglichkeiten zur synchronen Bearbeitung mehrerer Agenten zu untersuchen.

7.5 Überführung in Ludwigs Format

Nach der Diskretisierung des Modells in eine Menge an Bauteilen, können diese in das von Ludwigs Dissertation vorgesehene Schema konvertiert werden. Dazu müssen vor allem die Beziehungen einzelner Bauteile zueinander erkannt und die Constraints der Bauteilklassen und Robotertypen beachtet werden. Im Anschluss können diese Daten nun verwendet werden, um mithilfe von Ludiwgs Dissertation einen Bauteilplan zu suchen.

7.6 Format des Ergebnisses festlegen

Hier müsssen sich Luca und ich über den Verlauf unserer Masterarbeiten immer wieder absprechen und Anforderungen vergleichen.

References

[1] *05_maurerfibel_kap-4.pdf*, https://www.kalksandstein.de/media/08_downloadcenter/05_maurerfibel_kap-4.pdf, (Accessed on 06/29/2023).

[2] *Bemessung von ziegelmauerwerk nach din en 1996-3/na:2019-12*, https://www.wienerberger.de/content/dam/wienerberger/germany/marketing/documents-magazines/instructions-guidelines/wall/DE_MKT_DOC_POR_Bemessung_Ziegelmauerwerk.pdf, (Accessed on 06/29/2023).

[3] *Bim for health and safety in construction | autodesk university*, https://www.autodesk.com/autodesk-university/article/BIM-Health-and-Safety-Construction-2017, (Accessed on 04/18/2023).

[4] *Blender.org - home of the blender project - free and open 3d creation software*, https://www.blender.org/, (Accessed on 02/16/2023).

[5] *Blenderbim add-on - beautiful, detailed, and data-rich openbim*, https://blenderbim.org/, (Accessed on 02/16/2023).

[6] *Building information modeling – wikipedia*, https://de.wikipedia.org/wiki/Building_Information_Modeling, (Accessed on 02/16/2023).

[7] *Eurocode 6: Bemessung und konstruktion von mauerwerksbauten - teil 1-1: Allgemeine regeln für bewehrtes und unbewehrtes mauerwerk*, Norm, 2012.

[8] K. Dörfler, G. Dielemans, L. Lachmayer, *et al.*, "Additive manufacturing using mobile robots: Opportunities and challenges for building construction", *Cement and Concrete Research*, vol. 158, p. 106 772, Aug. 2022, ISSN: 0008-8846. DOI: 10.1016/J.CEMCONRES.2022.106772.

[9] *Examples - ifc wiki*, https://www.ifcwiki.org/index.php/Examples, (Accessed on 02/16/2023).

[10] *Fachkräftemangel und rohstoffpreise – die deutsche bauindustrie*, https://www.bauindustrie.de/zahlen-fakten/bauwirtschaft-im-zahlenbild/fachkraeftemangel-und-rohstoffpreise, (Accessed on 03/31/2023).

[11] *Ifc formats - buildingsmart technical*, https://technical.buildingsmart.org/standards/ifc/ifc-formats/, (Accessed on 02/16/2023).

[12] IfcOpenShell, *Ifcopenshell - the open source ifc toolkit and geometry engine*, https://ifcopenshell.org/, (Accessed on 05/05/2023).

[13] *Industrial automation systems and integration — product data representation and exchange — part 21: Implementation methods: Clear text encoding of the exchange structure*, https://www.iso.org/standard/63141.html, (Accessed on 05/05/2023).

[14] *Industry foundation classes – wikipedia*, https://de.wikipedia.org/wiki/Industry_Foundation_Classes, (Accessed on 02/16/2023).

[15] *Industry foundation classes (ifc) - buildingsmart technical*, https://technical.buildingsmart.org/standards/ifc, (Accessed on 02/16/2023).

[16] buildingSMART International, *Ifc 4.3.1.0 spezification*, https://ifc43-docs.standards.buildingsmart.org/, (Accessed on 05/05/2023).

[17] buildingSMART International, *Ifc 4.3.1.0 spezification chapter 1 scope*, https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/content/scope.htm, (Accessed on 05/05/2023).

[18] *Iso - iso 16739-1:2018 - industry foundation classes (ifc) for data sharing in the construction and facility management industries — part 1: Data schema*, https://www.iso.org/standard/70303.html, norm, (Accessed on 02/16/2023).

[19] *Lego brick dimensions and measurements - christoph bartneck, ph.d.* https://www.bartneck.de/2019/04/21/lego-brick-dimensions-and-measurements/, (Accessed on 07/07/2023).

[20] *Mauerwerksverband - baulexikon*, https://baulexikon.beuth.de/MAUERWERKSVERBAND.HTM, (Accessed on 06/29/2023).

[21] *Microsoft word - constructionlifecyclemanagementwithbim_080109*, https://www.researchgate.net/profile/Yusuf-Arayici/publication/243972464_Building_information_modelling_BIM_for_Construction_Lifecycle_Management/links/54f4456c0cf2f9e34f094781/Building-information-modelling-BIM-for-Construction-Lifecycle-Management.pdf, (Accessed on 02/16/2023).

[22] L. Nägele, "Parts: Automatische programmierung in der robotergestützten fertigung", doctoralthesis, Universität Augsburg, 2021, p. 236.

[23] M. Nasrun, M. Nawi, N. Baluch, and A. Y. Bahauddin, "Impact of fragmentation issue in construction industry: An overview; impact of fragmentation issue in construction industry: An overview", DOI: 10.1051/C. [Online]. Available: http://dx.doi.org/10.1051/matecconf/20141501009.

[24] *Revit-software | bim-software | autodesk*, https://www.autodesk.de/products/revit/, (Accessed on 02/16/2023).

[25] *The open source bim collective*, https://github.com/opensourceBIM, (Accessed on 02/16/2023).

[26] *Top 10 benefits of bim in construction*, https://bim360resources.autodesk.com/connect-construct/top-10-benefits-of-bim-in-construction, (Accessed on 04/18/2023).

[27] C. Xu, J. Liu, S. Li, Z. Wu, and Y. F. Chen, "Optimal brick layout of masonry walls based on intelligent evolutionary algorithm and building information modeling", *Automation in Construction*, vol. 129, p. 103 824, Sep. 2021, ISSN: 0926-5805. DOI: 10.1016/J.AUTCON.2021.103824.