



**Institut für Software & Systems Engineering**  
Universitätsstraße 6a D-86159 Augsburg



Masterarbeit im Studiengang  
„Informatik“

# **Individualisierbarer Konstruktionsplaner mit automatischer Bauplandeduktion**

Sebastian Rossi





**Institut für Software & Systems Engineering**  
Universitätsstraße 6a D-86135 Augsburg



## **Individualisierbarer Konstruktionsplaner mit automatischer Bauplandeduktion**

Sebastian Rossi

Erstgutachter: Prof. Dr. Wolfgang Reif  
Zweitgutachter: Prof. Dr. Bernhard Bauer  
Matrikelnummer: 1475010  
Abgabe der Arbeit: 8. Januar 2023  
Betreuer: Constantin Wanninger



## **Erklärung**

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Augsburg, den 8. Januar 2023

Sebastian Rossi



## **Zusammenfassung**

Aufgrund des sich zusätzenden Fachkräftemangels in der Baubranche ist es nicht verwunderlich, dass zunehmend Bestrebungen in Richtung der Automatisierung dieses Industriebereichs angestellt werden. Möglich sind automatisierte Bauprojekte erst aufgrund der fortschreitenden Digitalisierung und Vereinheitlichung durch internationale Standards. Denn der Automatisierung liegt oftmals ein nicht zu unterschätzender, vorangehender Planungsaufwand zugrunde, welcher erst dadurch softwareseitig durchführbar wird. Diese Arbeit beschäftigt sich mit den Herausforderungen ein digitales 3D Modell eines Gebäudes zu erstellen und in einen konkreten Bauplan zu überführen, während trotz angestrebter Nutzerfreundlichkeit ein größtmöglicher Freiheitsgrad in der Modellierungsphase beibehalten wird. Dabei liegt der Fokus auf dem Errichten von Mauerwerk mithilfe von Formsteinen.



# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>1</b>
<b>2</b>	<b>Fallstudien</b>	<b>3</b>
2.1	Von 3D Gebäudeplan zum Bauplanentwurf . . . . .	3
2.1.1	Szenario Turm . . . . .	3
2.1.2	Szenario LEGO Klemmbausteine . . . . .	5
2.1.3	Szenario Fabrikgebäude . . . . .	7
2.2	Regelbasierte Bauplandeduktion aus einem Bauplanentwurf . . . . .	8
<b>3</b>	<b>Verwandte Arbeiten</b>	<b>9</b>
3.1	Projekte mit automatisierten Lege-Robotern . . . . .	9
3.2	Planungsvorgehen . . . . .	9
3.2.1	Digitale Pläne von Mauerwerk für den automatisierten Mauerbau	10
3.2.2	Errechnung optimaler Legepläne für Mauerwerk mithilfe von evolutionären Algorithmen . . . . .	11
3.2.3	Automatisierter Mauerwerksgenerator mit maschinellem Lernen und Bildern . . . . .	11
3.2.4	Automatische Generierung von virtuellen Wandvorlagen für BIM	12
<b>4</b>	<b>Grundlagen</b>	<b>15</b>
4.1	Industry Foundation Classes . . . . .	15
4.1.1	IFC 4.3.1.0 Aufbau . . . . .	15
4.1.2	IfcPropertySets und IfcQuantitySets . . . . .	16
4.1.3	Positionierung von <i>IfcProducts</i> . . . . .	17
4.1.4	IfcOpeningElement . . . . .	17
4.2	IFC und Blender . . . . .	17
4.2.1	blenderbim . . . . .	17
4.2.2	IfcOpenShell . . . . .	18
4.3	Building Information Modeling . . . . .	19
4.4	opensourceBIM . . . . .	19
4.5	Mauerwerksbau . . . . .	20
4.5.1	Maßsysteme . . . . .	21
4.5.2	Mauerwerksverband . . . . .	22
4.6	LEGO System . . . . .	24
4.7	Ontologie . . . . .	25
4.7.1	Protégé . . . . .	26
4.7.2	Owlready2 . . . . .	26

<b>5 Konzept</b>	<b>29</b>
5.1 Modellierung . . . . .	29
5.1.1 Raster . . . . .	29
5.1.2 Wandstück . . . . .	30
5.1.3 Mauerwerksverband . . . . .	31
5.1.4 Beziehungen zwischen Wandstücken . . . . .	32
5.1.5 Lösen von Beziehungen . . . . .	36
5.1.6 Öffnungen . . . . .	39
5.1.7 Wandenden . . . . .	39
5.2 Wall Detailing . . . . .	40
5.3 Regelbasierte Bauplandeduktion . . . . .	41
<b>6 Realisierung</b>	<b>45</b>
6.1 Modellierung . . . . .	45
6.2 Wall Detailing . . . . .	45
6.2.1 Konvertieren und Filtern der Daten der IFC Datei . . . . .	46
6.2.2 Anwenden des Moduls . . . . .	47
6.2.3 Kombinieren passender Wandstücke . . . . .	47
6.2.4 Finden und Lösen von Beziehungen . . . . .	49
6.2.5 Anwenden der Öffnungen . . . . .	52
6.2.6 Anwenden der Mauerwerksverbände . . . . .	52
6.2.7 Nachbarschaftsbeziehungen zwischen Bausteinen . . . . .	54
6.2.8 Export . . . . .	55
<b>7 Proof of Concept</b>	<b>57</b>
7.1 Von 3D Gebäudeplan zum Bauplanentwurf . . . . .	57
7.1.1 Szenario Turm . . . . .	57
7.1.2 Szenario LEGO Klemmbausteine . . . . .	58
7.1.3 Szenario Fabrikgebäude . . . . .	59
7.2 Regelbasierte Bauplandeduktion aus einem Bauplanentwurf . . . . .	61
<b>8 Fazit und Ausblick</b>	<b>63</b>
<b>Abbildungsverzeichnis</b>	<b>65</b>
<b>Literatur</b>	<b>67</b>

# 1 Motivation

Der stetig steigende Fachkräftemangel trifft auch das Baugewerbe. 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 [47]. Abbildung 1.1 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. Dies ist ein bekanntes Phänomen und wird als *Fragmentierungsproblem der Bauindustrie* bezeichnet [62]. Um dem entgegenzuwirken, etablieren sich bereits seit einigen Jahren



Abbildung 1.1: Während wirtschaftliches Risiko durch eine potenziell sinkende Nachfrage nach Bauaufträgen und eventuell steigender Arbeitskosten unverändert blieb oder sogar als weniger relevant bewertet wurde, ist ein deutlicher Anstieg aufgrund des vorherrschenden Fachkräftemangels und der zunehmenden Energie- und Rohstoffpreise zu erkennen.

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 ressourcensparender Bau ermöglicht werden [41][66]. Zusätzlich steigen mit der Zunahme an digitalen Informationen zu Bauprojekten auch die Möglichkeiten diese ausführlicher zu analysieren, zu optimieren und an neue Technologien zu knüpfen. Erst dadurch wurde das seit einigen Jahren erforschte Gebiet der *Additiven Fertigung* von Gebäuden realisierbar. Einige erfolgreiche Beispiele

dafür sind in [21], [31] und [26] aufgezeigt und wurden etwa mit Beton druckenden Roboterarmen, mobilen Robotern oder aufgehängten Druckköpfen durchgeführt. Dabei werden teils herkömmliche, teils speziell für die druckenden Roboter entwickelte Materialien verwendet [26]. Obwohl es mittlerweile viele Projekte zur Additiven Fertigung von Gebäuden gibt, haben diese oft den Nachteil der Nicht-Parallelisierbarkeit der druckenden Roboter und die daraus resultierende, vergleichsweise lange Bauzeit. Auch die durch die Höhe der temporären Stützstrukturen (wie Kräne, Gerüste oder Aufhängungen) eingeschränkte Bauhöhe limitiert die Vielfalt der mit additiver Fertigung realisierbaren Projekte. Diesen Einschränkungen soll nun mithilfe eines Schwarmes bodengebundener autonomer Roboter, die gleichzeitig an dem Bauprojekt arbeiten können, entgegengewirkt werden. Dabei sollen sich die Roboter auf den Mauern des Gebäudes selbst bewegen können, während sie diese errichten. Im Gegensatz zur additiven Fertigung sollen die Roboter das Material nicht etwa drucken, sondern herkömmliche Bausteine verwenden können. In dieser Arbeit liegt der Schwerpunkt allerdings nicht auf der Entwicklung der Lege-Roboter, sondern auf dem Erarbeiten eines Vorgehens zur Berechnung von Bauplänen, die für Roboterschwärme geeignet sind. Ausgangspunkt hierfür sind 3D Modelle der Gebäude. Dies stellt die Grundlage für nachfolgende Automatisierungsprojekte im Bereich der Bauindustrie dar. Das Anfertigen der Modelle innerhalb eines 3D-Konstruktionsplaners ermöglicht nicht nur das Definieren geometrischer und physikalischer Eigenschaften eines Gebäudes in digitaler Form, sondern verschlankt zudem die Kommunikation zwischen Endnutzer und Architekturbüro oder ersetzt letzteres komplett. Gleichzeitig macht diese Arbeit damit einen Schritt in Richtung des relativ neuen Trends der sogenannten Massenpersonalisierung, welcher als Nachfolgetrend zur Massenproduktion und als „heiliger Gral“ der Fertigung angesehen wird [45]. Dieser Trend ist ebenfalls für die Bauindustrie interessant, denn auch hier schafft die Möglichkeit sämtliche Kundenwünsche an ein Produkt (oder in diesem Fall ein Gebäude) umzusetzen, ohne dafür spezielles Werkzeug herstellen oder Verfahren entwickeln zu müssen, neue Gewinnmöglichkeiten [22][15]. Dennoch existiert noch vergleichsweise wenig Forschung, die teilweise schon etablierten Konzepte der Massenpersonalisierung aus der Fertigungsindustrie ebenfalls in der Bauindustrie zu erproben [23]. Darum soll in dieser Arbeit untersucht werden, in welcher Weise sich bereits vorhandene digitale Standards und Austauschformate dafür eigenen, individuelle Baupläne aus den 3D-Plänen von Gebäuden zu generieren. Die Baupläne können dabei aufgrund der Anbindung an einen frei verfügbaren 3D Editor nicht nur von Experten, sondern ebenfalls von Laien stammen. So kann ein Endnutzer persönliche Wünsche selbst in das Modell integrieren. Zudem wird nach einer Möglichkeit gesucht, die resultierenden Baupläne durch adaptive Regelsets an die jeweilige Situation anpassen zu können, denn unterschiedliche Bauprojekte besitzen oft unterschiedliche Eigenheiten und Prioritäten. Diese Konzepte werden daraufhin anhand nachfolgender Fallstudien getestet.

## 2 Fallstudien

Anhand der nachfolgenden Fallstudie und deren Szenarien werden die dieser Arbeit zugrundeliegenden Konzepte zur Errechnung eines Bauplanentwurfs aus einem als 3D Modell vorliegenden Gebäudeplans veranschaulicht und auf Anwendbarkeit überprüft. Dabei werden die Szenarien zunehmend komplexer, um auch das Zusammenspiel verschiedener Teilkonzepte zur Lösung einzelner Probleme zu verifizieren. Abschließend wird in einer weiteren Fallstudie untersucht, wie Regeln erstens definiert und zweitens auf den Ergebnissen der ersten Fallstudie angewandt werden können. Damit soll die ungeordnete Menge der konkreten Bausteine eines Bauplanentwurfs durch Regeln in eine Reihenfolge gebracht werden, sodass daraus ein schrittweise umsetzbarer Bauplan entsteht.

### 2.1 Von 3D Gebäudeplan zum Bauplanentwurf

Diese erste Fallstudie enthält drei Szenarien anhand derer die Konzepte zur Errechnung von Bauplanentwürfen ausgehend von 3D Gebäudeplänen erläutert und getestet werden. Die Pläne sind das Ergebnis der Gebäudemodellierung innerhalb eines Konstruktionsplaners. Die Modellierung soll mithilfe der in Kapitel 4 näher behandelten Technologien geschehen, weshalb die Gebäudepläne in ihrer Struktur einem verbreiteten Industriestandard entsprechen.

#### 2.1.1 Szenario Turm

In diesem einleitenden Szenario werden mithilfe eines einfachen Turmes mit vier Wänden einige der Kernkonzepte geprüft. Dabei werden sowohl die Modellierung des Turmes innerhalb des Konstruktionsplaners, als auch das anschließende Errechnen des Bauplanentwurfs thematisiert. Insbesondere das Anwenden verschiedener Mauerwerksverbände soll die Flexibilität des erarbeiteten Vorgehens demonstrieren.

#### Beschreibung

Der Turm besteht lediglich aus vier 20 Meter hohen Wänden, die einen einzigen Raum einschließen. Er hat einen Grundriss von  $10 \times 10$  Metern. Die Wände sollen unter Anwendung folgender Mauerwerksverbände realisiert werden:

- Einem Läuferverband mit einem Versatz von 50 % der Bausteinlänge.
- Einem Kopf/Binderverband.
- Einem Kreuzverband.

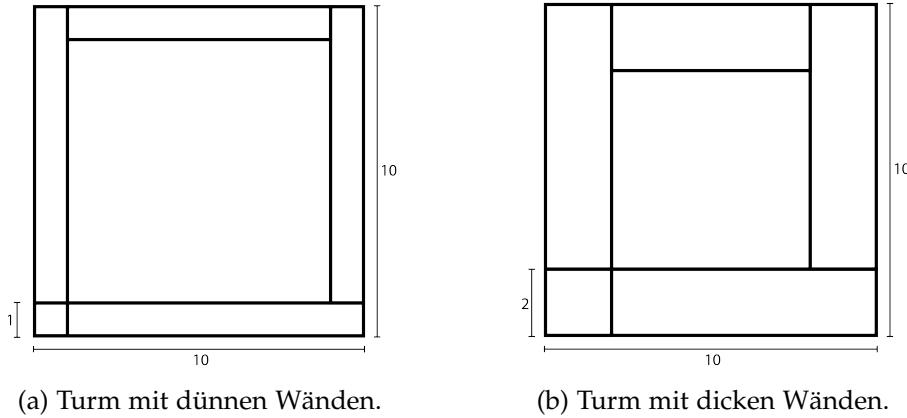


Abbildung 2.1: Grundrisse der beiden Türme.

Da die beiden letzteren Verbände bei gleichbleibendem Modul die Wanddicke verdoppeln, muss das Modell etwas angepasst werden, um nach wie vor denselben Grundriss aufzuweisen. Daher gibt es sowohl einen Bauplan mit 1 Meter dicken als auch einen mit 2 Meter dicken Wänden. Dies ist in Abbildung 2.1 zu sehen. Als Basismodul wird ein Baustein mit den Maßen  $2 \times 1 \times 0.5$  Metern verwendet. Das vereinfacht die Interpretation der generierten Lösungen. Das dazugehörige Raster hat eine Größe  $0.5 \times 0.5 \times 0.5$  Metern.

### Problemstellungen

Die Komplexität des Modellierungsvorgangs nimmt durch die Integration eines umfangreichen Industriestandards deutlich zu. Das liegt vor allem an dem immensen Abstraktionsgrad des Standards. Dieser ist notwendig, um möglichst alle in Realität vorkommenden Szenarien abzubilden. Gleichzeitig existiert aufgrund der Vielzahl der im Bauwesen beteiligten Akteure und Fachbereiche eine große Informationsfülle, die ebenfalls Teil davon ist. Dieser Standard wird im Detail in Kapitel 4.1 behandelt. Um der Komplexität entgegenzuwirken, werden Ansätze gesucht, die diese Phase dennoch intuitiv und einsteigerfreundlich zu gestalten.

Obwohl sich viele Wandbereiche ausschließlich mit einem am Grundmodul angepassten Bausteinformat errichten lassen, ist es in Realität zum Beispiel an Wandenden oder Ecken unumgänglich, Bausteine durch Zerschneiden zu verkleinern. Denn in diesen besonderen Bereichen werden oft Bausteine mit geringerer Länge benötigt, um gerade und lückenlose Übergänge umzusetzen. Darum ist es notwendig, das vorgegebene Bausteinformat während der Errechnung des Bauplanentwurfs gegebenenfalls flexibel anpassen zu können. Hieraus ergibt sich bereits die nächste Problemstellung: Wie können derartige Bereiche zwischen einzelnen geraden Wandstücken gefunden und der jeweilige Mauerwerksverband auch an diesen Stellen passend angebracht werden? Dabei dürfen das Überbindemaß (siehe Kapitel 4.5.2) oder andere Vorgaben allerdings nicht verletzt werden. In Abbildung 2.1 sind verschiedene Konstellationen von Wänden erkennbar, die zusammen eine Ecke bilden. Demnach muss das Konzept dieser Arbeit, trotz unterschiedlicher Möglichkeiten der Eck-Modellierung innerhalb des Konstruktionsplaners,

jede dieser Situationen auffinden und im Anschluss passend verarbeiten können. Ein Fokus dieser Arbeit liegt zudem auf der simultanen Unterstützung verschiedener Mauerwerksverbände innerhalb eines Modells. Darum muss ein generisches Format für die dafür relevanten Informationen entworfen werden, das eine Interpretation durch Algorithmen erlaubt. Außerdem wird durch diese Abstraktion das spätere Einpflegen weiterer Verbände leicht möglich.

Im letzten Schritt muss ein Datenformat für das Ergebnis des erarbeiteten Konzepts entwickelt und darüber entschieden werden, welche Informationen Teil des resultierenden Bauplanentwurfs sein müssen. Damit wird eine einheitliche Schnittstelle definiert, welche eventuellen Folgeprojekten die Anbindung an diese Arbeit erleichtert. Zusammengefasst birgt dieses erste Szenario demnach folgende Teilprobleme:

- Wie lässt sich eine intuitive Modellierungsphase ermöglichen?
- Wie können Bausteininformate definiert und gegebenenfalls verändert werden?
- Wie können einem Algorithmus die verschiedenen Mauerwerksverbände sinnvoll vorgegeben werden?
- Wie können Eckbereiche gefunden und der jeweilige Mauerwerksverband auch dort passend angebracht werden?
- Wie sieht das Ergebnis des erarbeiteten Konzepts aus und welche Informationen werden in den resultierenden Bauplanentwurf integriert?

### 2.1.2 Szenario LEGO Klemmbausteine

Nun folgt ein Szenario, das den Gegebenheiten eines realen Gebäudes bis auf dessen Skalierung eher entspricht. Es soll ein LEGO-Gebäude konstruiert werden, welches einen Innenraum, Fenster und Türen enthält. Darum wird das Modul stark verkleinert, sodass es den Maßen eines  $4 \times 2$  LEGO Steins entspricht. Mehr Informationen darüber werden in Kapitel 4.6 bereitgestellt.

#### Beschreibung

In Abbildung 2.2 ist der Plan eines einfachen Hauses mit einem Stockwerk zu sehen. Dessen Grundfläche beträgt 19.2 auf 14.4 Zentimeter. Das Haus besitzt eine Eingangstür, drei Fenster und eine weitere Tür, die das Badezimmer vom Hauptraum trennt. Es gibt außerdem breite Außen- und dünne Innenwände. Dafür müssen zwei Wandtypen definiert werden, die jeweils unterschiedliche Wanddicken vorgeben. Diese entsprechen in ihren Maßen dem Raster, welches das LEGO System vorgibt. Dickere Wände sollen zwei Noppen breit sein. Daher wird für diesen Wandtyp ein Grundmodul mit den Maßen  $31.8 \times 15.8 \times 9.6$  Millimetern und einem Raster von  $8 \times 8 \times 9.6$  Millimetern gewählt. Das entspricht dem oben genannten  $4 \times 2$  LEGO Stein. Für die dünneren Innenwände soll eine Breite von einer Noppe verwendet werden. Daher wird hierfür der  $2 \times 1$  LEGO Stein als Grundmodul vorgegeben. Dieses hat demnach die Maße  $15.8 \times 7.8 \times 9.6$  Millimeter mit gleichbleibendem Raster. Mögliche Rotationen von Fenstern, Türen und Wänden sind

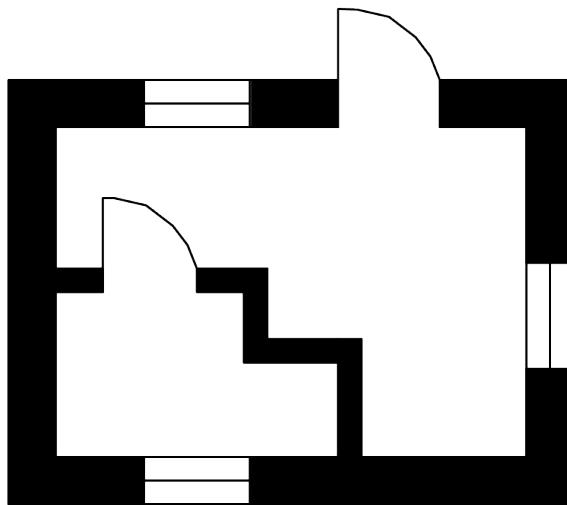


Abbildung 2.2: Grundriss des LEGO-Gebäudes mit einer Grundfläche von 19.2 auf 14.4 Zentimetern.

auf 90° Schritte limitiert. Das stellt im Fall von LEGO eine vertretbare Einschränkung dar, da es ohnehin dem intuitiven Umgang mit dessen Steinen und gleichzeitig dem Baustil der meisten einfachen Gebäuden entspricht.

### Problemstellungen

Zusätzlich zu den Teilproblemen des vorhergehenden Szenarios ergeben sich nun weitere Fragestellungen. Türen und Fenster stellen eine Herausforderung für den Planeralgorithmus dar, da der Verlauf einer ansonsten durchgängigen Wand dadurch unterbrochen wird und Lücken aufweist. Diese „Lücken“ werden nachfolgend als Öffnungen bezeichnet. Öffnungen müssen zunächst im Konstruktionsplaner modelliert werden können, um im Anschluss durch den Planeralgorithmus weiter verarbeitet zu werden. Darum wird ein Konzept für Öffnungen benötigt, das die Definition von Teilbereichen einer Wand ermöglicht, die nicht mit Bausteinen aufgefüllt werden dürfen. Wie oben bereits erwähnt, können Wände verschiedene Dicken aufweisen und zusätzlich unter Anwendung unterschiedlicher Grundmodule und Mauerwerksverbände realisiert werden. Auch diese Eigenschaften müssen sowohl innerhalb des Konstruktionsplaners, als auch im Planungskonzept dieser Arbeit berücksichtigt werden. Auffällig in diesem Szenario ist auch das Auftreten von T-Kreuzungen zwischen Innen- und Außenwänden. T- und X-Kreuzungen müssen, ähnlich zu den bereits besprochenen Eckbereichen, zunächst identifiziert und im Anschluss gesondert behandelt werden, da an den Stellen das Einhalten eines bestimmten Mauerwerksverbands besonders komplex ist. Auch der Übergang zwischen Wänden unterschiedlicher Dicke oder Verband stellt eine weitere Herausforderung dar. Zusammengefasst ergibt sich durch dieses Szenario folgende neue Liste an Teilproblemen:

- Wie werden Öffnungen im Konstruktionsplaner umgesetzt und im Anschluss weiter verarbeitet?

- Wie können einer Wand Dicke, Grundmodul und Mauerwerksverband zugewiesen werden?
- Wie können Übergänge zwischen Wänden verschiedener Dicke oder unterschiedlichem Verband realisiert werden?
- Wie können T- und X-Kreuzungen von Eckbereichen unterschieden und dem Mauerwerksverband entsprechend umgesetzt werden?

### 2.1.3 Szenario Fabrikgebäude

Um das Konzept auch für umfangreichere Projekte zu testen, wird für dieses Szenario ein Modell eines großen Fabrikgebäudes erstellt. Dessen Aussehen ist angelehnt an dem Stil der Industriearchitektur. In Abbildung 2.3 ist das geplante Modell der Fabrikhalle zu sehen. Alle in den Öffnungen liegenden Fenster und Türen sind zur besseren Veranschaulichung ausgeblendet worden.

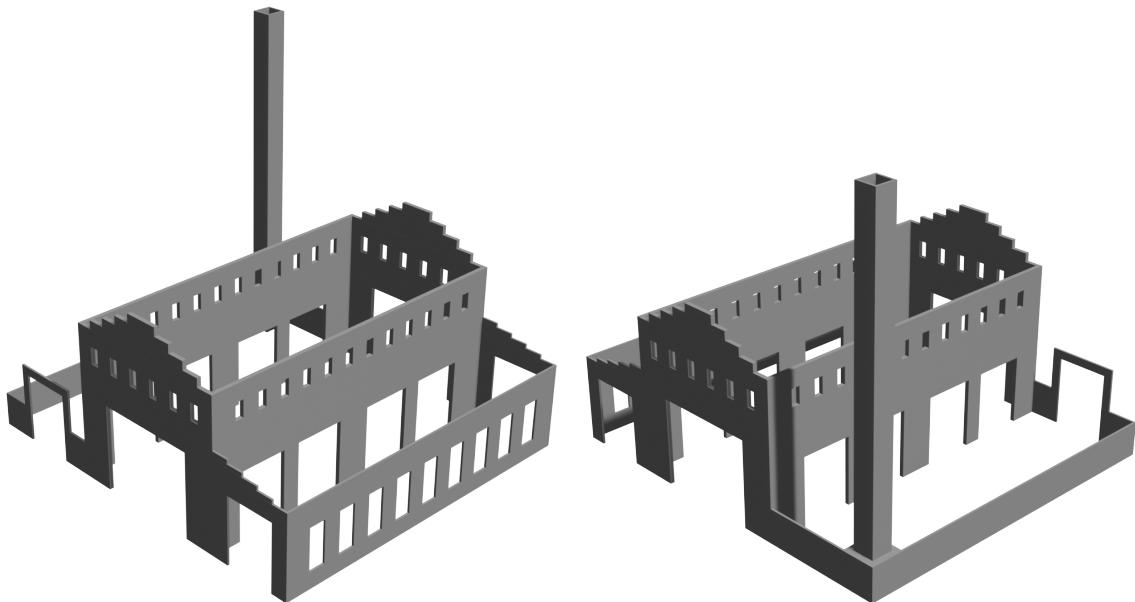


Abbildung 2.3: IFC Modell des Fabrikgebäudes aus zwei Perspektiven.

#### Beschreibung

Die dabei verwendeten Baustein- und Wanddimensionen entsprechen den Größenordnungen, die von dem sogenannten oktametrischen Maßsystem vorgegeben werden. Informationen dazu sind in Kapitel 4.5.1 zu finden. Es werden drei verschiedene Wandtypen verwendet. Diese weisen jeweils eine Dicke von 0.125, 0.25 und 0.5 Metern auf, während dabei Grundmodule in den Größen eines 1 DF, 2 DF und 16 DF Steines angewandt werden sollen. Zudem soll bei jedem der drei Wandtypen ein unterschiedlicher Mauerwerksverband benutzt werden. Bei der dünsten Wand ein einfacher, halbversetzter Läuferverband. Bei der 0.25 Meter dicken Wand ein Kreuzverband und für die Wand,

mit dem 16 DF Steinen, ein Kopfverband. Letztere findet lediglich als Sockel des Kamins Anwendung, während die anderen beiden Wandtypen häufiger vertreten sind. Insgesamt weist das eingeschlossene Gelände eine Grundfläche von 20 auf 22 Metern auf.

### **Problemstellung**

In diesem Szenario soll vor allem die Applikabilität der erarbeiteten Konzepte auf großen Datenmengen und dadurch die Effizienz der Implementierung auf die Probe gestellt werden. So können die Teilbereiche der Implementierung dieser Arbeit identifiziert werden, für die eventuell noch Optimierungsbedarf besteht und grundlegende Fehler in den Konzepten aufgespürt werden.

## **2.2 Regelbasierte Bauplandeduktion aus einem Bauplanentwurf**

In dieser Fallstudie wird untersucht wie das Integrieren von Regeln in das erarbeitete Konzept ermöglicht werden kann. Mit den Regeln sollen etwa Abhängigkeiten zwischen Bausteinen vorgegeben oder die maximale Bauhöhe beschränkt werden können. Dies wird anhand des in Abbildung 2.4 dargestellten Bauplanentwurfs getestet. Da-

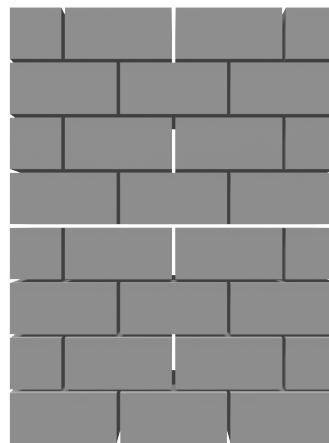


Abbildung 2.4: Bauplanentwurf des Experiments zur Bauplandeduktion unter Berücksichtigung von Regeln.

bei ist es wünschenswert Regeln erst im Nachgang an den vorangegangenen Schritt der Berechnung eines Bauplanentwurfs definieren zu können. Darum dürfen konkrete Regelbeschreibungen nicht Teil der Programmcodebasis dieser Arbeit sein, sondern müssen auf andere Weise in das Projekt integrierbar sein. Zur Informationsdefinition und -strukturierung existieren im Fachbereich der Informatik bereits potente Werkzeuge. Eines davon wird in Kapitel 4.7 thematisiert und im späteren Verlauf dieser Arbeit zur formalen Definition der genannten Regeln angewandt. Die Herausforderung für den Planungsalgorithmus stellt daher das korrekte Interpretieren und Anwenden dieser Regeln dar.

## 3 Verwandte Arbeiten

### 3.1 Projekte mit automatisierten Lege-Robotern

Die Automatisierung der Baubranche hat neben Faktoren wie Geschwindigkeit, Effizienz und Kostenreduzierung auch den Vorteil Bauprojekte an Orten zu realisieren, die für Menschen ungeeignet sind [8]. Dafür wurde unter anderem das Projekt *TERMES* von Kirstin Petersen et al. konzipiert, mit welchem das Team, angelehnt an dem Verhalten von Termiten, einen Schwarm bodengebundener Roboter koordiniert, die als Einheit Strukturen errichten [8]. Darüber hinaus stellen Kirstin Petersen et al. in einer anderen Veröffentlichung eine umfangreiche Übersicht von weiteren Projekten aus dem Bereich *Collective Robotic Construction* vor [24]. Doch nicht nur vollautomatische Roboterschwärme haben das Ziel die Baubranche zu verändern. Es gibt bereits etablierte Firmen, die semi-automatische Roboteranlagen zur Unterstützung der Arbeiter vor Ort anbieten und damit erfolgreich die Bauzeit reduzieren können. Dazu zählt zum Beispiel *Hadrian X* der Firma FBR: ein mobiler, bodengebundener Kran der in der Lage ist, ausgehend von einem CAD Modell, ein bis zu drei Stockwerke hohes Gebäude aufzuschichten [34]. Ein weiteres Beispiel stellt *SAM100* dar, ein mobiler, Ziegelstein legender Roboter der Firma *Construction Robotics* [28]. Dieser kann sich sogar erhöht auf einem Gerüst bewegen und damit in jedem Stockwerk zum Einsatz kommen.

### 3.2 Planungsvorgehen

Grundlage für sämtliche Projekte, in welchen mithilfe von Lege-Robotern automatisiert Gebäude beziehungsweise Gebilde mit Formsteinen errichtet werden, ist eine vorangehende Planungsphase. In dieser Planungsphase müssen die notwendigen Informationen zu allen benötigten Bausteinen gesammelt werden, um im Anschluss die Bauphase überhaupt zu ermöglichen. Das sogenannte *Wall Detailing* stellt ein ähnliches Problem wie das *3D Bin Packing Problem* dar und errechnet, ausgehend von einem 3D Modell eines Gebäudes und Informationen zu den gewünschten Mauerwerksverbänden und Bausteinformen, einen möglichen Bauplan für das gesamte Gebäude. Unter dem *Bin Packing Problem* (im Deutschen als Behälterproblem bekannt) versteht man das möglichst platzsparende Schichten von Objekten in einen oder mehrere limitierte Behälter. Dies kann sowohl im zwei- als auch im dreidimensionalen Raum geschehen. Besonders herausfordernd ist das Verteilen unregelmäßig geformter Objekte wie es etwa Xiao Liu et al. am Beispiel verschiedener geometrischer Formen oder Qiruyi Zuo et al. am Beispiel von Obstkisten zeigen [16] [32]. Während sich in diesem Bereich viel mit unregelmäßig geformten Objekten beschäftigt wird, kann im Falle von Wänden, die mit Bausteinen errichtet werden, eine Einschränkung auf quaderförmige Objekte vorgenommen werden. Zudem

ist das Ziel des *Wall Detailings* nicht eine möglichst gute, sprich eine möglichst lückenlose Lösung zu finden, sondern eine, die sinnvoll aufgeschichtete Wände hervorbringt. Dabei stellen vor allem Übergänge zwischen mehreren Wandstücken besonders kritische Bereiche dar, für die es vor Ort oft Expertenwissen benötigt. Zusätzlich muss Mauerwerk in Deutschland und Europa einigen Normen entsprechen. Darauf wird in Kapitel 4.5 näher eingegangen. Nachfolgend werden aber zunächst einige Veröffentlichungen vorgestellt, die ein ähnliches Ziel wie diese Arbeit verfolgen.

#### 3.2.1 Digitale Pläne von Mauerwerk für den automatisierten Mauerbau

In diesem wissenschaftlichen Papier stellen Usmanov et al. ein generelles Vorgehen für das Erstellen eines Ziegel-Legeplans für ein als digitales Modell vorliegendes Gebäude vor [29]. Dieses Vorgehen gliedern sie in sechs Schritte:

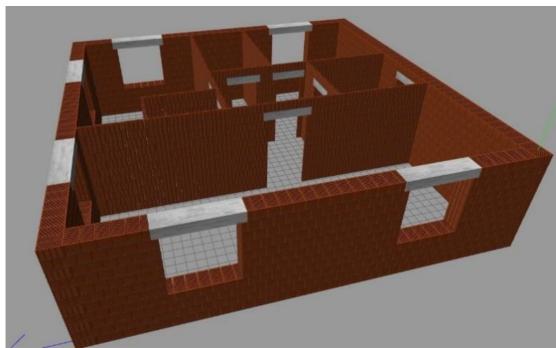


Abbildung 3.1: Ergebnis des Verfahrens zur Erstellung eines Ziegel-Legeplans nach Usmanov et al. [29].

1. Das vorliegende IFC Modell (siehe 4.1) nach Wandelementen durchsuchen und diese in das sogenannte BREP-Format konvertieren.
2. Das Aufteilen des gesamten Modells in Schichten, die der Modulhöhe des verwendeten Ziegelsteinformats entspricht.
3. Verbindungen von getrennt modellierten Wandelementen heraussuchen. Dies ist zum Beispiel an Eckstücken der Fall, da dafür oft zwei einzelne Wandelemente modelliert werden, welche in einem Winkel zueinander stehen und sich berühren. Für die nachfolgenden Schritte sind diese Verbindungen relevante Informationen.
4. Mit den Informationen der vorangegangenen Schritte können nun für jede Schicht kritische Bereiche identifiziert werden, an welchen später ein komplexer Legevorgang von Ziegeln vonnöten ist.
5. Nun werden zunächst die kritischen Bereiche anhand einer vorher definierten Legeanleitung mit teilweise angepassten Ziegelsteinen bestückt und im Anschluss die restlichen Bereiche aller Wände mit dem ausgewählten Standardziegel aufgefüllt. In diesem Schritt werden zusätzlich Fenster- und Türstürze über deren Öffnungen in den Wänden gelegt.

6. Dieser Schritt fasst das Anzeigen als 3D Modell und Konvertieren des Resultats in eine nicht konkreter definierte Listenform zusammen. Das Ergebnis für das von den Autoren ausgewählte Beispielgebäude ist in Abbildung 3.1 zu sehen.

Besonders detailliert ist ihr Ansatz für das Finden von besagten kritischen Teilbereichen einer Wand mithilfe mathematischer Gleichungen, die alle Wände zueinander in Beziehung stellen. Diese Teilbereiche sind Wanddecken, T-Kreuzungen (wie etwa der Übergang einer Innenwand an eine Außenwand) und Öffnungen innerhalb einer Wand und sind ebenfalls in Kapitel 2.1.2 dieser Arbeit als Fallbeispiel aufgeführt. Anhand der von ihnen zusammengetragenen Informationen konnten die Autoren erfolgreich die bestmögliche Platzierung von vier Roboterarmen errechnen, die den schnellstmöglichen Bau des Gebäudes ermöglichen. Dennoch werden zum Schluss noch einige Einschränkungen ihres Verfahrens angesprochen. Vor allem die Beschränkung auf 90° Ecken und das Gebunden sein an einen einzigen Standardziegel werden als besonders restriktiv wahrgenommen. Außerdem fehlt die Unterstützung anderer Mauerwerksverbände.

### **3.2.2 Errechnung optimaler Legepläne für Mauerwerk mithilfe von evolutionären Algorithmen**

Xu Chengran et al. haben in dieser Veröffentlichung verschiedene Optimierungsansätze aus dem Bereich des 2D Packaging Problems getestet [30]. Konkret wurden drei Algorithmen verwendet: Differential Evolution, Particle Swarm Optimization und Neighbourhood Field Optimization. Außerdem wird ein dreiphasiges Vorgehen vorgeschlagen: *Data collection*, *Brick layout* und *Data Output*. Dieses Vorgehen ähnelt dem von Usmanov et al. und eignet sich ebenfalls für das Konzept dieser Arbeit, da zuerst alle relevanten geometrischen Daten aus dem 3D Modell gesammelt werden müssen, bevor das eigentliche *Wall Detailing* stattfinden kann. Nach dem Optimieren der Bausteinkonfiguration muss das Ergebnis ebenfalls in ein Format gebracht werden, das für die nachfolgenden Schritte verwendet und eventuell auch dem Nutzer angezeigt werden kann. Dennoch werden auch in dieser Veröffentlichung lediglich zwei verschiedene Mauerwerksverbände betrachtet, die zwar von den Optimierungsalgorithmen umgesetzt werden konnten, aber nur an geraden Wandstücken getestet wurden. Das Anwenden der Lösungsalgorithmen auf Eck- und Kreuzungsbereiche stellt einen sinnvolleren Anwendungsfall dar, als damit das *Wall Detailing* gerader Wandabschnitte zu lösen. Letzteres kann, was mit dieser Arbeit gezeigt wird, ebenso iterativ berechnet werden. Dadurch werden potenzielle Fehler der Optimierungsalgorithmen verhindert. Das Lösen der kritischen Bereiche entspricht hingegen eher einem Problem, das mit „intelligenten“ Optimierungsverfahren bewältigt werden kann.

### **3.2.3 Automatisierter Mauerwerksgenerator mit maschinellem Lernen und Bildern**

Bárbara Andrade Zandavali et al. untersuchen in ihrem Paper wie gut sich Machine Learning Algorithmen dazu eignen, anhand von Bildern Pläne für konkretes Mauerwerk zu erzeugen [25]. Ein Kernaspekt dabei war das Anwenden verschiedener Mauerwerksverbände auf arbiträre Vielecke, welche die Form eines Wandabschnitts beschreiben.

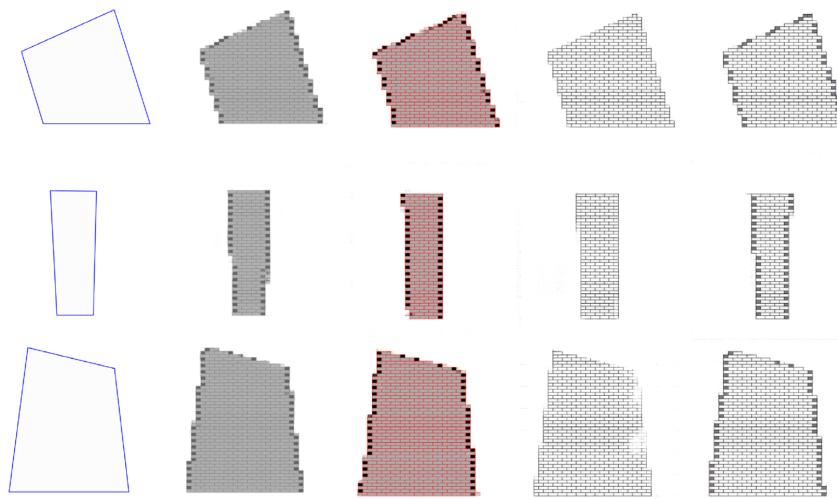


Abbildung 3.2: Eingaben und Ergebnisse des *Brick Pattern Generator* nach Bárbara Andrade Zandavali et al. [25].

In Abbildung 3.2 sind verschiedene Eingabeformen (linke Spalte) und deren Ergebnisse in vier verschiedenen Ausgabearten zu sehen. Da es sich sowohl bei dem Eingabe- als auch bei dem Ausgabeformat um Bilder handelt, musste im Anschluss zunächst untersucht werden, wie gut sich die verschiedenen Ergebnisarten dazu eignen daraus tatsächliches Mauerwerk zu interpretieren. Zusammenfassend konnten die Autoren berichten, dass das Vorgehen teilweise zu fragwürdigen Resultaten geführt hat, welches sich aber mit Sicherheit durch Anpassungen von Parametern des Algorithmus verbessern lässt. Dennoch ist der Anwendungsfall gerader Wandabschnitte schlecht gewählt, da sich dieselben Ergebnisse auch durch „Ausstanzen“ der Vielecke aus einer zuvor mit Bausteinen aufgefüllten, rechteckigen Wand erreichen lassen. Das Anwenden intelligenter Mechanismen wie Machine Learning zum automatischen Lösen der kritischen Eck- und Kreuzungsbereiche zwischen mehreren Wandstücken kann sich allerdings, wie bereits erwähnt, durchaus als nützlich erweisen.

### 3.2.4 Automatische Generierung von virtuellen Wandvorlagen für BIM

Tarek Zaki et al. beschreiben in dieser Veröffentlichung ein Vorgehen, das es ermöglicht die Werkstatt- und Montageplanung für ein Bauprojekt, das in einer BIM-fähigen Umgebung (siehe Kapitel 4.3) geplant wird, direkt innerhalb derselben Umgebung durchführen zu können [20]. Dies wurde bisher von Hand als zusätzliches sogenanntes *Level of Detail* (LOD) berechnet und im Anschluss wieder in das Projekt integriert. Das angestrebte LOD ist eine Art Mockup und siedelt sich einen Schritt vor dem letzten LOD, dem Baubestandsplan, an. Baubestandspläne stellen eine exakte Repräsentation der zu bauenden Elemente dar. Das Ziel der Veröffentlichung ist, dieses LOD direkt in der BIM Umgebung zu berechnen und anzuzeigen, da darin die dafür notwendigen Informationen ohnehin vorhanden sind. Insgesamt soll dies eine genauere Materialanalyse und Kostenplanung ermöglichen. Da es sich aber „nur“ um ein Mockup handelt, werden zum Beispiel Eckbe-

reiche nicht berücksichtigt, sondern durch sich überlappende Wandstücke und Bausteine realisiert. Dafür kann jedoch ein beliebiges Basismodul vorgegeben werden, das nach Erstellen des Mockups bereits offensichtliche Fehler sichtbar macht. So überschneidet sich etwa ein deplatziertes Fenster mit dem Mauerwerk, wenn sich dieses nicht auf dem durch das Grundmodul vorgegebene Raster befindet. Dennoch soll das Ergebnis dieser Arbeit im Gegensatz dazu ein konkreter Bauplanentwurf sein, der direkt in die Realität überführt werden kann; sei es von Robotern oder Menschen oder einer Kombination aus beiden.



# 4 Grundlagen

Zunächst 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* möglichst vollständig digital abbilden zu können. Hilfreich ist dabei, wichtige Daten über bestimmte Bestandteile des Gebäudes direkt in das Modell zu integrieren, auf Basis derer etwa Kostenberechnungen durchgeführt oder Materialmengen abgeschätzt werden können. Da diese Informationen oft von Experten verschiedener Fachbereiche (etwa aus den Bereichen der Architektur, des Bauwesens oder der Statik) stammen, muss das Format flexibel und im besten Fall auch zeitgleich bearbeitbar sein. Dafür werden seit dem Jahr 2000 die *Industry Foundation Classes* (IFC) von buildingsmart entwickelt, deren Anwendung im internationalen Bauwesen mittlerweile weit verbreitet ist [53].

## 4.1 Industry Foundation Classes

In der Spezifikation des Standards selbst, wird dieser wie folgt beschrieben (aus dem Englischen): „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 Infrastruktur-Typen umfassen Brücken, Straßen, Eisenbahnen, Wasserstraßen und Hafenanlagen“ [57]. Eine frühere Version des IFC Standards ist unter der Bezeichnung ISO 16739 registriert (siehe [58]). Da der IFC aber nach wie vor kontinuierlich weiterentwickelt wird, wird in dieser Arbeit die derzeit neueste Version verwendet. Diese ist die IFC Spezifikation 4.3.1.0 [56]. Das verbreitetste Austauschformat für IFC ist das *Step Physical File Format*, welches im ISO 10303 Teil 21 registriert ist [51]. Zudem gibt es speicherreduzierte Formate wie ifcZip oder für Menschen lesbare Formate wie ifcXML [52][49][46].

### 4.1.1 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 Hierarchie konkretisieren. 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 4.1 dargestellt. Objekte werden von dem Standard in Relation zueinander gestellt, um komplexe

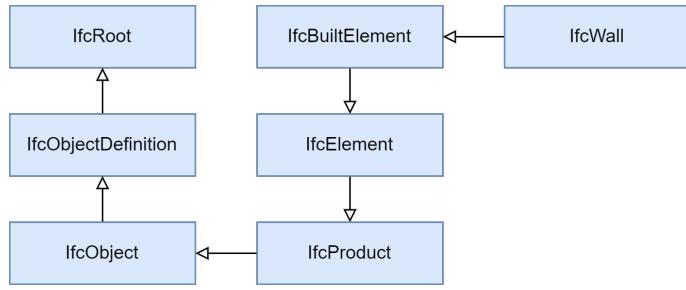


Abbildung 4.1: Klassenhierarchie am Beispiel der Klasse *IfcWall*.

Zusammenhänge darzustellen. In Abbildung 4.2 erkennt man den Zusammenhang zwischen einem Objekt des Typs *IfcWall*, des Stockwerks, welches diese Wand referenziert und wiederum selbst Teil eines *IfcBuildings* ist, bis hin zur obersten Komponente eines IFC Projektes, dem gleichnamigen *IfcProject*.



Abbildung 4.2: Relation der *IfcWall* und einem *IfcProject*.

### 4.1.2 IfcPropertySets und IfcQuantitySets

Mit dem Erweitern des Gebäudemodells um möglichst viele Informationen, schafft man einen detaillierten digitalen Zwilling, der neben der bloßen Darstellung des Gebäudes als 3D Modell zusätzlich die Untersuchung auf andere Eigenschaften ermöglicht. Dazu zählen zum Beispiel eine präzisere Kosten- und Zeitschätzung, Sicherheitsaspekte und Emissionen eines Bauvorhabens [52] [11]. Dies ist mithilfe von *IfcPropertySets* und *IfcQuantitySets* umsetzbar, welche Objekten oder Objekttypen angehängt werden können. Dabei werden *IfcQuantitySets* vorwiegend dazu verwendet numerische Werte zu geometrischen oder physikalischen Eigenschaften auszudrücken. *IfcPropertySets* dienen hingegen dazu bestimmten Objekten oder Objekttypen mit sämtlichen nicht numerischen Werten zu annotieren. Beispiele dafür sind etwa Material oder Farbe eines bestimmten Objekts. Die Verwendung der Bezeichnung *Set* röhrt daher, dass diese Informationen in einer Baumstruktur vorliegen und demnach verschachtelt sein können. Es gibt zu

vielen Klassenbeschreibungen des IFC Standards vordefinierte IfcPropertySetTemplates. Das hilft dabei wichtige Informationen zu bestimmten Objekttypen einheitlich angeben zu können. Im Falle des *IfcWallType* existiert beispielsweise das IfcPropertySet mit dem Namen *Pset\_WallCommon* [55]. Darin können die Ersteller von neuen WallTypes unter anderem Informationen über Brennbarkeit, thermisches Verhalten oder Akustik hinterlegen. Bei Erzeugung einer IfcWall Instanz aus einem bestimmten WallType sind die damit verbundenen Property- und QuantitySets automatisch damit verknüpft, sodass jedes Objekt relevante Informationen über sich selbst bereithält. Dies ermöglicht eine effiziente Inspektion des Gebäudemodells.

#### 4.1.3 Positionierung von *IfcProducts*

Die Platzierung eines *IfcProducts* wird durch eine Verkettung relativer Transformationen ausgehend vom *IfcProduct*, über sämtliche Strukturen, die dieses Objekt enthalten, bis hin zur *IfcSite* realisiert [33]. Diese Hierarchie kann der Abbildung 4.2 entnommen werden. Um die globale Position eines *IfcProducts* zu erhalten, müssen die relativen Transformationen miteinander verrechnet werden. Der IFC Standard unterstützt zudem die Möglichkeit Positionierungen anhand eines Rasters oder mithilfe des sogenannten *linear placements* anzugeben.

#### 4.1.4 *IfcOpeningElement*

Neben Wänden stellen Fenster und Türen wichtige Elemente eines Gebäudes dar. Diese sind ebenfalls Teil der Klassen des IFC Standards. Objekte des Typs *IfcOpeningElement* dienen dazu, die notwendigen Lücken in einer Wand zu definieren, in die später eine Tür oder ein Fenster eingebaut werden soll. In der Dokumentation wird dies wie folgt formuliert (aus dem Englischen): „[Das IfcOpeningElement] stellt eine Lücke in jedem Element dar, das eine physische Manifestation hat“ [54]. Es gibt zwei Arten der Klasse IfcOpeningElement, je nachdem ob die Öffnung durch die gesamte Breite eines Objektes reicht oder nicht. Folglich entsteht dadurch eine Unterscheidung zwischen Nischen und tatsächlichen Öffnungen, wobei nur letztere im Zusammenhang mit Fenstern oder Türen Sinn ergeben.

### 4.2 IFC und Blender

Blender ist eines der beliebtesten Open-Source-Programme zur Modellierung von 3D Modellen und Animationen [42]. Eine umfangreiche Python API erlaubt es Blender durch sogenannte Add-ons an die eigenen Bedürfnisse anzupassen [37] [39]. Aufgrund dessen existiert auch eine Vielzahl an freien Erweiterungen – unter anderem auch eine Integration des IFC Standards.

#### 4.2.1 **blenderbim**

Neben kommerziellen Produkten wie etwa revit von autodesk zur Modellierung von IFC Modellen, gibt es auch für Blender ein freies Plugin, um IFC Modelle zu erstellen [63] [43].

Dieses Plugin ermöglicht es neben dem bloßen Designen des Gebäudes in kurzer Zeit zum Beispiel detaillierte Zeichnungen verschiedener Perspektiven herauszuarbeiten, die etwa von Bauingenieuren 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 [50].

### 4.2.2 IfcOpenShell

```
1 import ifcopenshell
2 from ifcopenshell import geom
3 from stl import mesh, Mode
4 import numpy as np
5
6 settings = ifcopenshell.geom.settings()
7 settings.set(settings.USE_WORLD_COORDS, True)
8
9 ifc_file = ifcopenshell.open("model.ifc")
10 products = ifc_file.by_type("IfcProduct")
11 meshes = []
12
13 for product in products:
14     if product.Representation and product.is_a("IfcWall"):
15         shape = ifcopenshell.geom.create_shape(settings, product)
16         vertices = np.array(shape.geometry.verts).reshape((-1, 3))
17         edges = np.array(shape.geometry.edges)
18         faces = np.array(shape.geometry.faces).reshape((-1, 3))
19
20         m = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
21         for i, f in enumerate(faces):
22             for j in range(3):
23                 m.vectors[i][j] = vertices[f[j], :]
24         meshes.append(m)
25
26 # Create the combined mesh
27 combined = mesh.Mesh(np.concatenate([m.data for m in meshes]))
28 combined.save('model.stl', mode=Mode.ASCII)
```

Listing 4.1: Beispielprogramm zur Extraktion bestimmter Daten einer IFC Datei und Generierung eines Meshes aus deren geometrischen Representationen.

IfcOpenShell ist eine frei verfügbare Bibliothek, die es erleichtert mit Daten im IFC Format zu arbeiten [50]. Sie bietet unter anderem eine Python API an und ist wie bereits erwähnt ein Teil des Blender Add-ons blenderbim. Natürlich ist es dennoch möglich diese Bibliothek auch ohne Blender zu verwenden. In Listing 4.1 ist ein Beispielprogramm zu sehen, das aus einer IFC Datei alle Objekte des Typs IfcWall filtert und deren geometrische Repräsentationen in ein herkömmliches Mesh konvertiert. Auch andere Informationen, wie die in IfcPropertySets enthaltenen Daten, können mit der Bibliothek in ähnlicher Weise extrahiert werden. Damit existiert ein intuitiver Zugang zu den Daten von IFC Dateien.

## 4.3 Building Information Modeling

Ein weiterer Punkt, der für die Verwendung von IFC spricht, ist das sogenannte *Building Information Modeling* (BIM) [44]. Ein Definitionsvorschlag lautet wie folgt (aus dem Englischen): „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“ [6]. Zum Lebenszyklus eines Gebäudes gehören etwa anfangs das Planen und Designen, später das Bauen, daraufhin 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 [11]. Auch Verantwortlichkeiten sind Teil von BIM, was zu einer erhöhten Produktivität beiträgt. Um nun das Zusammenarbeiten der unterschiedlichen Fachbereiche zu erleichtern, gibt es sogenannte BIM-Server auf welchen mehrere Personen synchron an einem Projekt arbeiten können. Dabei können je nach Aufgabenbereich passende Teilansichten des Projektes herangezogen werden. BIM-Server unterstützen zusätzlich eine Versionierung des Fortschritts an einem Projekt.

In einem Gespräch mit einem Ingenieur aus dem Bereich „Energiesystemtechnik“ kam zur Sprache, dass viele Bereiche von BIM noch keinen Einzug in Deutschland gefunden haben. Eben jene „Kollaboration über einen BIM-Server mit Änderungsmanagement etc. [sei] 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 festgelegte Datenformat IFC das Maß der Dinge und auch bei uns so in Verwendung“. Auch das Einpflügen „ergänzende[r] Bauteilinformationen (z.B. zu Gewicht, Dämmwert, Recyclebarkeit, CO<sub>2</sub> 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 zum Beispiel 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 unrealistisch und auch etwas over-engineered“ eingestuft. Trotzdem sei „BIM [...] das große Ding in der Bauwelt und der einzige echte Standard“.

## 4.4 opensourceBIM

Während es vorwiegend kommerzielle Produkte gibt, die Unternehmen das Arbeiten mit BIM ermöglichen, existiert auch hier eine aktive Open Source Bewegung. Unter dem Namen „The open source BIM collective“ oder kurz „opensourceBIM“ werden derzeit

um die 70 Repositories betrieben. Darin enthalten sind unter anderem ein BIM-Server inklusive verschiedener Clients für Endanwender auf unterschiedlichen Systemen und Werkzeuge, die es erleichtern den IFC Files zu arbeiten [65]. Ein kurzer Test hat gezeigt, dass die in Blender modellierte IFC Files tatsächlich über einen „Anzeige-Client“, der mit einem lokal gehosteten BIM-Server verbunden ist, dargestellt und inspiziert 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 simultane Arbeiten an einem IFC File möglich ist, was in Blender nur teilweise und mit dem Einsatz von Plugins ermöglicht wird. 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 unterstützt 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.

## 4.5 Mauerwerksbau

Der Mauerwerksbau ist eine Art des Massivbaus, bei welchem Natur- oder Formsteine aufgeschichtet werden, um Wände beziehungsweise Mauern zu errichten. Eine derart erbaute Wand besteht demnach aus (Bau)-Steinen und den dazwischen entstehenden Fugen. Mörtel ist dabei nicht zwangsläufig notwendig. Man spricht von trocken versetz-

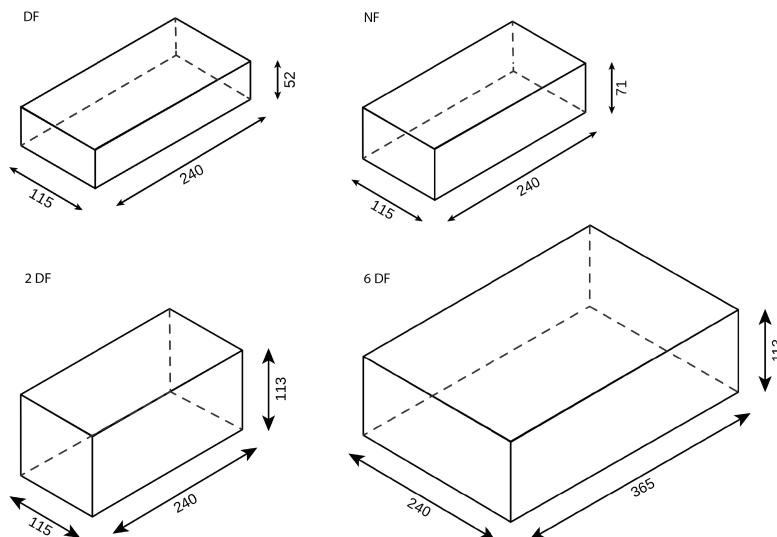


Abbildung 4.3: Darstellung verschiedener Steinformate nach DIN 4172 (Baunennmaß in Millimetern) [64].

ten Steinen oder einer Trockenmauer, wenn darauf verzichtet wird. Heutzutage wird fast ausschließlich mit quaderförmigen Formsteinen gebaut. Zur Beschreibung solcher Formsteine existieren zwei relevante Größen. Die eine ist das sogenannte *Baunennmaß*, mit dem die tatsächliche Größe des Steins angegeben wird. Die andere das *Baurichtmaß*,

das sich aus Baunennmaß und dem Fugenmaß zusammensetzt und sozusagen einem Raster entspricht, das durch die Bausteine vorgegeben wird.

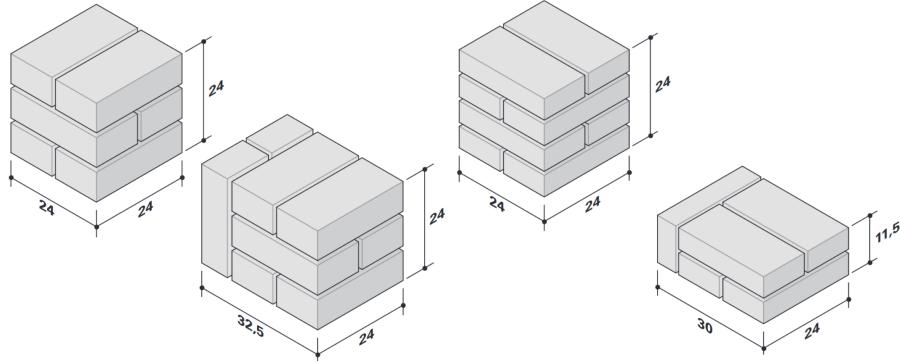


Abbildung 4.4: Besondere Eigenschaften der oktametrischen Maßordnung [27].

#### 4.5.1 Maßsysteme

Als Maßsystem bezeichnet man die Festlegung der Größen von Bausteinen anhand eines konkret definierten (Grund-) Moduls. Zwei in Deutschland populäre Grundmodule sind zum einen das *ISO-2848-Basismodul* mit Länge 100 mm und zum anderen der Achtelmeter, verankert in der *DIN 4172 Maßordnung im Hochbau* [59][67]. Letzteres bezeichnet man daher auch als das oktametrische Maßsystem. Nachfolgend wird auf dieses Maßsystem näher eingegangen.

##### Oktametrisches Maßsystem

Baurichtmaße sind gemäß dem oktametrischen Maßsystem immer ein Vielfaches von 12.5cm (das entspricht 1/8m) und nach der Norm aber mindestens 6.25cm. Dies gilt sowohl für Länge und Breite als auch für die Höhe der Steine. Das System ist in der *DIN 4172 Maßordnung im Hochbau* geregelt und ein fest definiertes Grundmaß für das Bauwesen in Europa [67]. Daraus gehen insbesondere folgende zwei Formate für Ziegelsteine hervor: das Normalformat (NF) mit 240 × 115 × 71mm und das Dünnformat (DF) mit 240 × 115 × 52mm (Länge × Breite × Höhe) [27]. Alle anderen Formate werden mithilfe dieser beiden Grundsteine angegeben. So sind zum Beispiel die in Abbildung 4.3 gezeigten 2 DF und 6 DF Steine eine Kombination aus mehreren Steinen im Dünnformat. Dabei sieht die Norm ein Fugenmaß von 10mm für Stoßfugen (vertikal) und 12mm für Lagerfugen (horizontal) vor. Für Systeme, die eine schmalere oder keine Fuge benötigen, werden entsprechend größere Steine hergestellt, um der Maßordnung weiterhin zu entsprechen. Durch Einhalten eines Systems ist man zusätzlich in der Lage Türen und Fenster an die daraus entstehenden Öffnungsgrößen anzupassen und vermeidet dadurch zeitaufwendiges, nachträgliches Anpassen oder Spezialanfertigungen. Da Höhe, Breite und Länge der Steine zusammen mit den dazwischenliegenden Fugen aufgrund des Maßsystems jeweils Vielfache voneinander sind, ergeben sich viele Möglichkeiten zur Aufschichtung

und Aneinanderreihung der Bausteine. Einige davon sind in Abbildung 4.4 zu sehen und sind gleichzeitig Beispiele für einen sogenannten *Mauerwerksverband*.

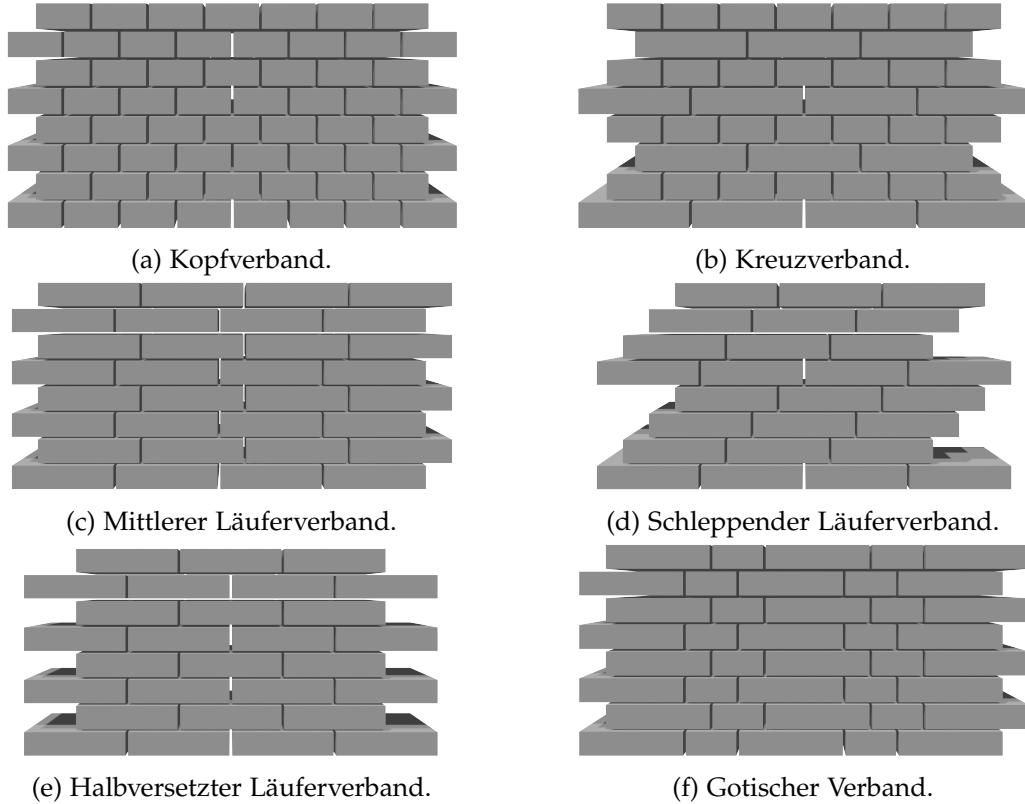


Abbildung 4.5: Typische Mauerwerksverbände. In diesem Beispiel weisen die Läuferverbände in 4.5c und 4.5d einen Versatz von  $1/4$  der Steinlänge auf.

#### 4.5.2 Mauerwerksverband

Als Mauerwerksverband bezeichnet man bestimmte, gleichmäßige Anordnungen von Mauersteinen, um einen homogenen Mauerwerkskörper zu erreichen [61]. 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  $h_u \leq 249\text{mm}$  nach DIN EN 1996-1-1:  $l_{ol} \geq 0,4h_u \geq 45\text{mm}$  [40][68]. Zudem wird darin die Mindestwanddicke für tragendes Mauerwerk, „sofern aus Gründen der Standsicherheit, der Bauphysik oder des Brandschutzes nicht größere Dicken erforderlich sind“ [40], auf  $t_{min} = 115\text{mm}$  festgelegt [68]. Dies ist, wie in Abbildung 4.3 zu sehen, exakt die Breite der kleinsten Ziegelformate NF und DF. Man unterscheidet zwei Arten von Mauerwerk: das Einsteinmauerwerk und das Verbandsmauerwerk. Wie schon dem Namen zu entnehmen, handelt es sich beim Einsteinmauerwerk um ein Mauerwerk, bei welchem die Wanddicke der Steinbreite entspricht. Hier muss das Überbindemaß lediglich über die Wandlängsrichtung eingehal-

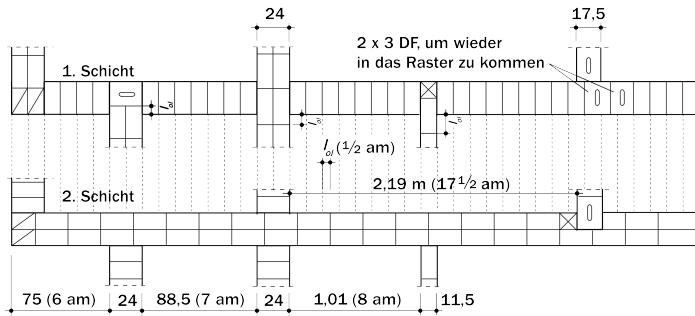


Abbildung 4.6: Lösung einer Kreuzung am Beispiel einer „Wand aus 2 DF im Kreuz- und Blockverband“ [48].

ten werden. Bei Verbandsmauerwerk gilt dies zusätzlich für die Wandquerrichtung [38]. Einige Beispiele sind in Abbildung 4.5 zu sehen. Als Wandstück wird von nun an ein gerader Wandabschnitt bezeichnet. Dieser hat eine Länge, eine Höhe und durch einen vorgegebenen Bausteintyp und einem gewünschten Verband eine gewisse Breite. Treffen zwei oder mehrere Wandstücke aufeinander, so gilt es diese dem Überbindemaß entsprechend miteinander zu verzahnen. Dabei treten verschiedene Sonderfälle auf, die für unterschiedliche Verbände nach unterschiedlichen Lösungen verlangen:

- **Ecken** werden durch zwei sich an Wandenden berührenden, meist rechtwinklig zueinanderstehenden Wandstücken gebildet.
- **Kreuzungen** stellen zum Beispiel zwei sich kreuzende Wandstücke dar.
- **T-Kreuzungen** entstehen, wenn ein Wandende eines einen Wandstücks auf einer anderen Wand steht. Sowohl bei Kreuzungen als auch T-Kreuzungen kann auf das aufwendige Verzähnen verzichtet und stattdessen die sogenannte Stumpfstoßtechnik angewandt werden. Dabei werden Stahlanker zwischen der Wand und den darauf treffenden „stumpfen“ Wandenden verwendet, um die beiden Wände sicher miteinander zu verbinden.
- **Wandenden** sind die „Enden“ eines Wandstücks, die kein anderes Wandstück berühren. Dafür muss der verwendete Mauerwerksverband zu einem geraden Abschluss gebracht werden. Oftmals lässt sich das Anpassen der Bausteine (etwa durch Zerschneiden) nicht vermeiden.
- **Öffnungen** innerhalb eines Wandstücks können in derselben Art behandelt werden, da der vorherrschende Verband in den betroffenen Schichten gerade unterbrochen werden muss. Über Öffnungen für Fenster und Türen wird ein sogenannter Sturz gelegt, welcher ebenfalls in den der Wand zugrunde liegenden Mauerwerksverband eingebunden werden muss.

Die Lösungen für die oben genannten Situationen variieren je nach angestrebten Mauerwerksverband und der verwendeten Modulgröße stark. Einige Beispiele sind in Abbildung 4.7 und Abbildung 4.6 zu sehen [27][48]. Während etwa beim Läuferverband

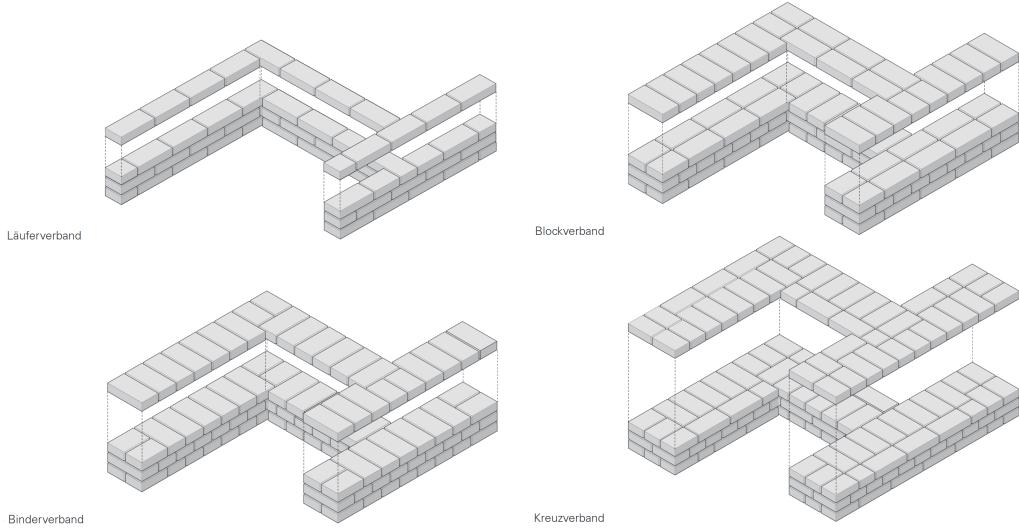


Abbildung 4.7: Lösungen für Ecken und T-Kreuzungen unterschiedlicher Verbände [27].

manchmal darauf verzichtet werden kann Bausteine für Eckbereiche und Kreuzungen zu zerschneiden, ist dies für andere Verbände teilweise unumgänglich.

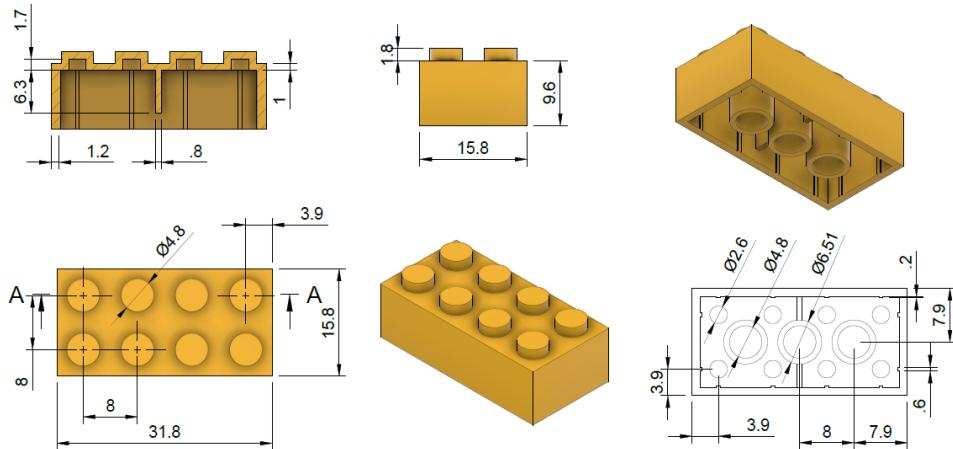


Abbildung 4.8: Maße des Standard  $2 \times 4$  LEGO Steins [60].

## 4.6 LEGO System

Ein  $1 \times 1$  LEGO Stein hat eine quadratische Grundfläche von  $7.8mm \times 7.8mm$ . Dies entspricht demnach dem Baunennmaß des  $1 \times 1$  LEGO Steins. Zwischen zwei nebeneinander platzierten Steinen ist ein Abstand von  $0.2mm$ . Daraus ergibt sich ein Baurichtmaß beziehungsweise ein Rastermaß von  $8mm \times 8mm$ . In Abbildung 4.8 werden zur Veranschaulichung die Maße des populären  $2 \times 4$  Steines aufgeschlüsselt. Die für ein dreidimensionales Maß 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 und demnach als Abstand von 0.0mm gewertet werden.

## 4.7 Ontologie

Der Begriff Ontologie stammt ursprünglich aus der Philosophie. Wörtlich übersetzt bedeutet er „Lehre des Seins“. Ein Definitionsvorschlag lautet wie folgt (aus dem Englischen): „Die Ontologie als Teilgebiet der Philosophie ist die Wissenschaft von dem, was existiert, den Arten und Strukturen von Objekten, Eigenschaften, Ereignissen, Prozessen und Relationen in allen Bereichen der Wirklichkeit“ [9]. Eine Anmerkung, die oftmals im Anschluss eines Definitionsvorschlags zu finden ist, besagt, dass dabei nicht nur das, *was ist*, sondern ebenfalls das, *was sein könnte* betrachtet werden müsse [9][36]. Dieser Gedanke spiegelt sich ebenfalls in dem Ontologiebegriff aus der Informatik wider und ist unter dem Begriff *Open World Assumption* bekannt. Diese Eigenschaft wird nach der Definition von Ontologien im Kontext der Informatik noch einmal ausführlicher aufgegriffen.

Im Fachbereich der Informatik wurde der Begriff Ontologie adaptiert und zunächst als „*explicit specification of a conceptualization*“ definiert [1]. Wörtlich übersetzt lautet diese Definition etwa „explizite Spezifizierung einer Konzeptualisierung“. Im Laufe der Jahre wandelte sich diese Definition, sodass sich daraus zum Beispiel folgendes ergab: „An ontology is a formal, explicit specification of a shared conceptualization“ [2]. Hier wird eine Ontologie also als „formale, explizite Spezifizierung einer geteilten Konzeptualisierung“ bezeichnet. Den wichtigsten Zusatz hierbei stellt das Wort „geteilt“ dar, da Ontologien unter anderem dem Zweck dienen, Informationen und Vokabulare mehrerer Wissensrepräsentationen zu vereinen. Der Begriff Konzeptualisierung wird oft als „vereinfachte, abstrakte Sicht auf einen für einen bestimmten Nutzen relevanten Teil der Welt“ beschrieben [5].

Eine solche „Spezifizierung einer Konzeptualisierung“ einer gewissen Domäne kann heutzutage mithilfe der sogenannten *Web Ontology Language* (OWL), welche auf dem *Resource Description Framework* (RDF) aufbaut, realisiert werden [7][14]. OWL erlaubt es Klassen zu definieren und daran Eigenschaften anzuhängen. Außerdem können sogenannte Individuen beziehungsweise Instanzen erstellt werden. Diese realisieren eine oder mehrere Klassen. Mithilfe dieser Klassen, Eigenschaften und Individuen können sogenannte *Reasoner* logische Schlussfolgerungen über implizit angegebenes Wissen ziehen. Eine Ontologie kann damit einerseits etwa mit neuen Klassenzuordnungen oder implizit angegeben Vererbungen erweitert, andererseits auf Validität überprüft werden. Werden Verletzungen von aufgestellten Axiomen oder widersprüchliche Einschränkungen von Klassen oder Eigenschaften entdeckt, so kann ein Reasoner diese nicht nur melden, sondern auch mithilfe logischer Beweisführung herleiten. Damit stellen Reasoner ein wertvolles Werkzeug für die Entwicklung und Erweiterung von Ontologien dar. Drei weit verbreitete Reasoner sind:

- ELK Reasoner der Universitäten Ulm und Oxford [13].
- HermiT Reasoner von der *Information Systems Group* der Universität Oxford [12][35].
- Open Source Pellet Reasoner der Clark & Parsia LLC [4].

Aufgrund der oben genannten Idee der *Open World Assumption* unterscheidet sich die Art der Reasoner logische Schlussfolgerungen zu ziehen von der, mit welcher logische Aussagen von zum Beispiel Programmiersprachen bewertet werden. Im Grunde versteht man unter der *Open World Assumption*, das nicht falsifizieren eines logischen Ausdrucks, falls Wissen existieren könnte, das den Ausdruck doch verifizieren würde. Ein Reasoner beziehungsweise eine Programmiersprache im Kontext der *Closed World Assumption* würde in so einer Situation davon ausgehen alle existierenden Informationen darüber bereits vorliegen zu haben und eventuell zu einem anderen Schluss gelangen. Definiert man zum Beispiel eine Klasse A als alle Individuen, die nur eine Eigenschaft E besitzen, die sie mit Individuen einer bestimmten anderen Klasse B verbindet, so kann ein Reasoner innerhalb dieser Ontologie ein Individuum, das zwar diese Anforderung erfüllt, nicht der Klasse A zuordnen, da es sein kann, dass zu einem späteren Zeitpunkt (oder an anderer Stelle) Informationen existieren, die diese Zuordnung fehlerhaft machen würden. In einer Umgebung, in der die *Closed World Assumption* gilt, wäre diese Zuordnung hingegen völlig korrekt. Die Annahme einer *offenen Welt* ermöglicht es ebenfalls die Idee des sogenannten „Semantic Web“ mithilfe von Ontologien zu realisieren [3]. Denn die Natur des Internets entspricht eher einer sich ständig ändernden und offenen Welt, als einer Geschlossenen. Als Semantic Web wird das Annotieren von sämtlichen Webinhalten mit semantischen Informationen bezeichnet. Damit wird das Ziel verfolgt, ein maschinenlesbares Internet zu schaffen.

### 4.7.1 Protégé

Protégé ist ein grafischer Editor zur Erstellung und Instandhaltung von Ontologien [17]. Entwickelt wird das Programm an der Universität Stanford und ist als Open Source Anwendung kostenfrei nutzbar. Die erste Version wurde bereits 1999 veröffentlicht. Durch seine Plugin-Struktur können neben zahlreichen Erweiterungen beispielsweise auch neue Reasoner an das Programm angeschlossen werden. Standardmäßig sind der ELK und der HermiT Reasoner integriert, aber es existieren Plugins, um auch Pellet in Protégé zu nutzen. Das Arbeiten mit einem grafischen Editor erleichtert den Einstieg in das Thema und ermöglicht später einen besseren Überblick über oftmals umfangreiche Ontologien, als es in einem herkömmlichen Texteditor möglich wäre. Darum wurde das Programm zur Erstellung einer Ontologie für diese Arbeit herangezogen.

### 4.7.2 Owlready2

Owlready2 ist eine Python 3 Bibliothek, die es ermöglicht *ontologieorientiert* zu programmieren [19]. Bisher mussten Ontologien mithilfe von Abfragesprachen (etwa SPARQL) oder APIs bearbeitet und ausgewertet werden [10]. Owlready2 bietet hingegen einen

einfacheren Umgang mit Ontologien, ebnet damit den Einstieg in das Thema und ermöglicht gleichzeitig die Integration von Ontologien in jedes mit Python umsetzbare Projekt. Im Zusammenspiel mit Protégé bietet diese Bibliothek eine für Programmierer intuitive Möglichkeit, die oben genannten Werkzeuge für IFC und BIM durch Nutzung von Python 3 direkt mit der Technologie der Ontologien zu koppeln.



# 5 Konzept

Mit den zuvor behandelten Grundlagen und den Inspirationen aus Kapitel 3 wird nun ein Konzept zur Bearbeitung der Problemstellungen der in Kapitel 2 definierten Fallstudien und Szenarien vorgestellt. Grob teilt sich das Vorgehen in drei Phasen: Falls noch kein Modell des geplanten Gebäudes vorliegt, muss dieses mithilfe eines Konstruktionsplaners erstellt werden. Abschnitt 5.1 beschäftigt sich mit verschiedenen Möglichkeiten diesen Modellierungsvorgang zu vereinfachen. Im Anschluss folgen einige Konzepte und Definitionen, die für das *Wall Detailing* relevant sind, das die zweite Phase darstellt und in Abschnitt 5.2 konkretisiert wird. Die dritte Phase beschäftigt sich damit, das Ergebnis des *Wall Detailings* durch Anwenden von Regelsätzen in eine geordnete Form zu überführen. Dies wird ab Abschnitt 5.3 behandelt.

## 5.1 Modellierung

Für gewöhnlich sind beim Planungsprozess eines Gebäudes oder anderer Infrastruktur eine Vielzahl an Experten aus unterschiedlichen Disziplinen beteiligt. Ein Ziel dieser Arbeit ist es allerdings eine intuitive Konstruktionsplanung zu ermöglichen, sodass eine Einzelperson mit relativ geringer Einlern- und Modellierungszeit in der Lage ist ein Gebäude zu entwerfen. Dieser Entwurf muss dennoch alle notwendigen Informationen für die anschließende Bauplandeduktion enthalten, ohne dass das Einpflegen dieser Daten spezielles Fachwissen voraussetzt.

Oftmals lässt sich die Komplexität einer Sache oder eines Vorgehens durch die Vorgabe von Einschränkungen reduzieren. Dabei ist es allerdings wichtig diese Einschränkungen so zu wählen, dass die damit erzielten Ergebnisse nach wie vor von Nutzen sind.

### 5.1.1 Raster

Im Fall von Gebäude- oder besser Gebilde-Konstruktionen existieren bereits einige Konzepte, die durch Einschränkungen eine intuitive und beschleunigte Modellierungsphase ermöglichen. Das wohl bekannteste ist das Lego System (siehe Kapitel 4.6). Neben dessen nützlichen Steckverbindungen, die es ermöglichen ohne Anwenden von Klebstoff oder Schrauben Steine aneinander zu befestigen, ist für diese Arbeit das dadurch vorgegebene Raster ein interessantes Konzept zur Vereinfachung der Modellierung von Gebäuden. In Kapitel 4.5 wurden auch schon die Begriffe *Baunennmaß* und des *Baurichtmaß* eingeführt und das oktometrischen Maßsystem vorgestellt. Dies entspricht im Prinzip ebenfalls einem Raster, das aber in Realität durch die Möglichkeit des Zerschneidens von Bausteinen nicht zwingend eingehalten werden muss. Da das Vorgeben eines Rasters, dem sowohl die Größen der zu verwendenden Bausteine, als auch (im Fall des Lego Systems)

deren Platzierung folgen, eine Einschränkung darstellt, die im Einklang mit der Intuition vieler Menschen steht, wurde dies in das Modellierungskonzept dieser Arbeit integriert. Das Vorgeben eines einzigen, fest definierten Rasters stellt allerdings eine zu große Einschränkung dar, weshalb das Definieren verschiedener Rastergrößen möglich sein muss. So können Modelle erstellt werden, die sowohl genau dem Lego System oder dem oktametrischen Maßsystem entsprechen, als auch beliebig anderen Rastern. Außerdem können dadurch unterschiedliche Raster auf verschiedene Objekte eines Modells angewendet werden.

Die Modellierungsumgebung kann mithilfe von Rasterinformationen eines Objektes für dessen korrekte Platzierung, Skalierung und Rotation sorgen, indem eine solche Transformation auf die nächstliegende Größenordnung des Rasters gerundet wird. Im Beispiel eines Rasters von  $[1.0m, 1.0m, 1.0m]$  würde demnach ein auf  $[0.9m, -0.7m, 0.1m]$  zu translatierendes Objekt an die Position  $[1.0m, -1.0m, 0.0m]$  versetzt werden. Da aber gewünschte valide Rotationen nicht aus einem derart vorgegebenen Raster interpretierbar sind, kann diese zusätzliche Information mithilfe eines kleinstmöglichen Winkelschritts angegeben werden. In den Modellen der Fallstudien aus Kapitel 2 werden zum Beispiel ausschließlich Rotationen eines Vielfachen von  $90^\circ$  verwendet, was neben den Rastern ebenfalls im Konstruktionsplaner hinterlegt wurde. So wird auch den Rotationen von Objekten durch Runden eine Art Raster aufgezwungen.

### 5.1.2 Wandstück

Auch die schiere Menge verschiedener Bestandteile eines Hauses ist für eine Einzelperson ohne Vorkenntnisse nicht überschaubar. Da das Ziel dieser Arbeit das automatische Generieren von Legeplänen für Bausteine innerhalb der Wände eines Gebäudes darstellt, lässt sich diese Menge vorerst auf zwei wesentliche Objekttypen reduzieren: Wände und Öffnungen in Wänden. Öffnungen werden zum Beispiel für Fenster oder Türen benötigt. Da eine Wand im Prinzip ein arbiträrer geometrischer Körper sein kann, dies abzubilden aber wieder die Komplexität der Modellierung steigert, wird deren Form auf einen beliebig skalierten Quader beschränkt. Ein solcher Quader wird nachfolgend auch als *Wandstück* bezeichnet. Ein solches Wandstück besitzt demnach auch die Eigenschaften eines Quaders: eine Skalierung, eine Rotation um seinen eigenen Mittelpunkt und eine Translation im Raum. Nachfolgend werden die Begriffe Länge, Breite und Höhe zur einfachen Unterscheidung von einer Skalierung in X, Y und Z Richtung verwendet. Während Länge und Höhe dem Raster entsprechend beliebig gewählt werden können, ergibt sich die Breite aus dem gewählten Bausteinformat (auch als *Modul* bezeichnet) und dem geplanten Mauerwerksverband (siehe Kapitel 4.5). Der Verband und das Modul können jedem Wandstück als sogenannter *Wandtyp* zugewiesen werden. Dadurch ist es möglich, verschiedene Arten von Wänden innerhalb eines Gebäudemodells zu verwenden. Diese Informationen werden später dafür verwendet, die durch das Wandstück abgesteckten Dimensionen sinnvoll mit Bausteinen zu füllen.

### 5.1.3 Mauerwerksverband

In Kapitel 4.5.2 wurden bereits einige verschiedene Mauerwerksverbände vorgestellt. Um dem Wall Detailing Verfahren diese und beliebig andere Verbände in einer interpretierbaren Weise zur Verfügung stellen zu können, wird dafür eine mathematische Notation benötigt. Bei genauer Betrachtung handelt es sich bei den meisten Mauerwerksverbänden um ein sich wiederholendes Muster. Dabei wiederholen sich sowohl die Bausteintransformationen innerhalb einer Schicht des Verbandes, als auch dessen Schichten selbst. Daher sind folgende Informationen zur Beschreibung eines Mauerwerksverbands notwendig:

1. Eine Menge einzigartiger Schichten, bis diese sich (mit einem Versatz in Z Richtung) wiederholen.
2. Für jede Schicht eine Menge verschiedener Bausteintransformationen, bis diese sich (mit einem Versatz in X Richtung) wiederholen.

Ein Mauerwerksverband kann damit als Menge von Schichten verstanden werden, die sich wiederum aus einer Menge von Bausteintransformationen zusammensetzen. Diese Schichten besitzen jeweils einen festgelegten, darunter liegenden Vorgänger sowie einen darüber liegenden Nachfolger. Das bedeutet, die Reihenfolge in der Schichten eines Mauerwerksverbands auftreten, darf nicht verändert werden. Leicht zu sehen ist dies am Beispiel eines Kreuzverbandes, der durch Veränderung der Reihenfolge seiner Schichten, seine markante, kreuzartige Form verlieren würde.

Eine Bausteintransformation wird in dieser Arbeit als Tupel mit folgendem Inhalt definiert:

1. Eine skalierbare Position  $\vec{p} = [x, y, z]^T$ .
2. Eine skalierbare Rotation  $r = (x, y, z)$  angegeben in Euler-Winkel.
3. Einen festen translativen Versatz  $\vec{p}_o = [x, y, z]^T$ .
4. Einen festen rotierenden Versatz  $r_o = (x, y, z)$  ebenfalls in Euler-Winkel.

Skalierbar bedeutet in diesem Fall, dass die Werte von beispielsweise der Position eines Bausteins in Abhängigkeit eines veränderbaren Faktors stehen. Zur endgültigen Platzierung eines Bausteins anhand einer Bausteintransformation aus einer Schicht eines Mauerwerksverbands wird lediglich noch ein Faktor benötigt, der angibt, die wievielte Wiederholung der Menge an Bausteintransformationen einer Schicht gerade Anwendung findet. Sei  $B_1 = (\vec{p}, r, \vec{p}_o, r_o) = ([l, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, 0))$  eine Bausteintransformation,  $M = (l, b, h) = (2, 1, 1)$  ein Modul und  $c \in \mathbb{N}$  ein Skalar, dann ergibt sich die Position  $P$  eines Bausteins zur Wiederholung  $c = 1$  wie folgt:  $P = \vec{p}_o + c * \vec{p} = [0, 0, 0]^T + 1 * [2, 0, 0]^T = [2, 0, 0]^T$ . Im nächsten Wiederholungsschritt mit  $c = 2$  ergibt dies eine Position  $P = [4, 0, 0]^T$ . Da die Rotationen als Euler-Winkel vorliegen, können diese in ähnlicher Weise berechnet werden. Sie beschreibt die Rotation um den Mittelpunkt des Bausteins. In den meisten Fällen ist  $r$  allerdings gleich  $(0, 0, 0)$  und  $r_o$  entweder ebenfalls  $(0, 0, 0)$  oder  $(0, 0, \frac{\pi}{2})$ . Letzteres entspricht einer festen Rotation um  $90^\circ$  um die z-Achse des Bausteins, unabhängig des Wiederholungsschritts  $c$ .

Mit  $B_2 = ([l, 0, 0]^T, (0, 0, 0), [\frac{l}{2}, 0, 0]^T, (0, 0, 0))$  und je einer Schicht gebildet aus den Mengen  $\{B_1\}$  bzw.  $\{B_2\}$  erhält man bereits eine komplette Beschreibung für den Läuferverband mit einem Versatz von 50 % der Steinlänge. Dieses Beispiel ist in Abbildung 5.1 schematisch dargestellt. Aber auch andere, komplexere Verbände können in dieser Form abgebildet und so dem nachfolgenden Wall Detailing in abstrahierter Weise übergeben werden.

Die speziellen Legepläne zur korrekten Auflösung von kritischen Bereichen wie zum Beispiel Ecken können ebenfalls in dieser Form für jeden Mauerwerksverband angegeben werden. Für das vorangegangene Beispiel sähe der Eckplan wie folgt aus: Eine Nulltransformation  $E_1 = ([0, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, 0))$  in der ersten Schicht und eine Bausteintransformationen  $E_2 = ([0, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, \frac{\pi}{2}))$  mit einer festen 90° Rotation um die z-Achse in der zweiten. Dieser Eckplan ist in Abbildung 5.2 zu sehen. Ein Beispiel, in welchem dieser Eckplan angewandt wird, wurde bereits in Abbildung 4.7 in Kapitel 4.5.2 gezeigt. Der einzige Unterschied zwischen dem Plan eines Mauerwerksverbands für gerade Wände und dessen Spezialplänen ist deren Anwendung. Während aus dem Eckplan später für jede Schicht nur ein Baustein pro hinterlegter Bausteintransformation erzeugt werden muss, können beliebig lange gerade Wandabschnitte mithilfe des oben angesprochenen Wiederholungsschritts mit Bausteinen aufgefüllt werden.

#### 5.1.4 Beziehungen zwischen Wandstücken

In dieser Arbeit stehen Wandstücke zueinander in einer Beziehung, sobald sie sich berühren oder schneiden. Diese Beziehungen sind relevant, um die gewählten Mauerwerksverbände ordnungsgemäß über mehrere voneinander abhängige Wandstücke anzuwenden, ohne dass es zu Verletzungen von Vorschriften und Normen wie etwa dem Überbindemaß aus Kapitel 4.5.2 kommt. Dort wurden zudem die verschiedenen Arten von Beziehungen, die für das nachfolgende Wall Detailing zu unterscheiden sind, thematisiert. Zur Modellierung von Gebäuden sind vor allem folgende Beziehungen unabdingbar: *Ecken*, *T-Kreuzungen* und *X-Kreuzungen*. Diese stellen wichtige Elemente bei der Planung von Gebäuden dar. Nur so ist es möglich abgeschlossene und aneinanderhängende Räume zu erstellen. Eine weitere, nicht ganz offensichtliche Beziehung besteht zwischen zwei Wandstücken, die in Verlängerung zueinander stehen, also gemeinsam einen größeren Wandbereich abdecken. Diese Beziehung wird nachfolgend *Wandstückver-*

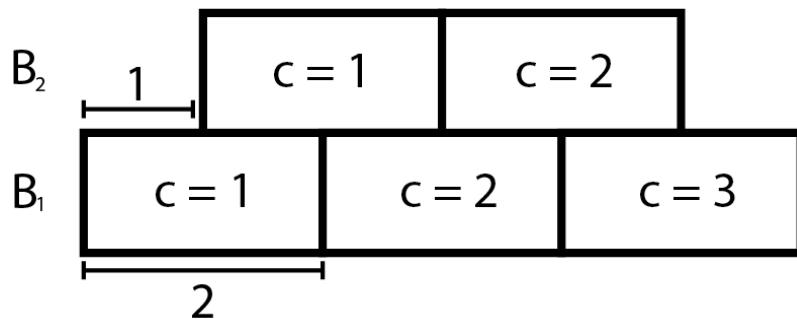


Abbildung 5.1: Der Läuferverband mit einem Versatz von 50 %.

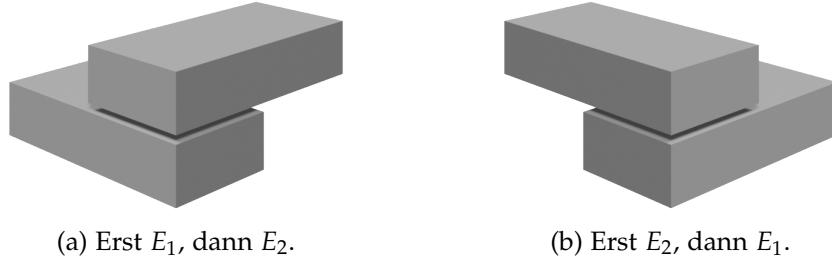


Abbildung 5.2: Eckpläne für den Läuferverband mit einem Versatz von 50%.

bund genannt. Bei allen vier Beziehungen handelt es sich Sonderfälle, für welche sich der Mauerwerksverband nicht ohne weiteres anwenden lässt, sondern teilweise explizit vorgegebene Legepläne angewandt werden müssen. Darum ist es notwendig diese Bereiche ausfindig zu machen und voneinander unterscheiden zu können. Es werden nun für jede dieser Beziehungen Eigenschaften aufgezeigt, anhand derer man diese in einer Menge an Wandstücken ermitteln kann.

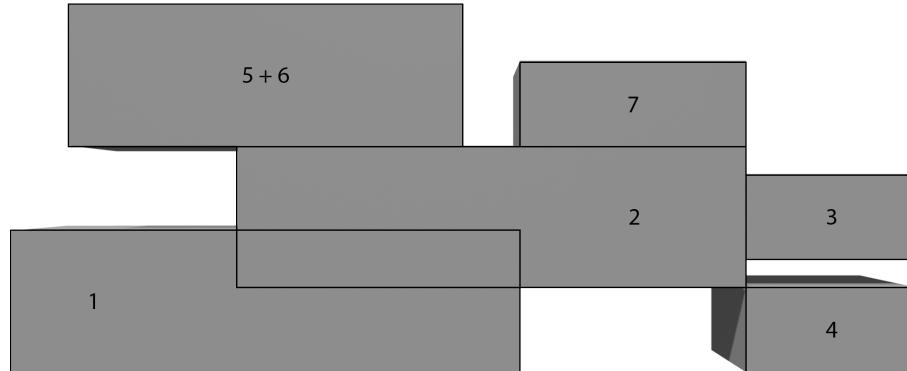


Abbildung 5.3: Modell einer Wand, die durch 7 einzelne Wandstücke desselben Typs gebildet wird, deren Mittelpunkte alle auf einer Ebene liegen.

**Wandstückverbände** sind alle Wandstücke, die zwar in dem Modellierungsprozess des Gebäudes durch mehrere einzelne Wandstücke realisiert wurden, eigentlich aber eine Einheit darstellen. Es ist notwendig alle Wandstücke zu identifizieren, die einen zusammenhängenden Wandbereich bilden, um den gewählten Mauerwerksverband korrekt über alle verbundenen Wandstücke hinweg anzuwenden. Andernfalls würde jedes Wandstück den Mauerwerksverband unabhängig seiner Nachbarn neu beginnen und ihn dadurch zwischen Wandstückübergängen häufig fälschlicherweise unterbrechen. Ein umfangreiches Beispiel ist in Abbildung 5.3 dargestellt. Um zu überprüfen, ob zwei Wandstücke eine Einheit bilden, kann das Paar auf folgende Eigenschaften getestet werden:

1. Beide Wandstücke verwenden sowohl dasselbe Modul als auch denselben Verband und sind während der Modellierung mit den gleichen Wandtyp annotiert worden. Dies verhindert vor allem das Kombinieren unterschiedlich dicker Wände.
2. Die lokalen Z-Achsen beider Wandstücke sind parallel. Dies verhindert das Kombinieren unpassend rotierter Wandstücke. Die Überprüfung der Parallelität der Z-Achsen beider Wandstücke wird wie folgt durchgeführt:

Sei  $\vec{v}_z = [0, 0, 1]^T$  eine z-Achse und  $v_z = (v_W, v_X, v_Y, v_Z) = (0, 0, 0, 1)$  das dazugehörige Quaternion. Außerdem seien  $q_1$  und  $q_2$  die beiden Rotationen der Wandstücke. Dann stellen  $v_{z^1} = q_1 * v_z * q_1^{-1}$  und  $v_{z^2} = q_2 * v_z * q_2^{-1}$  die jeweiligen „Z-Anteile“ dieser Rotationen dar. Daraus ergeben sich die „Z-Vektoren“ wie folgt:  $\vec{v}_{z^1} = [v_{z_x^1}, v_{z_y^1}, v_{z_z^1}]^T$  und  $\vec{v}_{z^2} = [v_{z_x^2}, v_{z_y^2}, v_{z_z^2}]^T$ . Ist nun  $|\vec{v}_{z^1} \cdot \vec{v}_{z^2}| = 1$  gegeben, so sind die lokalen Z-Achsen der Wandstücke gleich oder um exakt  $180^\circ$  verdreht und damit Parallelität erfüllt.

3. Die lokalen X-Achsen beider Wandstücke sind ebenfalls parallel. Das Vorgehen zur Überprüfung entspricht dem von Punkt 2, nur dass  $v_z$  durch  $v_x = (0, 1, 0, 0)$ , also der x-Achse, ersetzt werden muss.
4. Sie stehen auf derselben Höhe oder versetzt um ein Vielfaches der gemeinsamen Modulhöhe.
5. Es liegt eine Berührung oder gar eine Überlappung vor.

Insgesamt bilden in dem Beispiel in Abbildung 5.3 demnach alle Wandstücke mit Ausnahme von Wandstück Nummer 4 eine Einheit, die alle obigen Eigenschaften erfüllt. Das sind in der Tat all die Wandstücke, die den Start des gewählten Mauerwerksverbands ihren Nachbarn entsprechend anpassen müssen, sodass sich der Verband gleichmäßig über alle Wandstücke erstreckt. Dieses Beispiel wird in Kapitel 6 fortgeführt.

**Ecken, T-Kreuzungen und X-Kreuzungen** stellen Beziehungen zwischen einzelnen Wandstücken oder den kombinierten Wandstückverbänden aus Abschnitt 5.1.4 dar. Wie schon zu Beginn dieses Abschnitts vorweggenommen, existieren für jeden Mauerwerksverband und je nach verwendetem Modul eigene Varianten für Eck- und Kreuzungslegepläne. Einige Beispiele hierfür wurden schon in Kapitel 4.7 gegeben. Der Grund dafür ist die Notwendigkeit, in diesen Bereichen vorgeschriebenen Normen, wie etwa das im Abschnitt 4.5.2 genannte Überbindemaß, einzuhalten. Diese speziellen Pläne werden dann an den notwendigen Stellen angewandt – meistens bevor ein dazwischenliegender gerader Wandabschnitt mit Bausteinen aufgefüllt wird. Es liegt deshalb nahe, die Eck- und Kreuzungslegepläne zur Beschreibung des Mauerwerksverbands hinzuzufügen, wie es schon im Abschnitt 5.1.3 definiert wurde. Durch das vorangegangene Kombinieren mehrerer Wandstücke zu einer Einheit, können nachfolgend Wandstücke auch das Ergebnis solcher Kombinationen sein, aber aufgrund deren Parallelität wie ein einzelnes behandelt werden. Da sich Ecken, T-Kreuzungen und X-Kreuzungen stark ähneln, besitzen sie einige geteilte Eigenschaften:

1. Beide Wandstücke verwenden dasselbe Modul und sind während der Modellierung mit den gleichen Wandtyp annotiert worden.
2. Die lokalen Z-Achsen beider Wandstücke sind parallel. Das Vorgehen zur Überprüfung ist dasselbe wie in Abschnitt 5.1.4.
3. Sie stehen auf derselben Höhe oder versetzt um ein Vielfaches der gemeinsamen Modulhöhe.
4. Mindestens eines der beiden Wandstücke endet auf einem anderen.

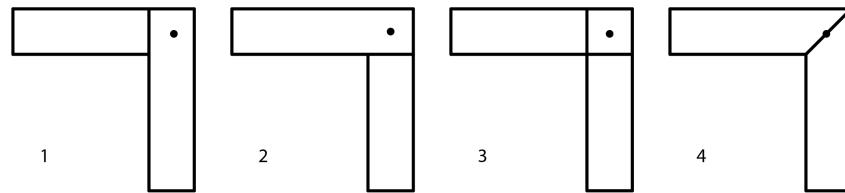


Abbildung 5.4: Draufsicht auf Varianten der Modellierung einer Ecke.

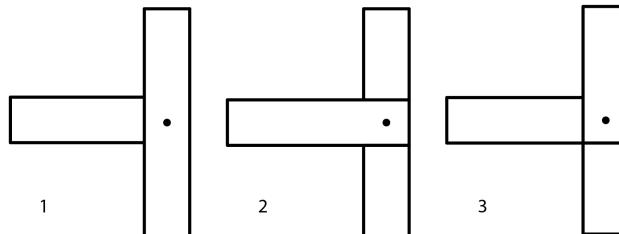


Abbildung 5.5: Draufsicht auf Varianten der Modellierung einer T-Kreuzung.

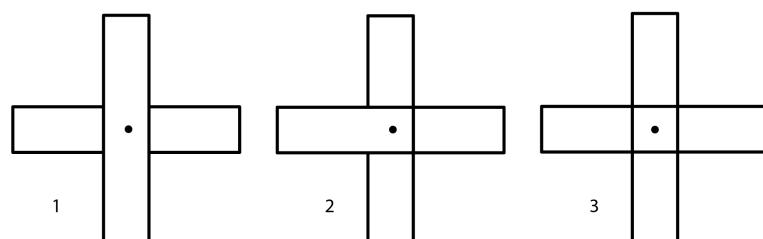


Abbildung 5.6: Draufsicht auf Varianten der Modellierung einer X-Kreuzung.

Sind diese Eigenschaften erfüllt, werden die Schnittpunkte der Richtungsvektoren entlang der lokalen X-Achsen beider Wandstücke errechnet. Im Falle einer einfachen Ecke oder einer T-Kreuzung, existiert nur ein einzelnes Paar an Wandstücken, die sich im Mittelpunkt der Ecke oder der T-Kreuzung schneiden. Der Unterschied ist lediglich die Stelle des Schnittpunkts relativ zu den beiden Wandstücken. Sei  $w_1$  die Breite von einem

und  $w_2$  die des anderen Wandstücks, so gilt nach Punkt 1  $w_1 = w_2$ . Liegt der errechnete Schnittpunkt bei beiden Wandstücken näher als  $\frac{w_1}{2}$  entlang der lokalen x-Achse an einer der beiden Außenkanten handelt es sich um eine Ecke, da beide Wandstücke an diesem Punkt enden. Falls der Schnittpunkt bei einem der beiden Wandstücke allerdings mindestens  $\frac{w_1}{2}$  innerhalb des Wandstücks liegt, so handelt es sich um eine T-Kreuzung. Ein Wandstück steht also auf einem Anderen. Dies ist in den Abbildungen 5.4 und 5.5 zu sehen. Werden für zwei unterschiedliche Paare dieselben Schnittpunkte errechnet, so kann es sich entweder um eine mit drei Wandstücken modellierte T-Kreuzung oder X-Kreuzung handeln. Treffen sich an dem Schnittpunkt zwei Ecken, ergibt sich daraus eine T-Kreuzung (siehe Fall 2 in Abbildung 5.5). Teilen sich aber zwei T-Kreuzungen dieselben Schnittpunkte, so sind alle Voraussetzungen für eine X-Kreuzung gegeben (siehe Fall 1 in Abbildung 5.6). Einen Sonderfall für Punkt 4 stellt eine X-Kreuzung dar - modelliert aus zwei sich schneidenden Wandstücken. Dieser Fall ist in Abbildung 5.6 mit der Variante Nummer 3 dargestellt. Hier liegt der Schnittpunkt des Wandstück-Paars mindestens  $\frac{w_1}{2}$  innerhalb beider Wandstücke und keines der Wandstücke endet auf dem Anderen. Durch das vorhergehende Kombinieren von Wandstücken werden die Fälle 3 aus Abbildung 5.5 und 2 aus Abbildung 5.6 eliminiert, denn dabei erfüllen jeweils zwei Wandstücke alle Eigenschaften, um nach Abschnitt 5.1.4 zu einer Einheit verschmolzen zu werden. So entsteht aus Fall 3 aus Abbildung 5.5 der Fall Nummer 1. Auch Fall Nummer 2 aus Abbildung 5.6 wird damit zu deren 1. Fall.

### 5.1.5 Lösen von Beziehungen

Durch Ecken, T- und X-Kreuzungen werden ansonsten unabhängige Wandstücke miteinander verkettet. Das kann zu Situationen führen, in welchen das einfache Anwenden eines Mauerwerksverbands Lücken in manchen Wandstücken hinterlässt. Das Beispiel in Abbildung 5.7 soll helfen diese Problematik zu veranschaulichen. Zu sehen ist eine Konstellation von vier Wandstücken, die in einer Kette durch drei Ecken miteinander verbunden sind. Das anzuwendende Modul hat die Dimensionen [2, 1, 0.5]. Die Höhe des Moduls ist für dieses Problem zunächst irrelevant, weshalb die folgenden Skizzen ausschließlich die Draufsicht auf die Situationen zeigen. Wie in Kapitel 5.1.3 am Beispiel des Läuferverbands gezeigt, gibt es mehrere Möglichkeiten einem Eckplan zu folgen. Solange die vorgegebene Reihenfolge eines Plans weiterhin eingehalten wird, kann mit jeder Schicht des Eckplans begonnen werden einen Eckbereich aufzufüllen. Das führt in dem Beispiel aufgrund des zweischichtigen Eckplans des Läuferverbands dazu, dass für jede der drei Ecken zwei mögliche Startschichten existieren. Dies resultiert demnach in insgesamt acht unterschiedlichen Varianten, von welchen allerdings nicht alle zielführend sind. In Abbildung 5.8 ist eine Variante dargestellt, die Lücken in der Wand hinterlassen würde. Das zunächst naheliegend wirkende Füllen der Lücken durch kleinere Bausteine ist oftmals nicht zulässig, da damit das Überbindemaß (siehe Kapitel 4.5.2) verletzt wäre. Darum ist es notwendig eine Kombination an Startschichten über alle miteinander verbundenen Eckbereiche zu finden, die es ermöglicht, die verbleibenden geraden Wandstücke im Anschluss durch einfaches Anwenden des Läuferverbands aufzufüllen, ohne dass Lücken entstehen.

Ein Verfahren, welches für das vorliegende Beispiel des halbversetzten Läuferverbands,

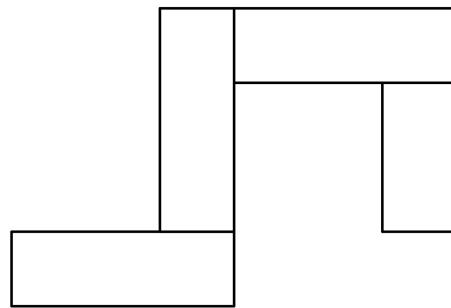


Abbildung 5.7: Draufsicht auf eine Verkettung von 4 Wandstücken durch 3 dazwischenliegende Ecken.

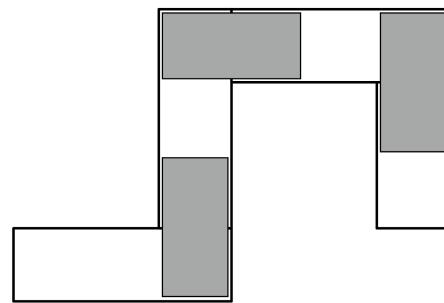


Abbildung 5.8: Draufsicht auf eine unzulässige Lösung der Eckplankonfiguration.

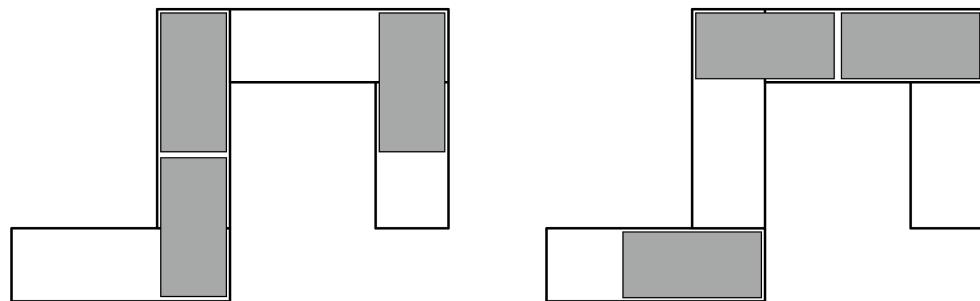


Abbildung 5.9: Eine mögliche Lösung und die durch die Eckpläne entstehende zweite Lösung für die darüber liegende Schicht.

dessen sehr einfachen Eckplans und das aufgrund seiner praktischen Form gewählten Modul Lösungen findet, löst nicht zwangsläufig auch eine beliebig andere Situation. Ebenfalls führt das Miteinbeziehen der 3. Dimension, also der Höhe der beteiligten Wandstücke, zu einer erheblichen Vervielfachung möglicher Eck-Konstellationen. Zieht man etwa das Beispiel aus Abbildung 5.3 heran und erweitert es durch Einfügen weiterer Wandstücke um Ecken an den Außenkanten von Bereich 1, 3, 4, 5 + 6 und 7, an welchen beliebig weitere verkettete Wandstücke in sich geschlossene Kreise bilden, so können sogar nicht lösbare Fälle auftreten. Durch die Anzahl der Schichten des Eckplans eines Mauerwerksverbands und die Anzahl der miteinander verketteten Wandstücke baut

sich schnell ein großer Lösungsraum auf. Aufgrund der Einschränkung eine mögliche Lösung lediglich mit zulässig oder unzulässig bewerten zu können, sind herkömmliche Optimierungsverfahren nicht in der Lage das Problem zu lösen. Diese benötigen meistens eine sogenannte *Fitness Funktion*, welche die Güte der derzeitigen Lösung angibt und dem Optimierungsverfahren sozusagen eine Richtung weist, in der ein lokales Minimum oder Maximum dieser Funktion zu erwarten ist. Wählt man die *Fitness Funktion* sinnvoll, kann damit ein kontinuierlicher Lösungsraum effizient nach einer passablen Lösung durchsucht werden. In dem vorliegenden Fall gibt es allerdings, wenn überhaupt, nur einige wenige zulässige Lösungen, die an den ganzzahligen Startindizes der Pläne der jeweiligen Eckbereiche gekoppelt sind. Diese Startindizes weisen aber nicht dieselben Eigenschaften auf, wie kontinuierliche Parameter, die durch kleine Veränderungen eine etwas bessere oder schlechtere Lösung erzeugen. Das Ändern eines Startindex einer einzelnen Ecke kann zwar eine Lösung mit einer reduzierten Anzahl an Lücken liefern, gibt aber deswegen nicht zwangsläufig eine Richtung vor in der man noch bessere Lösungen erwarten kann. Außerdem ist auch eine Lösung mit einer einzigen Lücke nicht zulässig, da dies die Stabilität eines Wandstücks stark beeinflussen könnte.

Darum bleibt zunächst nur die Brute-Force-Methode (im Deutschen auch als Exhaustionsmethode bekannt), also das Berechnen sämtlicher Konfigurationen bis eine zulässige gefunden wurde. Allerdings fällt schnell auf, dass das Wählen eines einzigen Startindex für eine Ecke alle direkt und indirekt damit verketteten anderen Ecken beeinflusst. Wählt man in dem obigen Beispiel in Abbildung 5.9a etwa die untere Ecke aus und legt deren Startindex auf die abgebildete Situation fest, so verbleibt ohnehin nur noch eine zulässige Konfiguration für die darüber liegende Ecke. Andernfalls befände man sich in der Situation aus Abbildung 5.8, welche keine zulässige Lösung darstellt. Setzt man nun den Eckplan-Index für diese Ecke ebenfalls fest, zwingt man deren zweiten Nachbarn zwangsläufig in die abgebildete Situation, denn der zweite Fall würde wieder zu einer unzulässigen Lösung führen. Da sich an einem Wandstück übereinander mehrere Ecken befinden können, bildet sich damit ein Abhängigkeitsgraph zwischen allen miteinander verketteten Wandstücken. Dieser kann wie folgt erstellt werden:

1. Wähle eine Startecke und finde alle darüber- und darunterliegenden Ecken, die durch Festlegen eines Startindex einer Ecke in einer dieser Schichten direkt davon betroffen sind. Dies stellt den ersten Knoten des Abhängigkeitsgraphen dar.
2. Finde alle Nachbarn aller zuvor gefunden Ecken.
3. Wiederhole für jeden Nachbarn das Vorgehen aus dem ersten Schritt. Daraus resultieren die nächsten Knoten, welche mit dem Vorgängerknoten verbunden werden.
4. Stößt man dabei auf einen schon existierenden Knoten, so bricht man diesen Pfad an der Stelle ab. Dieser Fall tritt zum Beispiel auf, wenn in dem Modell ein abgeschlossener Raum existiert.

Nun kann das Modell mithilfe des Graphen ähnlich einer Breitensuche durchlaufen und ausgehend von dem Startindex des Eckplans eines Knotens die Indizes von dessen

Nachbarn in passender Weise gewählt werden. Dies wiederholt man für alle Nachbarn des derzeitigen Knotens, im nächsten Schritt für deren Nachbarn, solange bis alle Knoten einen Startindex zugewiesen bekommen haben. In ungünstigen Situationen hängt das Finden einer passenden Lösung von dem Startknoten ab. Dies tritt manchmal auf, wenn mehrere Kreise im Abhängigkeitsgraph existieren. Wird für den gewählten Startknoten mithilfe des Verfahrens keine zulässige Lösung gefunden, so muss ein anderer Startknoten ausprobiert werden. Findet man für keinen Startknoten eine valide Lösung, so existiert für das Modell und den Modulen der gewünschten Mauerwerksverbände keine naheliegende, lückenlose Bausteinkonfiguration.

### 5.1.6 Öffnungen

Neben Wänden sind Öffnungen ein integraler Bestandteil bei der Planung von Gebäuden. Öffnungen sind Bereiche an welchen das Mauerwerk unterbrochen wird, um Lücken zu schaffen, in die später zum Beispiel Fenster oder Türen eingesetzt werden können. Da eine Öffnung nicht ohne eine Wand existieren kann, liegt es nahe, die dazugehörigen Informationen in der Wand zu hinterlegen, die von der Öffnung betroffen ist. Eine Öffnung kann ebenfalls als quaderförmiger Körper angesehen werden, der einen Bereich innerhalb eines Wandstücks definiert, an dem später keine Bausteine gelegt werden dürfen. In dieser Arbeit besitzt dieser Quader dieselbe Orientierung wie das dazugehörige Wandstück und schneidet zunächst immer durch die gesamte Breite der Wand. Daher sind nur Position, Länge und Höhe des Quaders von Bedeutung für das Wall Detailing. Da es allerdings üblich ist einen sogenannten Sturz über der eigentlichen Öffnung eines Fensters oder einer Tür anzubringen, muss auch für dieses Objekt Platz im Wandstück geschaffen werden. Der Sturz ist etwas länger als die Länge der Öffnung und liegt an beiden Seiten auf dem Mauerwerk auf, um den Baustein darunter eine stabile Grundlage zu bieten. So bleibt die Stabilität der Wand trotz Öffnungen erhalten.

### 5.1.7 Wandenden

Stehen eine oder beide Kanten eines Wandstücks allein, ohne dort über eine Beziehung mit einem anderen verbunden zu sein, muss der Mauerwerksverband zu einem geraden Abschluss gebracht werden. Häufig ist dies nur möglich, indem man für manche Bausteine die Länge des verwendeten Moduls anpasst.

In Abbildung 5.10a werden für den Läuferverband Bausteine mit halber Modullänge und für den Kreuzverband in der Abbildung daneben zwei angepasste Versionen des Moduls benötigt: Erneut ein Bausteinformat mit halber Modullänge um die Läuferverbandschichten des Kreuzverbands abzuschließen und ein Bausteinformat, dessen Länge sich aus der halben Modulbreite bildet und für den Abschluss der Kopfverbandschichten notwendig ist. Ersteres ist aber aufgrund der doppelten Wanddicke des Kreuzverbands zufällig gleich dem verwendeten Modul. Aber nicht nur an Wandenden muss ein Mauerwerksverband mit einer geraden Kante abgeschlossen werden. Auch an den zuvor angesprochenen Öffnungen ist dies für die Schichten erforderlich, die durch die Öffnung geteilt werden. Für den Fall, dass für das vollständige Detailing eines Modells das Anpassen des gewünschten Moduls notwendig wird, muss diese Information neben den

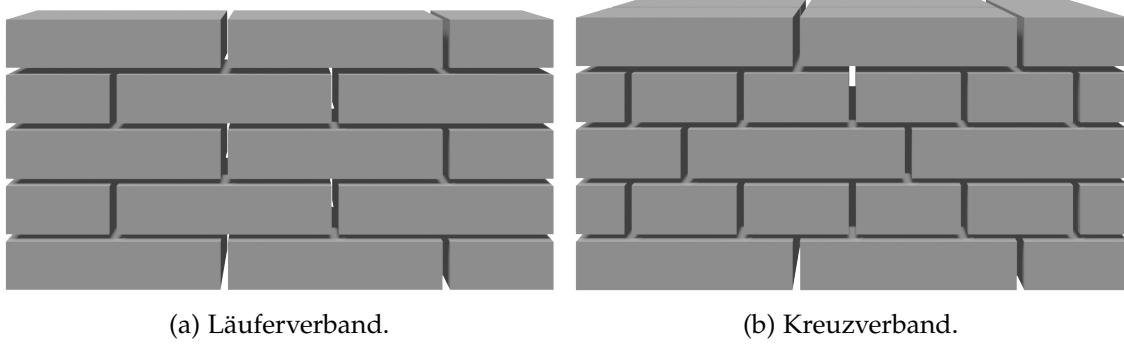


Abbildung 5.10: Wandenden verschiedener Mauerwerksverbände.

errechneten Transformationen der Bausteine ebenfalls in das Ergebnis integriert werden. Somit erhält jeder Baustein aus diesem Grund zusätzlich eine Beschreibung über dessen Dimensionen.

## 5.2 Wall Detailing

Als „Wall Detailing“ (sprich das „Detaillieren von Wänden“) wird in dieser Arbeit der Vorgang bezeichnet, ein als geometrischer Körper definiertes Wandstück in konkretes Mauerwerk zu überführen. Dieser Vorgang wendet die zuvor behandelten Konzepte an, um Lösungen für beliebig komplexe Gebäudemodelle zu errechnen. Als Lösung sollen nur diejenigen Bausteinmengen gelten, die sämtliche von Wandstücken abgedeckten Bereiche lückenlos mit Bausteinen füllen, während gleichzeitig die gewünschten Mauerwerksverbände eingehalten werden. Das nachfolgende schrittweise Verfahren weist aufgrund der ähnlichen Zielsetzung zwangsläufig Ähnlichkeiten zu dem von Usmanov et al. auf, das bereits in Kapitel 3.2.1 zusammengefasst wurde.

1. Extrahieren relevanter Informationen wie Geometrie und Typ-Annotationen aus dem vorgegebenen Gebäudemodell. Dabei interessieren in erster Linie Wände und deren Öffnungen sowie gewünschte Module, Raster und Mauerwerksverbände (siehe Abschnitt 5.1.3).
2. Anwenden des vorgegebenen Moduls für jedes gefundene Wandstück. Das führt zu einer schichtweisen Repräsentation jedes Wandstücks. Dies erleichtert es später Operationen an und zwischen mehreren Wandstücken durchzuführen.
3. Finden und Kombinieren von Wandstückverbänden nach Abschnitt 5.1.4.
4. Finden und Lösen von anderen Beziehungen wie Ecken T- und X-Kreuzungen nach Abschnitt 5.1.4 (Finden) und Abschnitt 5.1.5 (Lösen).
5. Schichtweises Anwenden der Öffnungen jedes Wandstücks. Dabei werden alle betroffenen Schichten in passender Weise geteilt und deren Länge entsprechend der Öffnungsmaße reduziert.

6. Schichtweises Anwenden der den Wandstücken zugewiesenen Mauerwerksverbände. Dazu gehören sowohl die besonderen Bereiche aus Punkt 4, als auch gerade Wandabschnitte und allein stehende Wandenden.
7. Finden von (direkten) Nachbarschaftsbeziehungen zwischen Bausteinen.
8. Konvertieren in das für die nachfolgende Bauplandeduktion notwendige Format.

Das Ergebnis dieser Schritte ist eine Menge aus Bausteinen, nachfolgend auch als *Bauplanentwurf* bezeichnet. Durch die Berechnungen in Schritt 7 werden bereits die grundlegendsten Beziehungen zwischen den Bausteinen hergestellt. Eine Aussage darüber in welcher Reihenfolge die Bausteine gesetzt werden müssen, um das geplante Gebäude unter Berücksichtigung etwaiger physikalischer oder anderer Einschränkungen zu errichten, wird allerdings noch nicht getroffen. Dazu müssen Schritt für Schritt Teilmengen aus der Gesamtmenge an Bausteinen extrahiert werden, die nur diejenigen Bausteine beinhalten, die in einer bestimmten Situation gelegt werden können. Welche das sind, soll anhand vorgegebener Regeln ausgewertet werden können. Diese Regeln werden dazu verwendet, bestimmte Voraussetzungen an einen Baustein zu setzen, um zum Beispiel das Ablegen eines Bausteins zu verhindern, unter dem sich weder fester Boden noch ausreichend andere Bausteine befinden. Denn setzt man einen Baustein mitten in der Luft ab, so fällt er in der echten Welt aufgrund der auf ihn einwirkenden Gravitation zu Boden. Außerdem weisen unterschiedliche Bausteinarten und Umgebungen, in die man das Gebäude mithilfe des zuvor errechneten Bauplanentwurfs errichten möchte, womöglich unterschiedliche Einschränkungen auf. So ist es notwendig je nach Situation voneinander abweichende Regelsets auf dem Bauplanentwurf anzuwenden zu können. Die Möglichkeit der nachträglichen Definition dieser Regeln verhindert zusätzlich die Notwendigkeit den Bauplanentwurf für sich unterscheidende Situationen neu berechnen zu müssen.

### 5.3 Regelbasierte Bauplandeduktion

Ontologien bieten verschiedene Werkzeuge an, mithilfe derer nicht nur streng definierte Objekttypen und deren Relationen zueinander modelliert, sondern auch konkrete Instanzen gruppiert und gefiltert werden können (siehe Kapitel 4.7). Da damit demnach sowohl Bausteindefinitionen und Instanzen, als auch Regeldefinitionen und konkrete Regeln formulierbar sind, ermöglicht dies ein flexibles Hinzufügen und Entfernen von Regeln und Bausteinen ohne dafür Programmcode schreiben zu müssen. Durch geschicktes Aufstellen von Einschränkungen an die Bausteinklassen kann ein Reasoner dazu verwendet werden, ein den Regeln entsprechendes Subset derjenigen Bausteine zu bilden, die in der derzeitig in der Ontologie angelegten Ausgangssituation platzierbar sind. Dies geschieht durch die Neuzuordnung einiger Bausteine zu einer speziell dafür definierten Klasse. Notwendige Informationen für eine generische Ausgangsklasse zur Beschreibung von Bausteinen sind:

- Eine eindeutige ID anhand derer für Bausteine aus der Ontologie die passenden Instanzen aus dem Programmcode identifiziert werden können.

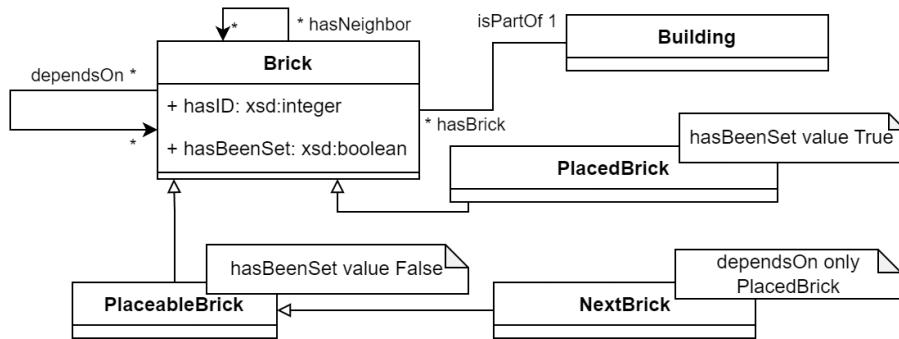


Abbildung 5.11: Klassendiagramm der Ontologie zur Bausteinbeschreibung.

- Nachbarschaftsrelationen zu anderen Bausteinen. Da Bausteine sechs Seiten haben, an welchen andere Bausteine anliegen können, ist eine Unterscheidung zwischen den verschiedenen Nachbarschaftsarten (untere, obere, linke, rechte, vordere und hintere Nachbarn) möglich.
- Eine *Abhängigkeits*-Eigenschaft zwischen Bausteinen, die angibt, an welchen Bausteinen jeder Baustein gebunden ist. Diese Relation kann durch eine entsprechende Regel manipuliert werden und dadurch die Schlussfolgerungen eines Reasoners direkt beeinflussen. Sie wird als Objekteigenschaft mit dem Namen *dependsOn* realisiert, deren Wertebereich (Range) Individuen der Klasse *Brick* sind.
- Einen Indikator *hasBeenSet* der angibt, ob der Baustein bereits gesetzt wurde oder nicht. Darüber kann ein Reasoner verstehen, welche benachbarten Bausteine über die *dependsOn*-Eigenschaft nicht mehr zu berücksichtigen sind.

Diese Klasse wird nachfolgend als *Brick* bezeichnet und ist in Abbildung 5.11 dargestellt. Um verschiedene Bauplanentwürfe gleichzeitig zu bearbeiten, kann eine Klasse *Building* definiert werden, die alle Bausteine des dazugehörigen Bauplanentwurfs referenziert. Eine Klasse *Rule* stellt die Grundlage zur Regeldefinition dar. Davon ausgehend können verschiedene Regelarten erstellt werden, die durch konkrete Individuen realisiert werden. Ein Beispiel wären Regeln, die eine Kardinalität zu einer ebenfalls in der Regel festgelegten Daten- oder Objekteigenschaft der Ontologie vorgeben (für Informationen zu Eigenschaften im Kontext von Ontologien siehe Kapitel 4.7). Damit könnte etwa die Menge aller Nachbarn eines Bausteins auf exakt, minimal oder maximal eine bestimmte Anzahl beschränkt werden. Eine weitere Art wären Regeln, die das Übernehmen einer bestimmten Eigenschaft auf alle über eine andere Eigenschaft assoziierten Bausteine voraussetzen. So könnten zum Beispiel alle über die *untere Nachbarn*-Eigenschaft verbundenen Bausteine eines konkreten Bausteins in dessen *dependsOn*-Eigenschaft übernommen werden. Notwendige Eigenschaften für diese Klasse sind:

- Eine Assoziation auf eine Objekt- oder Dateneigenschaft der Ontologie, die angibt auf welche Eigenschaft die Regel Einfluss nimmt. Dazu wurde die Eigenschaft `holdPropertyByName` eingeführt. Daraus werden die beiden Klassen *DataPropertyHolder* und *ObjectPropertyHolder* abgeleitet. Alle Instanzen dieser Klassen haben den

Namen einer Daten- oder Objekteigenschaft als String-Wert hinterlegt, denn Eigenschaften können im Kontext von Ontologien nicht direkt auf andere Eigenschaften verweisen.

- Eine Unterscheidung zwischen Regeln, die Kardinalitäten voraussetzen und jenen, die Eigenschaften übertragen. Nachfolgend werden diese Klassen als *CardinalityRule* und *ApplyObjectPropertyRule* bezeichnet.
- Für Kardinalitäts-vorgebende Regeln existieren die Eigenschaften *min*, *max* und *exactly*, welche auf einen Zahlenwert verweisen.
- Für Eigenschafts-übertragende Regeln existiert eine Eigenschaft, die ähnlich zu der aus dem obersten Punkt den Namen der „Zieleigenschaft“ enthält.

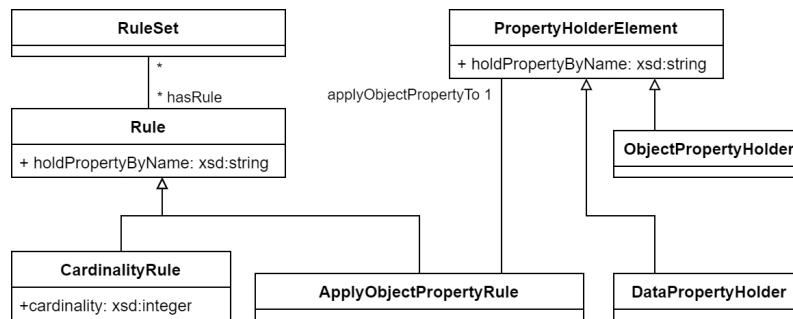


Abbildung 5.12: Klassendiagramm der Ontologie zur Regelbeschreibung.

Eine Übersicht über diese Klassenstruktur ist in Abbildung 5.12 zu sehen. Um zwischen verschiedenen Situationen wechseln zu können, die manchmal die Anwendung mehrerer Regeln voraussetzen, wird die Klasse *Ruleset* definiert, die lediglich auf beliebig viele Individuen der Rule-Klasse referenziert.

Mit diesem Aufbau wurde die Grundlage für einen Reasoner geschaffen, der die Individuen der Klasse Brick konkreteren Subklassen zuweist. Eine Übersicht über diese Klassen ist der Abbildung 5.11 zu entnehmen. Dabei stehen die Notizen, die an manche Klassen gehängt wurden, für die in der Ontologie hinterlegten Bedingungen, die für Individuen der jeweilige Klasse gelten müssen. So lässt sich über die Werte in der *hasBeenSet* Dateneigenschaft zwischen schon gesetzten und noch zu setzenden Brick-Individuen unterscheiden. Diese Klassen werden als *PlacedBrick* und *PlaceableBrick* bezeichnet und sind zwangsläufig disjunkt. Es können nur diejenigen Brick-Individuen, die ebenfalls der Klasse *PlaceableBrick* zugeordnet werden können, Teil des Subsets an Bausteinen sein, das in der derzeit gegebenen Situation platziert werden kann. Daraus lässt sich die erste Einschränkung für die als *NextBrick* bezeichnete Klasse aufstellen: Sie ist eine Subklasse der Klasse *PlaceableBrick*. Die *dependsOn*-Eigenschaft der Klasse *Brick* wird nun zur Überprüfung der „Legbarkeit“ eines Bausteins herangezogen. Sind für einen *PlaceableBrick* nur *PlacedBricks* oder keine Bricks über diese Eigenschaft referenziert, so gilt dieser als derzeit ablegbar und wird ebenfalls der Klasse *NextBrick* zugeordnet. Insgesamt sieht die Klassenbeschreibung der Klasse *NextBrick* schließlich wie folgt aus: *PlaceableBrick* and (*dependsOn* only *PlacedBrick*). Zu beachten ist allerdings die

*Open World Assumption*, nach welcher sich Ontologien richten (siehe Kapitel 4.7). Darum ist es notwendig bei der Befüllung der Ontologie zusammen mit den im Bauplanentwurf errechneten Baustein-Instanzen explizite Einschränkungen vorzugeben. Für jeden Baustein müssen alle konkreten Individuen definiert werden, die mit ihm über Eigenschaften verbunden sind. Zusätzlich muss angegeben werden, dass diese Menge an Bausteinen auch tatsächlich die einzige unveränderbare Menge darstellt, die über sämtliche Eigenschaften damit verbunden ist. Dieses Vorgehen ist unter der Bezeichnung *Closure Axiom* bekannt. Falls dies nicht vorgenommen wird, ist ein Reasoner nicht in der Lage ein PlaceableBrick-Individuum der Klasse NextBrick zuzuordnen, da es theoretisch sein könnte, dass an anderer Stelle ein bislang unbekannter PlaceableBrick existiert, der der *dependsOn*-Eigenschaft des Individuums zugeordnet werden kann.

Werden all die notwendigen Informationen zu den Bausteinen auf diese Weise und unter Einhaltung der definierten Regeln eines Regelsets in die Ontologie gespeist, kann durch Starten eines Reasoners das erste Subset aller platzierbaren Bausteine anhand der Zuordnung zu der Klasse NextBrick ausgelesen werden. Nun kann daraus schrittweise ein Baustein gewählt und als platziert markiert werden, sodass mit erneuten Starten des Reasoners ein neues Set an Individuen der Klasse NextBrick errechnet wird, das zu der veränderten Situation und den vorgegebenen Regeln passt.

# 6 Realisierung

Mit dem in Kapitel 4.1 vorgestellten *IFC Standard* existiert ein umfangreiches Framework zur Beschreibung sämtlicher für die Baubranche relevanter Daten. Im Zusammenspiel mit dem darauf aufbauenden *Building Information Modeling* wird ein einheitliches Verwaltungskonzept für das Planen, Bauen und Bewirtschaften von Infrastruktur vorgeschlagen (siehe Kapitel 4.3). Darum stellen im IFC Format vorliegende Gebäudepläne den Ausgangspunkt dieser Arbeit dar.

## 6.1 Modellierung

Dank den in Kapitel 4.2.2 vorgestellten Open-Source-Projekten zu den Technologien *IFC* und *BIM* sowie deren Anbindungen an die ebenfalls öffentliche 3D-Grafiksoftware Blender (siehe Kapitel 4.2) ist es möglich auf einen komplett kostenlosen Technologie-stack zur Modellierung von Gebäuden im IFC Format zurückzugreifen. Die Option Blender mithilfe sogenannter *Add-ons* an spezielle Projekt-abhängige Anforderungen anzupassen, macht diese Modellierungsumgebung noch zweckdienlicher. Ein solches Add-on ist zum Beispiel *blenderbim* (siehe Abschnitt 4.2.1). Dieses ermöglicht es ein IFC Projekt direkt in Blender entweder neu zu beginnen oder ein Vorhandenes zu bearbeiten. Für diese Arbeit relevant ist zunächst das Erstellen und Annotieren von Wänden beziehungsweise Wandtypen, sowie das Anbringen von Öffnungen. Beide Konzepte sind Teil des IFC Standards und wurden bereits ausführlich in Kapitel 4.1 vorgestellt. Blenderbim ermöglicht es in wenigen Schritten einen neuen *IfcWallType* zu definieren und daraus *IfcWall* Objekte zu erstellen. Diese können mit den von Blender nativ angebotenen Werkzeugen angepasst werden. Zusätzlich gibt es weitere hilfreiche, von Blenderbim eingeführte Modellierungsmöglichkeiten. Dank der Möglichkeit eigene *IfcPropertySets* (siehe Kapitel 4.1.2) zu definieren, lassen sich die in Kapitel 5.1.1 konzeptionierten Raster- und Modulinformationen zu den neu definierten Wandtypen hinzufügen. Bei Generierung eines *IfcWall* Objekts werden diese Properties ebenfalls an das neue Wandstück geknüpft. Damit können diese Informationen sowohl an das nachfolgende Wall Detailing übermittelt, als auch zur intuitiveren Modellierung des 3D Modells herangezogen werden, indem man Nutzern durch ein für diese Arbeit entwickeltes Add-on das für jeden Wandtyp zugeordnete Raster bei sämtlichen Transformationsvorgängen aufzwingt (siehe Abschnitt 5.1.1). Dadurch werden kleine Modellierungsfehler von vornherein vermieden.

## 6.2 Wall Detailing

In Abschnitt 5.2 wurde das Wall Detailing als der Vorgang, ein als geometrischer Körper definiertes Wandstück in ein konkretes Mauerwerk zu überführen, bezeichnet. Innerhalb

des IFC Standards werden einige mathematische/geometrische Repräsentationen der sogenannten *IfcWall* unterstützt, um neben einfachen Boxen auch komplexere Formen abbilden zu können. Beispielsweise ist es möglich in das Modell eines Hauses zunehmend dünner werdende Wandstücke, kurvige Wandstücke oder Wandstücke, welche nur durch ein arbiträres Vieleck beschrieben werden können zu integrieren. Standardmäßig haben *IfcWalls* aber einen einfachen Quader als Grundform, was für die Fallstudien dieser Arbeit ausreichend ist. Die Unterkapitel dieses Abschnitts folgen den in Kapitel 5.2 genannten Schritten, um von einer Menge solcher Wände hin zu einer Menge an Bausteinen zu gelangen, die den gewünschten Mauerwerksverband darauf anwenden.

### 6.2.1 Konvertieren und Filtern der Daten der IFC Datei

Den ersten Schritt stellt das Extrahieren aller notwendigen Daten aus dem vorliegenden IFC Modell dar. Für die Fallstudien dieser Arbeit sind sowohl alle Objekte des Typs *IfcWall* als auch die daran angeknüpften Objekte vom Typ *IfcOpeningElement* (siehe 4.1.4) von Interesse. Diese entstehen während der Modellierungsphase etwa durch Anbringen von Fenstern oder Türen an einer Wand. Zusätzlich werden aus den in den *IfcPropertySets* der Wandstücke hinterlegten Daten Informationen über das zu verwendende Modul und Raster ausgelesen. Mithilfe der Werkzeuge der in Kapitel 4 vorgestellten Python Bibliothek *IfcOpenShell* (siehe 4.2.2) ist dies in wenigen Schritten möglich.

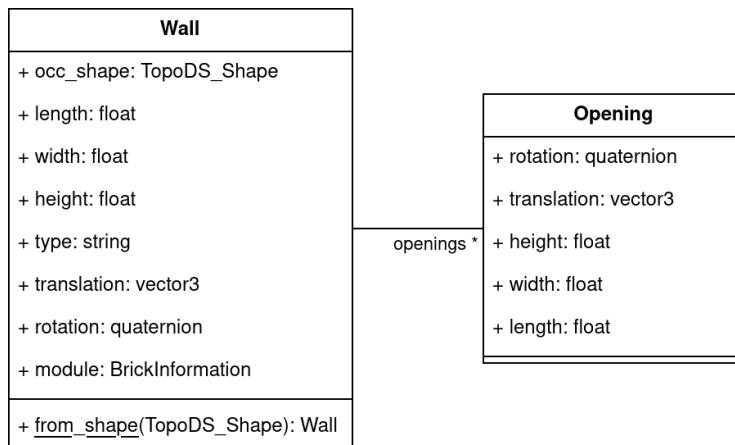


Abbildung 6.1: Klassendiagramm von der Klasse *Wall* und der Klasse *Opening*.

Zunächst werden aus jeder *IfcWall* ein Objekt der Klasse *Wall* erzeugt und anschließend mit deren in der IFC Datei angegebenen Öffnungen (*Opening*) versehen. Deren Zusammenspiel in dieser Arbeit kann dem Klassendiagramm aus Abbildung 6.1 entnommen werden und ist eine vereinfachte Version der Struktur innerhalb des IFC Modells. Um die globalen Transformationen der *IfcWalls* und *IfcOpenings* zu errechnen, muss der in Abbildung 4.1 gezeigten Klassenhierarchie nach oben gefolgt werden, da die in den Kindern eines Objekts angegebene Transformation relativ zu dessen Eltern angegeben ist. Im Anschluss werden die nun globalen Transformationen der Openings wiederum in eine relative Transformation zum davon betroffenen Wall-Objekt umgewandelt. Das erleichtert später die Berechnungen für das Anwenden der Öffnungen in einer Wand. Da sich diese

Arbeit zunächst ausschließlich mit quaderförmigen Wandstücken beschäftigt, müssen alle zuvor aus dem IFC Modell extrahierten Wandstücke auf diese Eigenschaft geprüft werden. Alle anders geformten *IfcWalls* werden in dieser Arbeit zunächst ignoriert. Somit ist gewährleistet, dass lediglich passende Wandstücke an die nachfolgenden Schritte weitergegeben werden.

### 6.2.2 Anwenden des Moduls

Mit dem zu jedem Wandstück festgelegten Modul werden nun alle Wandstücke in Schichten aufgeteilt. Diese Schichten werden von der Klasse *WallLayer* in Abbildung 6.2 repräsentiert und durch eine Instanz der Klasse *WallLayerGroup* gruppiert. Deren Höhe entspricht im Normalfall der Höhe des jeweiligen Moduls. Lediglich die oberste Schicht kann durch falsch modellierte Wandstücke eine niedrigere Schichthöhe aufweisen. Dies ist der Fall, wenn die Gesamthöhe des Wandstücks nicht exakt einem Vielfachen der Höhe des Moduls entspricht und ein nicht aufzuteilender Rest existiert. Das Aufteilen in Schichten erleichtert es im Anschluss Berechnungen an Wandstücken durchzuführen und Beziehungen zwischen ihnen zu finden.

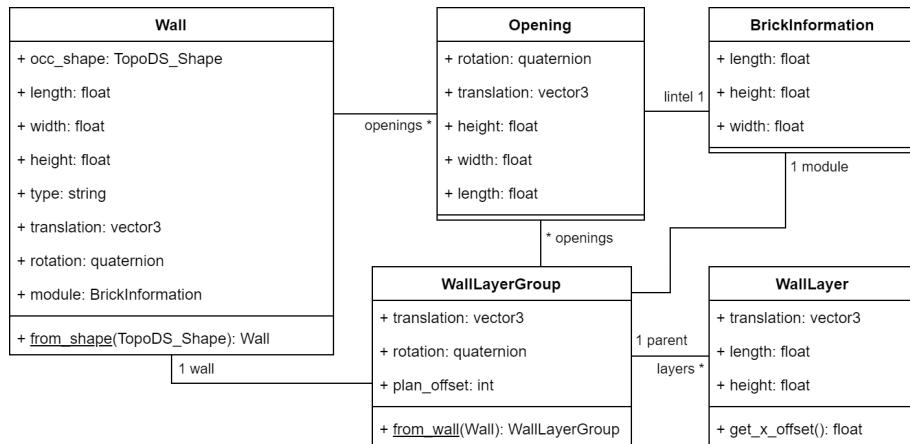


Abbildung 6.2: Klassendiagramm nach Anwenden des Moduls.

### 6.2.3 Kombinieren passender Wandstücke

Eine solche Beziehung stellen Wände dar, die, wie bereits in Kapitel 5.1.4 definiert, durch mehrere einzelne Objekte modelliert wurden, eigentlich aber eine Einheit bilden. Daher werden in diesem Schritt alle Wandstücke miteinander verglichen und eventuell kombiniert, sodass jeweils ein gefundenes Paar durch ein einzelnes Wandstück repräsentiert wird. Um zwei Wandstücke zu sinnvoll kombinieren zu können, müssen die Eigenschaften aus Kapitel 5.1.4 gelten. Allerdings kann Punkt 5, welcher Berührung oder Überlappung voraussetzt, mittlerweile wie folgt verschärft werden:

5. Mindestens eine Schicht des einen Wandstücks berührt, überlappt oder befindet sich exakt eine Modulhöhe ober- oder unterhalb einer Schicht des anderen Wandstücks.

In Abbildung 5.3 treten verschiedene Konstellationen von Wandstücken auf, die miteinander kombiniert werden müssen. Das aus Wandstück 1 und 2 gebildete Paar erfüllt alle oben genannten Eigenschaften und weist eine Teil-Überlappung auf. Somit müssen beide Wandstücke miteinander kombiniert werden. Den obersten Bereich füllen sowohl Wandstück 5 als auch Wandstück 6, sodass dort eine komplette Überlappung vorliegt. Auch diese beiden Wandstücke werden kombiniert, wobei dabei im Prinzip einfach eines verworfen wird. Zusätzlich muss dieser Bereich, wie auch Wandstück 7, mit Wandstück 2 kombiniert werden, da die beiden Paare sich an Ober- und Unterkante berühren. Einen weiteren Fall stellen seitliche, nicht überlappende Berührungen dar, wie sie zwischen Wandstück 2 und 3 zu sehen ist. Auch für dieses Paar sind alle Voraussetzungen zur Fusion erfüllt. Lediglich Wandstück 4 muss nicht mit dem Rest vereint werden, da dafür kein Paar existiert, das die obigen Eigenschaften erfüllt. Anhand von Abbildung 6.3 kann man erkennen, dass Wandstück 4 im weiteren Verlauf des Detailings tatsächlich unabhängig betrachtet werden kann.

Während dem Kombinieren von zwei Wänden wird ein Wandstück schichtweise in das Andere überführt. Dabei werden alle Schichten paarweise miteinander verglichen, um diejenigen Paare zu finden, die die Eigenschaften aus Punkt 5 erfüllen. Ein solches Paar wird dann wie folgt miteinander verschmolzen:

1. Errechne die kleinste und die größte relative x Koordinate der linken und rechten Ecken beider Schichten.
2. Die Differenz der beiden Werte entspricht der neuen Länge der resultierenden Schicht.
3. Die kleinste x Koordinate ist deren neue linke Eckkoordinate.
4. Mit dieser Eckkoordinate aus Punkt 3 und der halben neuen Länge kann nun der Mittelpunkt der resultierenden Schicht berechnet werden. Dieser entspricht der relativen Translation einer Schicht in Abhängigkeit der dazugehörigen *WallLayer-Group*.

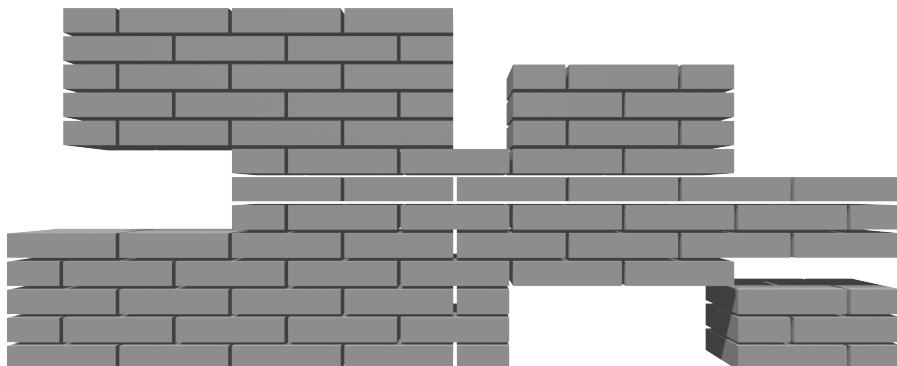


Abbildung 6.3: Ergebnis mit berücksichtigtem *x\_offset*.

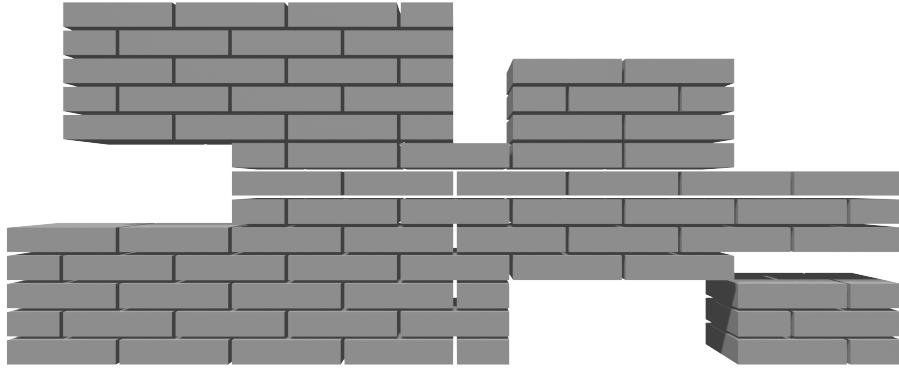


Abbildung 6.4: Ergebnis mit ignoriertem  $x\_offset$ .

Steht ein Wandstück in X-Richtung versetzt auf einem anderem, so ist es notwendig diesen Versatz während dem nachfolgenden Detailing zu berücksichtigen. Ignoriert man dies, kann das, durch Verletzung des vorgeschriebenen Überbindemaßes (siehe Abschnitt 4.5.2), zu den in Abbildung 6.4 gezeigten Fehlern innerhalb des Mauerwerksverbands führen (zum Beispiel zwischen Wandstück 2 und 7). Dieser, nachfolgend als  $x\_offset$  bezeichnete Versatz ist definiert als die Differenz zwischen der kleinsten lokalen X-Koordinate aller Schichten eines Wandstückes (mittlerweile repräsentiert durch eine *WallLayerGroup*) und der lokalen X-Koordinate der zu betrachtenden Schicht. Der daraus resultierende Wert wird später dazu verwendet den anzuwendenden Mauerwerksverband erst an passender Stelle zu beginnen. Dadurch erzielt man einen einheitlichen Verband über das gesamte Wandstück und verhindert den in Abbildung 6.4 gezeigten Fehlerfall. Eine weitere Eigenschaft, die aus dem Kombinieren mehrerer Wandstücke entstehen kann, ist das Vorhandensein unterbrochener Schichten oder, anders ausgedrückt, mehrerer Schichten auf einer Höhe innerhalb des resultierenden Wandstückes. Dies ist ebenfalls in Abbildung 5.3 zu sehen. Zwischen drei Schichten des Bereichs der Wandstücke 5 und 6 und des Wandstücks 7 ist eine Lücke. Durch das Einbeziehen des  $x\_offset$  können derartige Situationen jedoch ebenfalls gelöst werden, da für jedes Teilstück einer Schicht ein eigener  $x\_offset$  berechnet wird. Nach Vereinigung aller passenden Paare reduziert sich die Zahl der ursprünglichen Wandstücke aus dem Modell in Abbildung 5.3 von sieben auf zwei. Die korrekte Durchführung mit angewandtem  $x\_offset$  ist in Abbildung 6.3 zu sehen.

#### 6.2.4 Finden und Lösen von Beziehungen

Nun wird die aus dem vorherigen Schritt entstandene neue Menge an Wandstücken auf weitere Beziehungen untersucht. Für das Wall Detailing relevante Beziehungen stellen Ecken, T-Kreuzungen und X-Kreuzungen dar (siehe Kapitel 5.1.5). Der Grund dafür ist die Komplexität in diesen Bereichen die vorgeschriebenen Normen einzuhalten. Dazu zählt zum Beispiel das in Abschnitt 4.5.2 genannte Überbindemaß. In Abschnitt 5.1.4 wurden bereits die Eigenschaften aufgeführt, mithilfe derer sich die gesuchten Bezie-

## 6 Realisierung

---

hungen identifizieren lassen. Wichtige Informationen für jede dieser Beziehungen sind einerseits die Punkte, an denen sie sich im Raum befinden, andererseits die Art der Beziehung (also die Unterscheidung zwischen Ecke, T- oder X-Kreuzung). Die Position ist der Schnittpunkt der Richtungsvektoren beider Wandstücke entlang ihrer lokalen X-Achsen. Die Unterscheidung der Beziehungen kann mithilfe des Vorgehens, das ebenfalls in Abschnitt 5.1.4 erläutert wurde, durchgeführt werden. Dabei spielen die Anzahl der Wandstücke, die sich in einem Punkt schneiden und der Ort des Punktes innerhalb der beteiligten Wandstücke eine Rolle. Da ein Wandstück zu diesem Zeitpunkt schon als Menge an Schichten vorliegt, wird das Verfahren, wie es bereits oben getan wurde, paarweise auf alle Schichten angewandt. Somit werden für jede, sich durch Verkettungen über mehrere Wandstücke erstreckende Schicht alle Kreuzungs- und Eckbereiche ausfindig gemacht. Bevor aber eine neue Eckinstanz angelegt wird, muss überprüft werden, ob an dem errechneten Schnittpunkt bereits ein Eckbereich gefunden wurde. Falls dem so ist, kann die dazugehörige und bereits vorhandene Eckinstanz lediglich um die Schichten des neuen Paars erweitert werden. Dadurch kann im selben Schritt bereits die Zuweisung der Art der Beziehung vorgenommen werden. Durch die Eckinstanzen werden nun die bis dahin unabhängigen Wandstücke, welche durch den vorherigen Schritt bereits schichtweise und kombiniert vorliegen, miteinander verknüpft. Ein nützli-

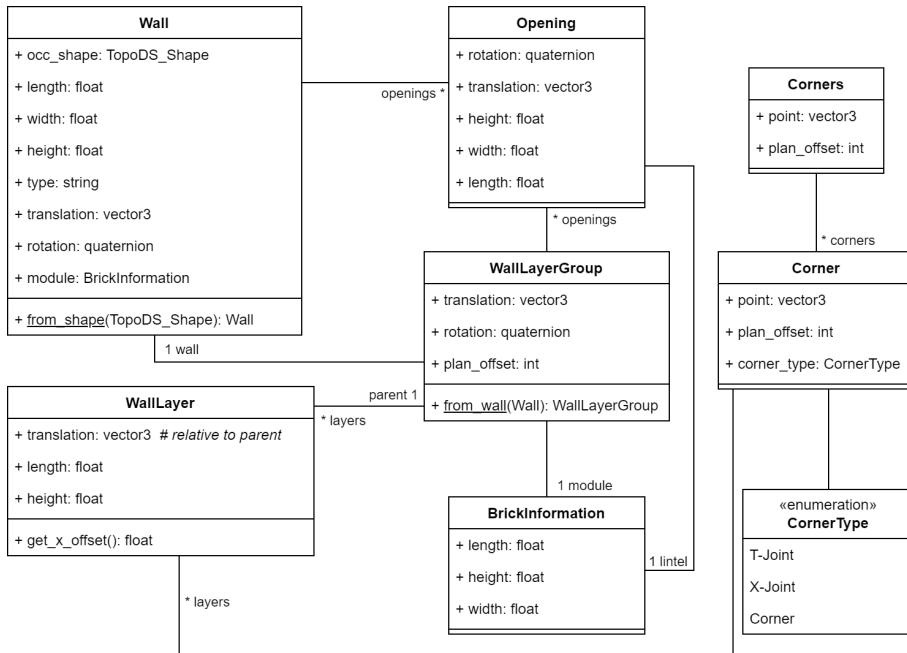


Abbildung 6.5: Klassendiagramm nach Errechnen der Eckbereiche.

cher Nebeneffekt der schichtweisen Betrachtung der Eckbereiche ist, dass nun auch zwei Wände mit unterschiedlichen Höhen, die gemeinsam einen Eckbereich bilden, korrekt dargestellt werden können, denn es werden nur für die Schichten Eckinstanzen erzeugt, die in beiden Wandstücken vorhanden sind. Die daraus resultierende neue Datenstruktur ist im Klassendiagramm in Abbildung 6.5 zu sehen. Objekte der Klasse *Corner* (bisher als Eckinstanzen bezeichnet) halten alle notwendigen Informationen zu jedem gefunden

Eckbereich einer Schicht, während ein Objekt der Klasse *Corners* dafür zuständig ist, die wachsende Liste an Eckinstanzen auf geteilte Schnittpunkte zu überprüfen. Wird eine Eckinstanz an einer schon vorhandenen Position im Raum in die Liste eingefügt, so wird der dort bereits liegende Eckbereich lediglich um die neuen *WallLayer* erweitert und falls nötig, dessen *CornerType* angepasst. Wie bereits erwähnt werden dabei die Voraussetzungen aus Kapitel 5.1.4 zur Unterscheidung der *CornerTypes* herangezogen.

## Lösen der Beziehungen

In Abschnitt 5.1.5 wurde die Problematik der Eckbereiche bereits ausführlich behandelt. Je nach gewählten Verband und anzuwendenden Modul können unterschiedliche Problemzonen beim Setzen der Bausteine auftreten. Das Ziel dieses Schritts ist es, die jeweiligen Eck- oder Kreuzungspläne der betroffenen Wandstücke so anzutragen, dass die dazwischenliegenden Wandbereiche unter Einhaltung des festgelegten Mauerwerksverbands lückenlos aufgefüllt werden können. Dies ist nur möglich, wenn die Längen der modellierten Wandstücke zum gewählten Verband passen. Unter der Annahme ein Modell ist mit den ausgewählten Modulen und Mauerwerksverbänden lückenlos baubar, führt das in Abschnitt 5.1.5 vorgestellte Vorgehen zu einer Lösung. Dabei werden die sogenannten *plan\_offsets* so manipuliert, bis eine lückenlose Lösung gefunden wurde. Sowohl Eckbereiche (siehe Klasse *Corner* im Klassendiagramm aus Abbildung 6.5) als auch die Wandstücke (mit der Klasse *WallLayerGroup* realisiert) besitzen dieses Attribut. Diese *plan\_offsets* geben an, mit welchem Index des angegebenen Eck- oder Wandstückplans in der untersten Schicht einer *WallLayerGroup* beim Berechnen der konkreten Bausteine begonnen werden muss.

Das in dieser Arbeit angewandte Verfahren stellt derzeit lediglich eine Verbesserung der Exhausionsmethode dar. In Kapitel 5.1.5 wurde bereits festgestellt, dass alle durch mindestens einen *Corner* verbundenen *WallLayerGroups* verkettet zusammenhängen. Daraus ergibt sich ein Abhängigkeitsgraph, der im Normalfall sämtliche Wandstücke eines Gebäudemodells miteinander verbindet. Durch Festlegen des *plan\_offset* eines einzigen Eckbereichs, können alle direkt und indirekt damit verbundenen Wandstücke und Eckbereiche daran angepasst werden. Denn ändert man den *plan\_offset* eines Eckbereichs, müssen alle darüber und darunterliegenden *Corner* ebenfalls angepasst werden, um die vorgeschriebene Reihenfolge des Planes weiterhin einzuhalten. Auch die Distanz, die ein Eckbereich in die daran anschließenden geraden Schichten ragt, kann sich dadurch verändern. Dies ist anhand des Beispiels in Abbildung 5.7 nachvollziehbar. Je nach Eckplankonfiguration sind die weißen Reste jeder Schicht unterschiedlich groß. Darum muss auch die *length* (ebenfalls Teil des Klassendiagramms in Abbildung 6.5) jeder Verbindungsschicht aller darüber und darunterliegenden *Corner* neu berechnet werden.

Das Anpassen eines geraden Reststücks an einen damit verbundenen *Corner* ist eine von zwei Operationen, die der Lösungsalgorithmus anwenden muss. Die Zweite ist das umgekehrte Anpassen eines *Corners* an ein bereits angepasstes gerades Reststück. Bei beiden Operationen werden diejenigen *plan\_offsets* aller Schichten der Ecke beziehungsweise des Wandstücks gesucht, die zu einer lückenlosen Lösung führen, ohne die Reihenfolge der Pläne zu verletzen. Die Startkonfiguration des Verfahrens ist lediglich ein Knoten des

Abhängigkeitsgraphen und ein daran angewandter *plan\_offset*. Davon ausgehend können diese Operationen abwechselnd auf die nächsten Nachbarknoten und die dazwischenliegenden Wandstücke angewendet werden, bis alle Knoten des Graphen abgearbeitet wurden. Findet sich mit einer Startkonfiguration keine valide Lösung, so wird eine neue gewählt; entweder durch Verändern des *plan\_offsets* oder dem Auswählen eines neuen Startknotens. So gibt es je nach Komplexität eines Modells immer noch einen großen Lösungsraum, dieser wurde aber mithilfe des Abhängigkeitsgraphen deutlich minimiert, da das Modell dadurch in sinnvoller Weise durchsucht wird und Szenarien, die ohnehin zu keiner Lösung geführt hätten, übersprungen werden. Falls keine lückenlose Lösung gefunden werden kann, wird diejenige mit dem insgesamt kleinsten Lückenvolumen zurückgegeben.

### 6.2.5 Anwenden der Öffnungen

Den nächsten Schritt stellt das Anwenden der Öffnungen sämtlicher *WallLayerGroups* dar. Diese werden durch die Klasse *Opening* im Klassendiagramm in Abbildung 6.5 realisiert. Auch diese Operation wird schichtweise durchgeführt. Auf jeder Schicht der betroffenen *WallLayerGroup* wird folgendes Verfahren angewandt:

1. Zunächst muss überprüft werden, ob die Öffnung innerhalb der betrachteten Schicht liegt. Dies kann aufgrund des geteilten Elternobjekts (einer *WallLayerGroup*) mit den relativen Koordinaten beider Objekte berechnet werden.
2. Falls Punkt 1 zutrifft, wird die Schicht in zwei Teile geteilt. Als Schnittpunkt wird der Mittelpunkt der Öffnung gewählt. Die ursprüngliche Schicht ist damit obsolet.
3. Nun muss ausgehend vom Schnittpunkt jeweils das rechte oder linke Ende der beiden neuen Schichten um jeweils die halbe Länge der Öffnung verschoben werden.

Oftmals werden über Öffnungen im Mauerwerk sogenannte Stürze eingesetzt. Das ist ein meist vorgefertigtes Bauteil aus Stahlbeton, welches etwas länger ist als die Öffnung selbst. Dieses Bauteil wird über eine Öffnung gelegt, um die Stabilität der Wand trotz Integration eines Fensters oder einer Tür zu gewährleisten. Stürze werden in dieser Arbeit wie eine eigene Bausteingröße behandelt und können aus diesem Grund als spezielles Modul betrachtet werden. Dies wird im Klassendiagramm in Abbildung 6.5 durch die Verbindung zwischen den Klassen *Opening* und *BrickInformation* dargestellt und als *lintel* bezeichnet. Der für den Sturz benötigte Platz kann ebenfalls mit dem Öffnungsverfahren von oben geschaffen werden.

### 6.2.6 Anwenden der Mauerwerksverbände

Die für Mauerwerksverbände entwickelte mathematische Darstellung aus Kapitel 5.1.3 wurde mithilfe der in Abbildung 6.7 dargestellten Klassenstruktur umgesetzt. Beispielhaft sind darin auch die den Kreuzverband und den Läuferverband repräsentierenden Klassen zu sehen. Letzterer kann zusätzlich durch Parameter angepasst werden, um zum Beispiel den Versatz zu verändern. Dabei ändert sich aber nicht nur der Legeplan für gerade

Wandabschnitte. Auch die speziellen Pläne für Ecken und Kreuzungen müssen dann den Parameterwerten entsprechend abgewandelt werden.

Durch die Vorbereitung der vorausgegangen Schritte, können nun Objekte der Klasse *Brick* anhand der vorgegebenen Mauerwerksverbände und den in den einzelnen Wandbereichen festgesetzten *plan\_offsets* erzeugt werden. Diese können mithilfe der Transformationen der jeweiligen *Corner* oder *WallLayerGroups* und den Informationen des Verbands passend positioniert und rotiert werden. Es gibt aber drei Besonderheiten, die neben dem bloßen Abarbeiten der jeweiligen Mauerwerksverbände zusätzlich zu beachten sind:

**Lokale Corner Rotationen** Im Falle von *Corners* ist neben der globalen Transformation noch eine weitere Information zu beachten. Bei T-Kreuzungen und Ecken ist die globale Rotation nicht ausreichend, um die Bausteine mithilfe des vorgegebenen Plans richtig zu platzieren. Es wird zusätzlich ein weitere lokale Rotation um die *Achse* einer T-Kreuzung oder Ecke benötigt. Dies soll in Abbildung 6.6 anhand einer Ecke veranschaulicht werden. Angenommen der Eckplan für den darauf abgebildeten Verband liegt in der Art der linken Ecke vor. So ist deutlich zu erkennen, dass das Anwenden desselben nicht rotierten Eckplans am zweiten Eckpunkt zu einem ungewünschten Ergebnis führen würde. Auch die beiden rechten Eckbereiche haben jeweils eine andere lokale Rotation um ihre, als Punkt dargestellte Achse. Um diese lokalen Eckrotationen zu berechnen wird für jede

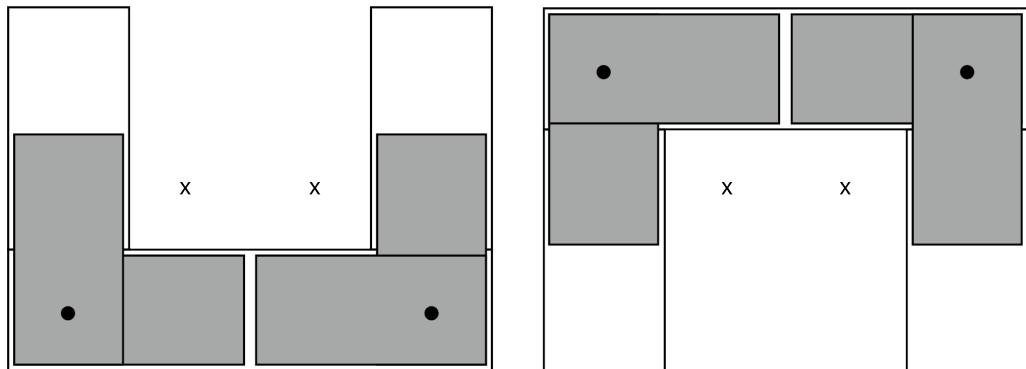


Abbildung 6.6: Die vier Rotationsmöglichkeiten eines Eckbereichs.

Ecke zunächst ein normierter Vektor in Richtung des Punktes  $x$  anhand der Mittelpunkte beider *WallLayer* und dem Eckpunkt gebildet. Dies entspricht sozusagen der *Richtung* der Ecke. Dieser Vektor wird dann um das Inverse der Rotation einer der am *Corner* beteiligten *WallLayerGroup* rotiert, um ihn in das globale Koordinatensystem zu überführen. Durch diese Rotation ist nun die Achse der Ecke gleich der z-Achse des Koordinatensystems. Deshalb kann mithilfe des Arkustangens der Quadrant herausgefunden werden, in dessen Richtung der Vektor zeigt. Da sowohl bekannt ist, dass der Vektor des Punktes  $x$  um  $45^\circ$  zur eigentlichen Eckrotation versetzt ist, als auch, dass die Rotation nur die z-Achse betrifft, kann mit dem Ergebnis des Arkustangens nun ein Quaternion erstellt werden, welches die spezielle lokale Eckrotation beschreibt. Auch mit T-Kreuzungen kann in ähnlicher Weise verfahren werden. Allerdings existieren hierbei statt vier nur

zwei Fallunterscheidungen.

**Abschluss eines Wandstücks** Wandenden sind mithilfe der *Corner* identifizierbar. Existiert kein *Corner* an einem Ende einer Schicht, handelt es sich entweder um ein Wandende oder den Außenbereich einer Öffnung. Dank den Informationen in den einzelnen Schichten eines Wandstücks, können die durch den Verband an diesen Bereichen gebliebenen Lücken kleinere Bausteine eingefügt werden, sofern dies gewünscht ist. Beispiele für aufgefüllte Wandenden sind in Abschnitt 5.1.7 zu sehen. Im Gegensatz dazu schließen die in Abbildung 4.5 vorgeführten Wandstücke nicht mit verkleinerten Bausteinen ab.

**Besondere Bausteine über Öffnungen** Wird von einer Öffnung ein Sturz definiert, so muss dieser ähnlich zu einem Baustein des Standardmoduls ebenfalls in einen konkreten *Brick* überführt werden. Die lokale Position innerhalb des Wandstücks liefert die Öffnung selbst. Die Dimensionen des Bausteins sind mithilfe des Moduls, das zur Beschreibung des Sturzes erstellt wurde, wie gewohnt vorgegeben.

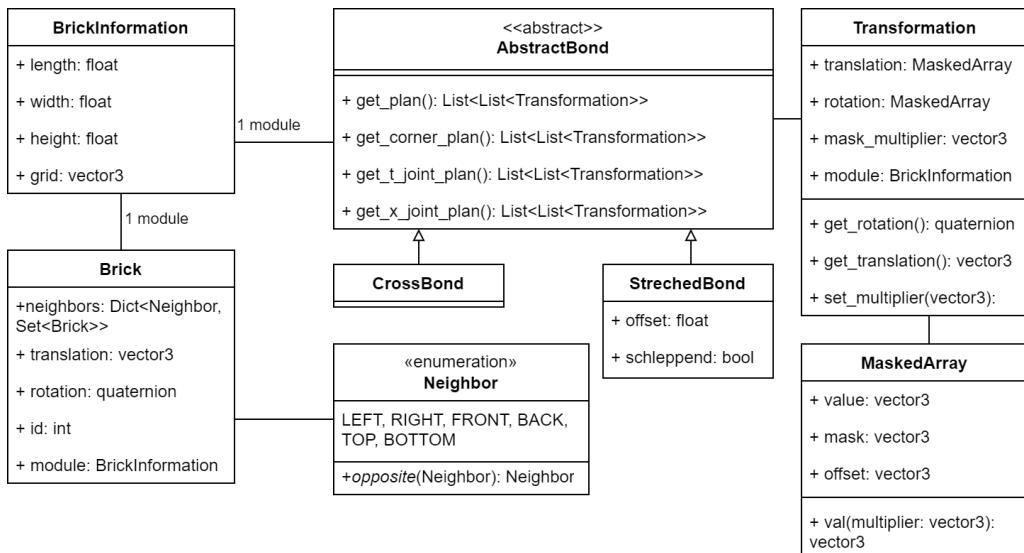


Abbildung 6.7: Klassendiagramm für die Mauerwerksverbände.

### 6.2.7 Nachbarschaftsbeziehungen zwischen Bausteinen

Um valide Nachbarschaften zwischen den berechneten Bausteinen festzustellen, werden die den Wandstücken zugeordneten Raster herangezogen. In Abbildung 6.8 ist der Effekt zweier unterschiedlicher Raster auf den gleichen Baustein dargestellt. Die gestrichelten Plätze sind potenzielle Nachbarfelder. Da es sich allerdings nicht um eine zwei-, sondern eine dreidimensionale Umgebung handelt, gibt es auch unter- und oberhalb der Bausteine mögliche Nachbarn. Um Bausteinen diese Informationen hinzuzufügen, wurde die Enumeration *Neighbor* und das *neighbors* Feld in der Klasse *Brick* erstellt. Dies ist im Klassendiagramm in Abbildung 6.7 zu sehen. Die Enumeration *Neighbor* definiert sechs mögliche Nachbarschaftsarten: linke, rechte, vordere, hintere, darüber und darunter

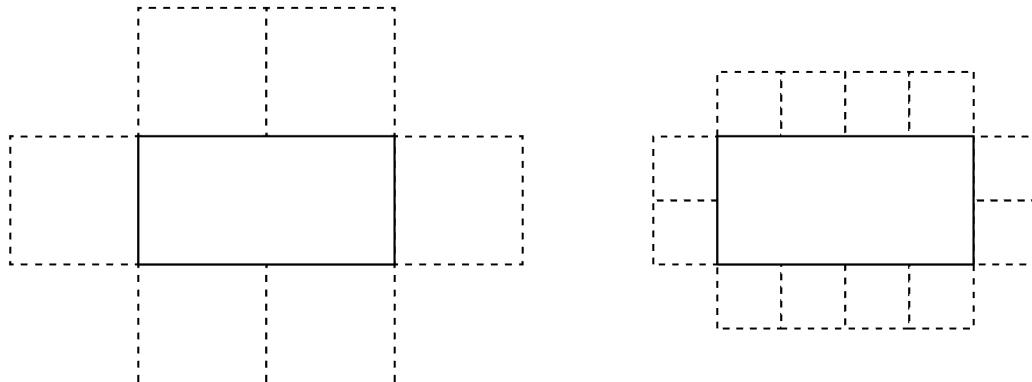
(a) Nachbarschaften in einem  $1 \times 1$  Raster. (b) Nachbarschaften in einem  $0.5 \times 0.5$  Raster.

Abbildung 6.8: Mögliche rasterabhängige Nachbarschaften.

liegende Nachbarn. Die Klasse *Brick* wird mit dem Feld *neighbors* um ein Dictionary ergänzt, welches zu jedem dieser Fälle eine Menge eindeutiger Bausteine beinhalten kann. Der Grund dafür ist, dass es einem Baustein je nach Raster möglich ist etwa mehrere linke Nachbarn zu haben (siehe Abbildung 6.8b). Nun kann für jeden Baustein durch den Vergleich mit allen anderen herausgefunden werden, welche Bausteine welche Nachbarfelder besetzen. Über die Funktion *opposite* der Enumeration *Neighbor*, kann im selben Zug oftmals auch den gefundenen Bausteinen ein Nachbar hinzugefügt werden. Das hängt allerdings von den lokalen Rotationen der Nachbarn ab.

### 6.2.8 Export

In Kapitel 5.3 wurde bereits erarbeitet, wie sich Regeln in einer Ontologie definieren lassen. Gleichzeitig wurde die Klasse *Brick* konzeptioniert, die alle notwendigen Informationen zu einem Baustein enthält, um diesen nach weiteren Eigenschaften zu analysieren. Sämtliche Klassen und Eigenschaften des Konzepts aus diesem Kapitel konnten mithilfe des Ontologie-Editors Protégé (siehe Kapitel 4.7.1) umgesetzt werden. Dank der Python Bibliothek *Owlready2* (siehe Kapitel 4.7.2) kann diese Ontologie leicht an den bisher entstandenen Programmcode angeschlossen werden. Zunächst werden die in der Ontologie erstellten Regel-Individuen ausgelesen und während der Befüllung der Ontologie mit *Brick*-Instanzen auf deren Eigenschaften angewandt. Insbesondere das Auffüllen der *dependsOn*-Eigenschaft anhand der zuvor ausgerechneten Nachbarschaften spielt hierbei eine wichtige Rolle. Owlready erlaubt das Wählen und Starten unterschiedlicher Reasoner. Nachdem die Berechnungen des Reasoners beendet sind, kann über eine Anfrage, welche alle Individuen einer bestimmten Klasse zurückgibt, die derzeitige Liste platzierbarer Bausteine ausgelesen werden. In Abbildung 6.9b ist das Objektdiagramm eines kleinen Beispiels zu sehen, in welchem bereits der erste und der zweite Baustein als platziert gekennzeichnet wurden. Darin enthalten sind 6 *Brick* Individuen, die durch einen Reasoner anhand ihrer Eigenschaften zu den dargestellten Klassen zugeordnet werden konnten. Da es in anderen Programmiersprachen womöglich schwieriger ist,

Daten aus Ontologien auszulesen, werden zusätzlich alle relevanten Informationen zu den Bausteinen in einer sogenannten JSON-Datei zur Verfügung gestellt. JSON (JavaScript Object Notation) ist ein weit verbreitetes, menschenlesbares Austauschformat, zu welchem in den meisten Programmiersprachen hilfreiche Bibliotheken existieren, die das Auslesen und Strukturieren von Daten aus diesen Dateien erleichtern [18]. Um eine direkte und nutzerfreundliche Alternative zu textuellen Ausgaben anzubieten, werden außerdem noch 3D Meshes erzeugt und als STL-Dateien exportiert. Ein Beispiel dafür ist in Abbildung 6.9a zu sehen. Dabei wird nicht nur ein einziges Mesh des gesamten Ergebnisses generiert, sondern auch von einigen wichtigen Zwischenergebnissen. Diese Meshes lassen sich in einem beliebigen 3D Viewer betrachten und auf Probleme untersuchen. So können Fehler in der Modellierung oder in der Implementierung leichter erkannt werden.

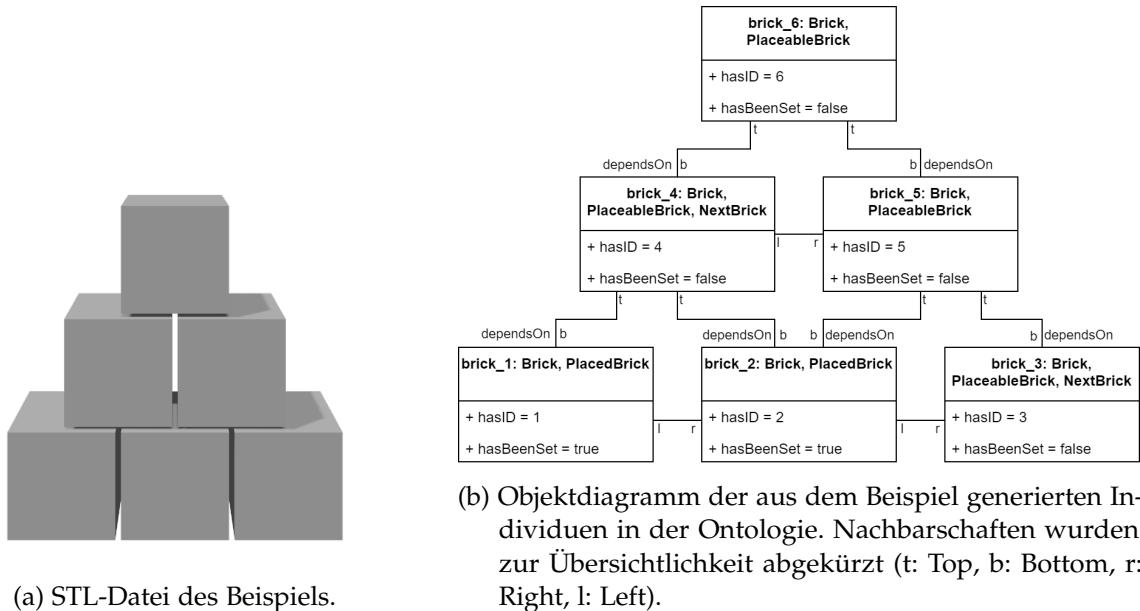


Abbildung 6.9: Zwei Exportformate für ein einfaches Beispiel mit 6 Bausteinen. Dabei wurden die unteren Nachbarn jedes Bausteins mithilfe einer entsprechenden Regel in dessen *dependsOn*-Eigenschaft übertragen (siehe Kapitel 5.3).

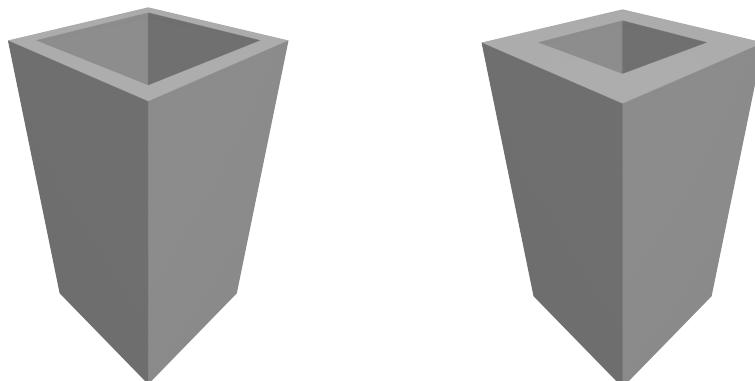
# 7 Proof of Concept

Nun wird das realisierte Konzept anhand der in Kapitel 2 vorgestellten Fallstudien auf Applikabilität geprüft. Aus den Ergebnissen dieses Kapitels kann im Anschluss ein Fazit gezogen und ein potenzieller Ausblick auf künftige Erweiterungen des Konzepts gegeben werden.

## 7.1 Von 3D Gebäudeplan zum Bauplanentwurf

Zunächst müssen für einzelnen Szenarien der ersten Fallstudie Gebäudepläne in einem Konstruktionsplaner modelliert werden. Dazu wurde, wie schon in Kapitel 6.1 beschrieben, die 3D Modellierungs-Software Blender herangezogen (siehe Kapitel 4.2). Diese kann mit der frei verfügbaren Erweiterung *blenderbim* zu einem funktionsfähigen BIM Editor mit IFC Unterstützung ausgebaut werden. Auch diese Technologien wurden bereits in den Kapiteln 4.2.1 und 4.1 vorgestellt. Darin können sowohl individuelle Wandtypen definiert, als auch Annotationen über Modul und Rastermaß mithilfe eigener *IfcPropertySets* daran angehängt werden. Mit den in Kapitel 5.1.1 definierten Modellierungseinschränkungen, deren Umsetzung in Kapitel 6.1 thematisiert wurde, konnte die Modellierungsphase sowohl erleichtert, als auch beschleunigt werden.

### 7.1.1 Szenario Turm



(a) Der Turm mit 1 Meter dicken Wänden.

(b) Der Turm mit 2 Meter dicken Wänden.

Abbildung 7.1: Die beiden Eingabemodelle des Turmes im IFC Format.

Die Modellierung der zwei sich in der Wanddicke unterscheidenden Versionen des 20 Meter hohen Turms mit einer Grundfläche von  $10 \times 10$  Metern in Blender war wenig

komplex. Es bedarf lediglich vier Wände, die einen quadratischen Raum bilden. Die vollständigen Gebäudemodelle sind in Abbildung 7.1 zu sehen. Das Ergebnis nach Anwenden der drei verschiedenen Mauerwerksverbände ist wie erwartet. Die verschiedenen Verbände lassen sich, wie in Kapitel 6.2.6 gezeigt, in den Programmcode einpflegen. Das Basismodul ist überall dort zu sehen, wo ein gerader Wandabschnitt vorliegt. Alle vier Eckbereiche sind korrekt identifiziert worden, denn dort wurde der für den jeweiligen Mauerwerksverband vordefinierte Eckplan passend umgesetzt. Im Falle des Kreuz- und Kopfverbandes wurde das Basismodul dabei etwas verkürzt. Die drei den Bauplanentwürfe werden als JSON-Dateien ausgegeben. Darin sind alle in Abschnitt 6.2.8 angegebenen Informationen enthalten. In Abbildung 7.2 sind die ebenfalls exportierten Meshes der drei mit Bausteinen realisierten Türme zu sehen.

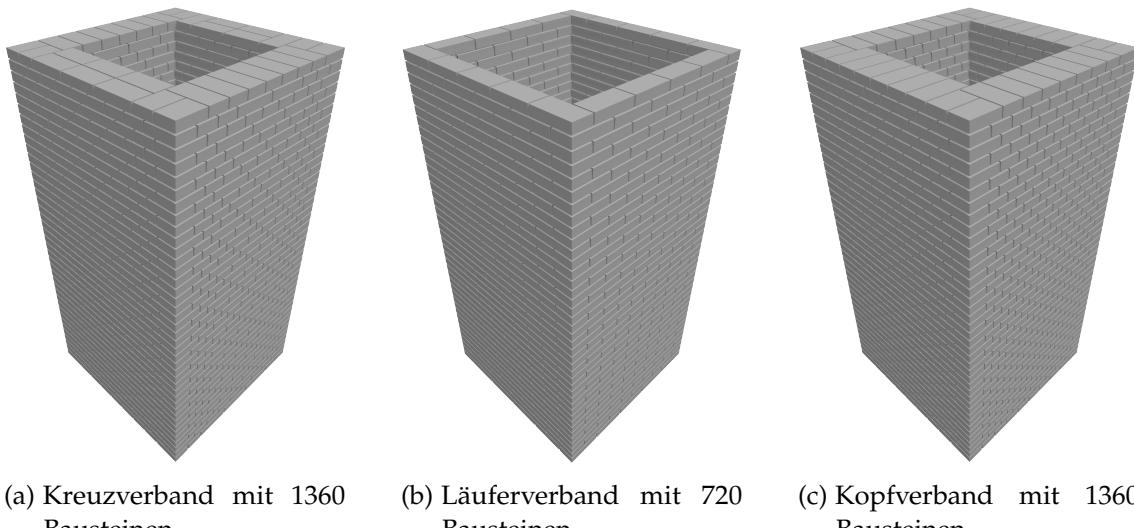


Abbildung 7.2: Ergebnisse des Wall Detailings mit unterschiedlichen Verbänden.

### 7.1.2 Szenario LEGO Klemmbausteine

Die Modellierung dieses Szenarios war aufgrund der Fenster und Türen ein wenig anspruchsvoller. Das dafür vorgesehene Vorgehen innerhalb der BIM Erweiterung für Blender ist zunächst etwas versteckt. Zumindest ist das Hinzufügen neuer Tür- und Fensterarten ist der Definition neuer Wandarten sehr ähnlich. Das finale Modell ist in Abbildung 7.3a zu sehen. In Kapiteln 5.1.6 und 6.2.5 wurde bereits aufgeführt, wie Öffnungen aus dem Modell extrahiert und als Teil des Wall-Detailings berücksichtigt werden. Das in Abbildung 7.3b zu sehende Ergebnis zeigt die korrekte Umsetzung der Fenster und Türbereiche. In Kapitel 4.5.2 wurde schon die sogenannte Stumpfstoßtechnik angesprochen. Dabei werden Anker dazu verwendet zwei Wandstücke miteinander zu verbinden, ohne dass diese verzahnt werden müssen. Da es für T- und X-Kreuzungen selbst bei Wandstücken mit gleichem Modul, Raster und Verband zu einer Vielzahl unterschiedlicher Kreuzungssituationen kommen kann, wurde zunächst darauf verzichtet dies in den Kreuzungslegeplänen der verschiedenen Verbände einzupflegen, obwohl das

mit dem erarbeiteten Konzept möglich wäre. Stattdessen wird sich an diesen Stellen auf die Stumpfstoßtechnik berufen und das algorithmische Auflösen der Kreuzungsbereiche als weiterführendes Konzept betrachtet. Wie bereits mehrfach in Kapitel 3.2 erwähnt, könnten dazu „intelligente“ beziehungsweise lernende Algorithmen eingesetzt werden.

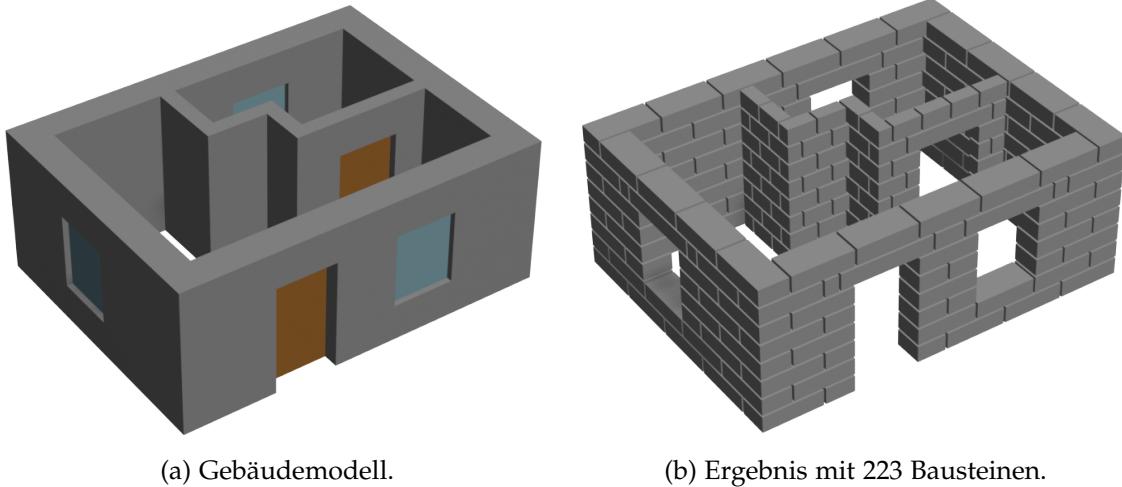


Abbildung 7.3: Eingabemodell und Ergebnis für das Szenario LEGO Klemmbausteine.

### 7.1.3 Szenario Fabrikgebäude

Dieses Szenario besteht aus insgesamt 36 einzelnen Wandstücken. Es gibt drei unterschiedliche Wandtypen. Vom ersten Typ, der eine 0.125 Meter dicke Wand aus 1 DF Steinen und halbversetztem Läuferverband vorschreibt, gibt es 20 Stück. Wände des zweiten Typs sind 0.25 Meter dick und werden im Kreuzverband mit 2 DF Steinen realisiert. Davon existieren 12 Stück. Für den Sockel des Kamins wurde ein eigener 0.5 Meter dicker Wandtyp definiert. Der Sockel soll mit Steinen im 16 DF Format und unter Anwendung des Kopfverbands erbaut werden. Darum gibt es von diesem Typ lediglich 4 Wände, auf welchen der eigentliche Kamin ruht. Außerdem wurden insgesamt 45 Fenster und 13 Türen an den Wänden angebracht. Um einen Dachgiebel anzudeuten, wurden niedrige und kürzer werdende Wandstücke auf den Mittelteil und dem seitlichen Anbau des Gebäudes gesetzt. Diese kleineren Wandstücke werden im Kombinationsschritt des Wall Detailings mit den großen darunterliegenden Wandstücken verschmolzen (siehe Abschnitt 6.2.3). Darum werden aus den 12 Wandstücken des zweiten Typs 4 und aus den 20 Wandstücken des ersten Typs 12. Dieser Schritt stellt zusammen mit dem Schritt des Findens und Lösens der Eckbereiche (siehe Abschnitt 6.2.4) den zeitaufwendigsten dar. Der Aufwand diese beiden Schritte zu berechnen verhält sich jeweils quadratisch zur Anzahl der einzelnen Schichten der beteiligten Wandstücke. Die Zeit zur Berechnung der konkreten Bausteine ist linear zur Anzahl der notwendigen Ziegel. In einem früheren Implementierungsversuch hatte das Finden von Nachbarschaftsbeziehungen zwischen Bausteinen ebenfalls eine quadratische Laufzeit in Abhängigkeit zur Anzahl der Bausteine und der Größe des angegebenen Rasters. Später konnte dieses Vorgehen

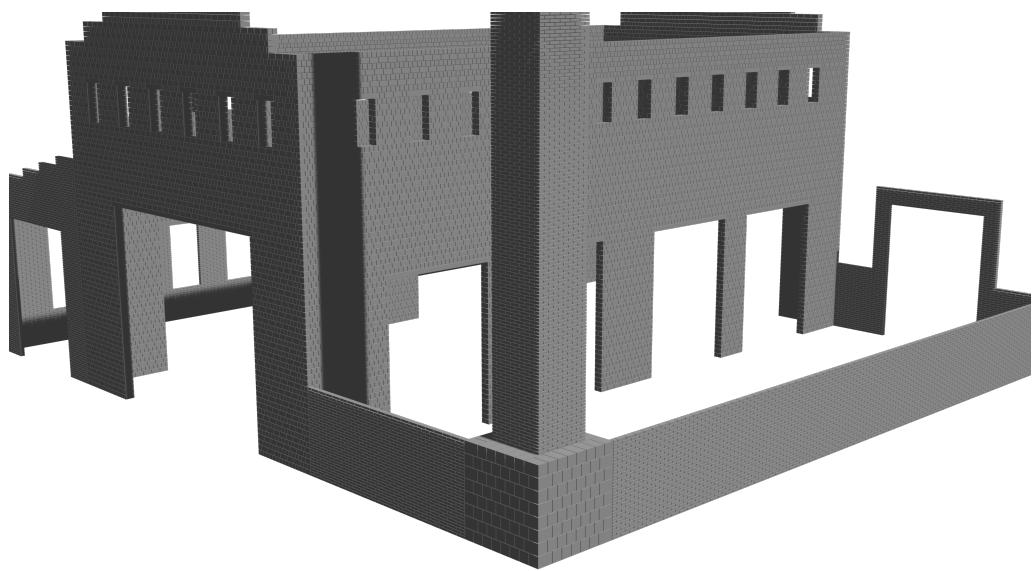
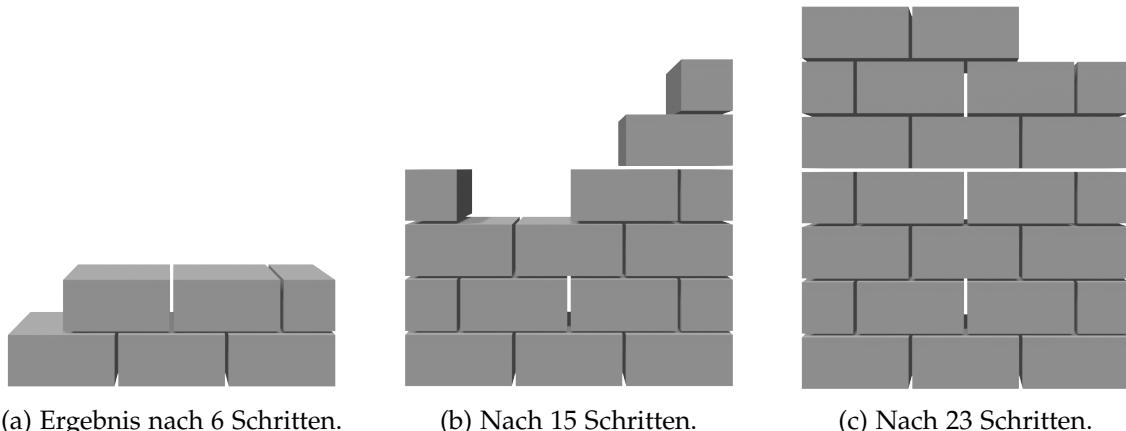


Abbildung 7.4: Ausschnitt des Ergebnisses von Szenario Fabrikgebäude mit 40409 Bausteinen.

aber optimiert und ebenfalls eine lineare Laufzeit erreicht werden. Ein ähnlicher Ansatz könnte für das Kombinieren und Lösen von Eckbereichen womöglich auch anwendbar sein.



(a) Ergebnis nach 6 Schritten. (b) Nach 15 Schritten. (c) Nach 23 Schritten.

Abbildung 7.5: Schrittweises Platzieren eines gefundenen NextBrick.

## 7.2 Regelbasierte Bauplandeduktion aus einem Bauplanentwurf

Mithilfe der in Kapitel 4.7 eingeführten Technologie einer Ontologie und der konkreten Umsetzung der Baustein- und Regeldefinitionen nach dem Konzept aus Abschnitt 5.3 konnte der resultierende Bauplanentwurf des Modells dieser Fallstudie in die dafür erstellte Ontologie eingepflegt werden. Das Konzept der Regeln wurde in dieser Fallstudie anhand der dafür erstellten *RuleDependsOnBottomNeighbor* Regel getestet. Diese soll alle nach unten benachbarten Bausteine eines Bausteins in dessen *dependsOn* Objekteigenschaft übertragen, sodass nur diejenigen Bausteine der Klasse *NextBrick* zugeordnet werden können, die keine Individuen der Klasse *PlaceableBrick* als untere Nachbarn besitzen. Dafür wurde für das Regel-Individuum über die Dateneigenschaft *holdPropertyByName* der Name der Ziel-Objekteigenschaft *hasBottomNeighbor* festgelegt. Über die Objekteigenschaft *applyObjectPropertyTo* wurde schließlich auf das dafür erstelle *ObjectPropertyHolder*-Individuum *dependsOnObjectPropertyHolder* verwiesen. Dieses wiederum hält lediglich den Namen der Objekteigenschaft *dependsOn*. Mit diesen Informationen kann nun während der Befüllung der Ontologie mit konkreten Bausteinen deren *dependsOn* Objekteigenschaften mit den Einträgen aus deren unteren Nachbarn gleichgesetzt werden.

Im Anschluss wurde durch wiederholtes Ausführen eines Reasoners und Auswählen einer zufälligen nächsten Instanz der Klasse *NextBrick* ein Bauplan erstellt, der die in Abbildung 2.4 gezeigte Wand von unten nach oben erbaut. In Abbildung 7.5 sind drei Zwischenergebnisse dieses Vorgehens zu sehen. Insgesamt enthält die Wand 28 Steine.



## 8 Fazit und Ausblick

Mit dem Ergebnis dieser Arbeit konnten erstaunlich viele Teilprobleme bei der Überführung eines 3D Modells eines Gebäudes in konkretes Mauerwerk identifiziert und für viele davon gute Lösungen gefunden werden. Vor allem durch die Unterstützung beliebiger Basismodule und das einfache Definieren neuer Mauerwerksverbände unterscheidet sich das erarbeitete Konzept von einigen in Kapitel 3 vorgestellten Veröffentlichungen. Mit der flexiblen Definition von Regelsets innerhalb einer Ontologie konnte der Grundstein für eine komplexe regelbasierte Analyse des errechneten Bauplanentwurfs gelegt werden. Dies konnte anhand von Fallstudien und diverser Szenarien belegt werden.

Dennoch gibt es selbstverständlich einige Erweiterungsmöglichkeiten. Insbesondere das programmatische Lösen kritischer Bereiche wie etwa T-Kreuzungen und anderen Situationen, die nicht zuvor in den beteiligten Mauerwerksverbänden eingepflegt wurden, stellt ein interessantes aber komplexes Problem dar. Hier bietet sich das Verwenden von Optimierungsansätzen aus der Domäne des *Bin Packings* oder das darauf gezielte Trainieren von *Machine Learning Algorithmen* an. Damit könnten eventuell auch Probleme wie abgerundete Ecken und Ecken, die keinen 90° Winkel aufweisen umgesetzt werden. Der bereits angesprochene Anwendungsfall nicht quadratischer Wandstücke kann mit dem vorliegenden Grundgerüst der schichtweisen Betrachtung einer Wand ebenfalls nachträglich in das Konzept integriert werden. Die Definition eines Wandabschnitts durch ein arbiträres Vieleck lässt sich leicht durch „Ausstanzen“ dieses Vielecks aus einem bereits mit Bausteinen aufgefüllten, rechteckigen Wandstück realisieren. Eine Erweiterung um eine simulative Komponente ist ebenfalls denkbar. Darin könnten etwa Statik-Berechnungen durchgeführt werden, um potenziell fehlerhafte Bereiche im Gebäudemodell frühzeitig ausfindig zu machen. Derzeit wird die Fugengröße zwischen Bausteinen durch Abzug eines Bruchteils der Bausteingröße dynamisch errechnet. Da alle Berechnungen ohnehin auf Basis des angegebenen Rastermaßes eines Bausteins getätigten werden, verfälscht dies zwar nicht die errechneten Ergebnisse, aber die exportierten Meshes entsprechen dadurch nicht exakt der Realität. Es ist möglich diese zusätzliche Information den Bausteindefinitionen hinzuzufügen und bei der Generierung der Meshes zu berücksichtigen. Trotz dieser nützlichen Erweiterungsmöglichkeiten bietet allein der derzeitige Stand dieser Arbeit schon einen umfangreichen Startpunkt für das automatisierte Erbauen von Gebäuden. Die strukturierten Baupläne, die das Ergebnis dieser Arbeit darstellen, können in weiterführenden Projekten etwa auf parallelisierbare Abläufe untersucht werden. Mithilfe dieser Abläufe wäre es möglich, das geplante Gebäude durch die Koordination mehrerer Roboter schneller zu erbauen, als es mit den herkömmlichen Verfahren der additiven Fertigung derzeit möglich ist. Damit lässt sich wiederum die Planungsphase des in Kapitel 1 erwähnten Projekts eines heterogenen Roboterschwarmes realisieren, dessen bodengebundene Lege-Roboter sich auf den Mauern des geplanten Gebäudes bewegen, während sie diese gemeinsam errichten.



# Abbildungsverzeichnis

1.1	Während wirtschaftliches Risiko durch eine potenziell sinkende Nachfrage nach Bauaufträgen und eventuell steigender Arbeitskosten unverändert blieb oder sogar als weniger relevant bewertet wurde, ist ein deutlicher Anstieg aufgrund des vorherrschenden Fachkräftemangels und der zunehmenden Energie- und Rohstoffpreise zu erkennen. . . . .	1
2.1	Grundrisse der beiden Türme. . . . .	4
2.2	Grundriss des LEGO-Gebäudes mit einer Grundfläche von 19.2 auf 14.4 Zentimetern. . . . .	6
2.3	IFC Modell des Fabrikgebäudes aus zwei Perspektiven. . . . .	7
2.4	Bauplanentwurf des Experiments zur Bauplandeduktion unter Berücksichtigung von Regeln. . . . .	8
3.1	Ergebnis des Verfahrens zur Erstellung eines Ziegel-Legeplans nach Usmanov et al. [29]. . . . .	10
3.2	Eingaben und Ergebnisse des <i>Brick Pattern Generator</i> nach Bárbara Andrade Zandavali et al. [25]. . . . .	12
4.1	Klassenhierarchie am Beispiel der Klasse <i>IfcWall</i> . . . . .	16
4.2	Relation der <i>IfcWall</i> und einem <i>IfcProject</i> . . . . .	16
4.3	Darstellung verschiedener Steinformate nach DIN 4172 (Baunennmaß in Millimetern) [64]. . . . .	20
4.4	Besondere Eigenschaften der oktametrischen Maßordnung [27]. . . . .	21
4.5	Typische Mauerwerksverbände. In diesem Beispiel weisen die Läuferverbände in 4.5c und 4.5d einen Versatz von 1/4 der Steinlänge auf. . . . .	22
4.6	Lösung einer Kreuzung am Beispiel einer „Wand aus 2 DF im Kreuz- und Blockverband“ [48]. . . . .	23
4.7	Lösungen für Ecken und T-Kreuzungen unterschiedlicher Verbände [27].	24
4.8	Maße des Standard 2×4 LEGO Steins [60]. . . . .	24
5.1	Der Läuferverband mit einem Versatz von 50 %. . . . .	32
5.2	Eckpläne für den Läuferverband mit einem Versatz von 50%. . . . .	33
5.3	Modell einer Wand, die durch 7 einzelne Wandstücke desselben Typs gebildet wird, deren Mittelpunkte alle auf einer Ebene liegen. . . . .	33
5.4	Draufsicht auf Varianten der Modellierung einer Ecke. . . . .	35
5.5	Draufsicht auf Varianten der Modellierung einer T-Kreuzung. . . . .	35
5.6	Draufsicht auf Varianten der Modellierung einer X-Kreuzung. . . . .	35
5.7	Draufsicht auf eine Verkettung von 4 Wandstücken durch 3 dazwischenliegende Ecken. . . . .	37

## *Abbildungsverzeichnis*

---

5.8	Draufsicht auf eine unzulässige Lösung der Eckplankonfiguration. . . . .	37
5.9	Eine mögliche Lösung und die durch die Eckpläne entstehende zweite Lösung für die darüber liegende Schicht. . . . .	37
5.10	Wandenden verschiedener Mauerwerksverbände. . . . .	40
5.11	Klassendiagramm der Ontologie zur Bausteinbeschreibung. . . . .	42
5.12	Klassendiagramm der Ontologie zur Regelbeschreibung. . . . .	43
6.1	Klassendiagramm von der Klasse <i>Wall</i> und der Klasse <i>Opening</i> . . . . .	46
6.2	Klassendiagramm nach Anwenden des Moduls. . . . .	47
6.3	Ergebnis mit berücksichtigtem <i>x_offset</i> . . . . .	48
6.4	Ergebnis mit ignoriertem <i>x_offset</i> . . . . .	49
6.5	Klassendiagramm nach Errechnen der Eckbereiche. . . . .	50
6.6	Die vier Rotationsmöglichkeiten eines Eckbereichs. . . . .	53
6.7	Klassendiagramm für die Mauerwerksverbände. . . . .	54
6.8	Mögliche rasterabhängige Nachbarschaften. . . . .	55
6.9	Zwei Exportformate für ein einfaches Beispiel mit 6 Bausteinen. Dabei wurden die unteren Nachbarn jedes Bausteins mithilfe einer entsprechenden Regel in dessen <i>dependsOn</i> -Eigenschaft übertragen (siehe Kapitel 5.3). . .	56
7.1	Die beiden Eingabemodelle des Turmes im IFC Format. . . . .	57
7.2	Ergebnisse des Wall Detailings mit unterschiedlichen Verbänden. . . . .	58
7.3	Eingabemodell und Ergebnis für das Szenario LEGO Klemmbausteine. .	59
7.4	Ausschnitt des Ergebnisses von Szenario Fabrikgebäude mit 40409 Bausteinen. . . . .	60
7.5	Schrittweises Platzieren eines gefundenen NextBrick. . . . .	60

# Literatur

- [1] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5 (2 Juni 1993), S. 199–220. issn: 10428143. doi: 10.1006/knac.1993.1008.
- [2] Rudi Studer, V.Richard Benjamins und Dieter Fensel. „Knowledge engineering: Principles and methods“. In: *Data and Knowledge Engineering* 25 (1-2 März 1998), S. 161–197. issn: 0169023X. doi: 10.1016/S0169-023X(97)00056-6.
- [3] Tim Berners-Lee, James Hendler und Ora Lassila. „The Semantic Web“. In: *Scientific American* 284 (5 Mai 2001), S. 34–43. issn: 0036-8733. doi: 10.1038/scientificamerican0501-34.
- [4] Evren Sirin u. a. „Pellet: A practical OWL-DL reasoner“. In: *Journal of Web Semantics* 5 (2 Juni 2007), S. 51–53. issn: 15708268. doi: 10.1016/j.websem.2007.03.004.
- [5] Nicola Guarino, Daniel Oberle und Steffen Staab. „Handbook on Ontologies“. In: Springer Berlin Heidelberg, 2009, S. 1–17. doi: 10.1007/978-3-540-92673-3.
- [6] Yusuf Arayici und Ghassan Aouad. *Building information modelling (BIM) for Construction Lifecycle Management*. Okt. 2010. url: <https://www.researchgate.net/publication/243972464>.
- [7] OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation. <https://www.w3.org/TR/2012/REC-owl2-overview-20121211/>. W3C, Dez. 2012.
- [8] Kirstin Petersen, Radhika Nagpal und Justin Werfel. „TERMES: An Autonomous Robotic System for Three-Dimensional Collective Construction“. In: The MIT Press, Juni 2012, S. 257–264. doi: 10.7551/mitpress/9481.003.0038. url: <https://direct.mit.edu/books/book/3728/chapter/123555/TERMES-An-Autonomous-Robotic-System-for-Three>.
- [9] Barry Smith. „Ontology“. In: Leiden, Niederlande: Brill, 2012, S. 47–68. isbn: 9789401207799. doi: [https://doi.org/10.1163/9789401207799\\_005](https://doi.org/10.1163/9789401207799_005).
- [10] Steven Harris und Andy Seaborne. SPARQL 1.1 Query Language. W3C Recommendation. <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>. W3C, März 2013.
- [11] Lieyun Ding, Ying Zhou und Burcu Akinci. „Building Information Modeling (BIM) application framework: The process of expanding from 3D to computable nD“. In: *Automation in Construction* 46 (2014), S. 82–93. issn: 09265805. doi: 10.1016/j.autcon.2014.04.009.
- [12] Birte Glimm u. a. „HermiT: An OWL 2 Reasoner“. In: *Journal of Automated Reasoning* 53 (3 Okt. 2014), S. 245–269. issn: 0168-7433. doi: 10.1007/s10817-014-9305-1.

- [13] Yevgeny Kazakov, Markus Krötzsch und František Simančík. „The Incredible ELK“. In: *Journal of Automated Reasoning* 53 (1 Juni 2014), S. 1–61. ISSN: 0168-7433. doi: 10.1007/s10817-013-9296-3.
- [14] Markus Lanthaler, Richard Cyganiak und David Wood. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. W3C, Feb. 2014.
- [15] Kim Noergaard Jensen, Kjeld Nielsen und Thomas Ditlev Brunoe. „Application of mass customization in the construction industry“. In: Bd. 459. Springer New York LLC, 2015, S. 161–168. ISBN: 9783319227559. doi: 10.1007/978-3-319-22756-6\_20.
- [16] Xiao Liu u. a. „HAPE3D—a new constructive algorithm for the 3D irregular packing problem“. In: *Frontiers of Information Technology and Electronic Engineering* 16 (5 Mai 2015), S. 380–390. ISSN: 20959230. doi: 10.1631/FITEE.1400421.
- [17] Mark A. Musen. „The protégé project“. In: *AI Matters* 1 (4 Juni 2015), S. 4–12. ISSN: 2372-3483. doi: 10.1145/2757001.2757003. URL: <https://dl.acm.org/doi/10.1145/2757001.2757003>.
- [18] Felipe Pezoa u. a. „Foundations of JSON schema“. In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, S. 263–273.
- [19] Jean Baptiste Lamy. „Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies“. In: *Artificial Intelligence in Medicine* 80 (Juli 2017), S. 11–28. ISSN: 18732860. doi: 10.1016/j.artmed.2017.07.002.
- [20] Tarek Zaki, Khaled Nassar und Osama Hosny. „Parametric Blockwall-Assembly Algorithms for the Automated Generation of Virtual Wall Mockups Using BIM“. In: American Society of Civil Engineers, Apr. 2017, S. 844–854. ISBN: 9780784480502. doi: 10.1061/9780784480502.071. URL: <http://ascelibrary.org/doi/10.1061/9780784480502.071>.
- [21] Daniel Delgado Camacho u. a. „Applications of additive manufacturing in the construction industry; A forward-looking review“. In: *Automation in Construction* 89 (Mai 2018), S. 110–119. ISSN: 09265805. doi: 10.1016/j.autcon.2017.12.031.
- [22] Kim Noergaard Jensen u. a. „Productivity, Challenges, and Applying Mass Customization in the Building and Construction Industry“. In: Bd. 113. Springer Science und Business Media B.V., 2018, S. 551–565. ISBN: 9783319775555. doi: 10.1007/978-3-319-77556-2\_34.
- [23] Maria Stoestrup Schioenning Larsen u. a. „Mass Customization in the House Building Industry: Literature Review and Research Directions“. In: *Frontiers in Built Environment* 5 (Okt. 2019). ISSN: 22973362. doi: 10.3389/fbuil.2019.00115.
- [24] Kirstin H. Petersen u. a. „A review of collective robotic construction“. In: *Science Robotics* 4 (28 März 2019). ISSN: 2470-9476. doi: 10.1126/scirobotics.aau8479. URL: <https://www.science.org/doi/10.1126/scirobotics.aau8479>.

- [25] Bárbara Andrade Zandavali und Manuel Jiménez García. „Automated Brick Pattern Generator for Robotic Assembly using Machine Learning and Images“. In: Editora Blucher, Dez. 2019, S. 217–226. doi: 10.5151/ proceedings-ecaadesigradi2019\_605. URL: <http://www.proceedings.blucher.com.br/article-details/34359>.
- [26] Trayana Tankova und Luís Simões da Silva. „Robotics and Additive Manufacturing in the Construction Industry“. In: *Current Robotics Reports* 1 (1 März 2020), S. 13–18. doi: 10.1007/s43154-020-00003-8.
- [27] José Luis Moro. „MASSORDNUNG“. In: *Baukonstruktion – vom Prinzip zum Detail: Band 1 Grundlagen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, S. 65–97. ISBN: 978-3-662-64155-2. doi: 10.1007/978-3-662-64155-2\_4. URL: [https://doi.org/10.1007/978-3-662-64155-2\\_4](https://doi.org/10.1007/978-3-662-64155-2_4).
- [28] SAM. [Online; accessed 15. Dec. 2023]. Juli 2021. URL: <https://www.construction-robotics.com/sam-2>.
- [29] Vjačeslav Usmanov, Jan Illetško und Rostislav Šulc. „Digital plan of brickwork layout for robotic bricklaying technology“. In: *Sustainability (Switzerland)* 13 (7 Apr. 2021). issn: 20711050. doi: 10.3390/su13073905.
- [30] Chengran Xu u. a. „Optimal brick layout of masonry walls based on intelligent evolutionary algorithm and building information modeling“. In: *Automation in Construction* 129 (Sep. 2021), S. 103824. issn: 0926-5805. doi: 10.1016/J.AUTCON.2021.103824.
- [31] Kathrin Dörfler u. a. „Additive Manufacturing using mobile robots: Opportunities and challenges for building construction“. In: *Cement and Concrete Research* 158 (Aug. 2022), S. 106772. issn: 0008-8846. doi: 10.1016/J.CEMCONRES.2022.106772.
- [32] Qiruyi Zuo, Xinglu Liu und Wai Kin Victor Chan. „A Constructive Heuristic Algorithm for 3D Bin Packing of Irregular Shaped Items“. In: 2022, S. 393–406. doi: 10.1007/978-3-031-15644-1\_29. URL: [https://link.springer.com/10.1007/978-3-031-15644-1\\_29](https://link.springer.com/10.1007/978-3-031-15644-1_29).
- [33] 4.1.7.4.3 Product Local Placement - IFC4.3.2.0 Documentation. (Accessed on 12/11/2023). Dez. 2023. URL: [https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/concepts/Product\\_Shape/Product\\_Placement/Product\\_Local\\_Placement/content.html](https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/concepts/Product_Shape/Product_Placement/Product_Local_Placement/content.html).
- [34] Hadrian X® | Outdoor Construction & Bricklaying Robot from FBR. [Online; accessed 15. Dec. 2023]. Dez. 2023. URL: <https://www.fbr.com.au/view/hadrian-x>.
- [35] Hermit Reasoner: Home. <http://www.hermit-reasoner.com>. (Accessed on 12/03/2023). Dez. 2023.
- [36] Ontologie - Fakultät für Philosophie, Wissenschaftstheorie und Religionswissenschaft - LMU München. <https://www.philosophie.uni-muenchen.de/fakultaet/schwerpunkte/ontologie/index.html>. (Accessed on 12/02/2023). Dez. 2023.
- [37] Python.org. <https://www.python.org>. (Accessed on 11/30/2023). Nov. 2023.
- [38] 05\_maurerfibel\_kap-4.pdf. [https://www.kalksandstein.de/media/08\\_downloadcenter/05\\_maurerfibel\\_kap-4.pdf](https://www.kalksandstein.de/media/08_downloadcenter/05_maurerfibel_kap-4.pdf). (Accessed on 06/29/2023).

- [39] *API Overview - Blender Python API*. <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcOpeningElement.htm>. (Accessed on 11/30/2023).
- [40] *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](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).
- [41] *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).
- [42] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. <https://www.blender.org/>. (Accessed on 02/16/2023).
- [43] *BlenderBIM Add-on - beautiful, detailed, and data-rich OpenBIM*. <https://blenderbim.org/>. (Accessed on 02/16/2023).
- [44] *Building Information Modeling – Wikipedia*. [https://de.wikipedia.org/wiki/Building\\_Information\\_Modeling](https://de.wikipedia.org/wiki/Building_Information_Modeling). (Accessed on 02/16/2023).
- [45] I Caddy und M & Callan. *From Mass Production to Mass Customization: Impact on Integrated Supply Chains*.
- [46] Charles M. Eastman u. a. „BIM handbook : a guide to building information modeling for owners, managers, designers, engineers and contractors“. In: Kap. 3.4.4. ISBN: 9781119287544.
- [47] *Fachkräftemangel und Rohstoffpreise – Die Deutsche Bauindustrie*. <https://www.bauindustrie.de/zahlen-fakten/bauwirtschaft-im-zahlenbild/fachkraeftemangel-und-rohstoffpreise>. (Accessed on 03/31/2023).
- [48] Kalksandstein-Dienstleistung GmbH. *Mauern von Stößen und Kreuzungen*. <https://www.ks-maurerfibel.de/maurerfibel/4-mauerwerksverbaende/4-7-mauern-von-stoessen-und-kreuzungen/>. (Accessed on 12/01/2023).
- [49] *IFC Formats - buildingSMART Technical*. <https://technical.buildingsmart.org/standards/ifc/ifc-formats/>. (Accessed on 02/16/2023).
- [50] IfcOpenShell. *IfcOpenShell - The open source IFC toolkit and geometry engine*. <https://ifcopenshell.org/>. (Accessed on 05/05/2023).
- [51] *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).
- [52] *Industry Foundation Classes – Wikipedia*. [https://de.wikipedia.org/wiki/Industry\\_Foundation\\_Classes](https://de.wikipedia.org/wiki/Industry_Foundation_Classes). (Accessed on 02/16/2023).
- [53] *Industry Foundation Classes (IFC) - buildingSMART Technical*. <https://technical.buildingsmart.org/standards/ifc>. (Accessed on 02/16/2023).
- [54] buildingSMART International. *IFC 4.3.1.0 OpeningElement*. <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcOpeningElement.htm>. (Accessed on 11/30/2023).

- [55] buildingSMART International. *IFC 4.3.1.0 Pset\_WallCommon*. [https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/Pset\\_WallCommon.htm](https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/Pset_WallCommon.htm). (Accessed on 11/30/2023).
- [56] buildingSMART International. *IFC 4.3.1.0 Spezification*. <https://ifc43-docs.standards.buildingsmart.org/>. (Accessed on 05/05/2023).
- [57] 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).
- [58] 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).
- [59] ISO - ISO 2848:1984-04 - *Building construction; Modular coordination; Principles and rules*. <https://www.iso.org/standard/7846.html.norm>. (Accessed on 12/01/2023).
- [60] 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).
- [61] Mauerwerksverband - Baulexikon. <https://baulexikon.beuth.de/MAUERWERKSVERBAND.HTM>. (Accessed on 06/29/2023).
- [62] Mohd Nasrun u. a. „Impact of Fragmentation Issue in Construction Industry: An Overview; Impact of Fragmentation Issue in Construction Industry: An Overview“. In: (). DOI: 10.1051/C. URL: <http://dx.doi.org/10.1051/matecconf/20141501009>.
- [63] Revit-Software | BIM-Software | Autodesk. <https://www.autodesk.de/products/revit/>. (Accessed on 02/16/2023).
- [64] Steinformat – Wikipedia. <https://de.m.wikipedia.org/wiki/Steinformat>. (Accessed on 07/25/2023).
- [65] The open source BIM collective. <https://github.com/opensourceBIM>. (Accessed on 02/16/2023).
- [66] 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).
- [67] DIN 4172:2015-09 Maßordnung im Hochbau. Norm. 2015.
- [68] Eurocode 6: Bemessung und Konstruktion von Mauerwerksbauten - Teil 1-1: Allgemeine Regeln für bewehrtes und unbewehrtes Mauerwerk. Norm. 2012.