



Encoding Building Products

Creating a design workflow for a modular infill system using
Discrete Building Information Modelling and Design Grammars

Tim Schumann
Student ID: 5374456

TU Delft University
Faculty of Architecture and the Built Environment
Master of Architecture, Urbanism and Building Sciences
Track Building Technology

Final Report - Defended 17th of January 2023

Mentoring Team:
Dr. P. Nourian, Design Informatics
Prof. Dr.-Ing. T. Klein, Building Product Innovation
Ir. H. Hoogenboom, Design Informatics

Delegate of the Board of Examiners:
Dr. Ir. S. van der Spek

External Advisor:
Ir. S. Azadi

 **TU Delft**
BK Bouwkunde

Abstract

One million homes will be needed in the Netherlands by 2030. Prefabrication in the construction industry can contribute to construct affordable houses, but that results often in standardized and not customized designs. These repetitive designs can be overcome through the introduction of mass-customization in architecture. Computational Architectural Design offers the possibility to tackle mass-customization through developing tools to let the user easily customize a design while providing guidelines and feedback towards a successful building design. This can be done by discretising a building into building blocks that can be controlled through a computational workflow.

This thesis elaborates on the potential of developing a design tool that allows the customization of houses through discrete building information modelling. Since the housing design process is a complex and multi-layered problem, the process is broken down into the sub-problems of topological design, building product development, configuration, and data export. For these steps, algorithms from the gaming industry are tested to improve the participation of the end user through a simplification of the design process. Through design grammars, a relational data structure is created that is compatible to the BIM environment of the industry.

The evolved methods are applied to the test case of a rowhouse design in Delft to predict the possible impact of the design tool. The results of this are set in the context of the AEC industry.

Keywords: *Mass Customization, Building Information Modeling, Discrete Architecture¹, Participatory Design*

List of Abbreviations

BMC – Boolean Marching Cubes Algorithm
BMS – Boolean Marching Squares Algorithm
PGC – Procedural Content Generation
WFC – Wave Function Collapse Algorithm
CAD – Computer-Aided Design
BIM – Building Information Modelling
AEC – Architecture, Engineering and Construction
ERD – Entity Relationship Diagram

¹ The term *discrete architecture* is explained in chapter 1.1.0

Preface

Motivation

My interest in mass customization started during my undergraduate thesis, where I developed an individualizable modular lightweight construction system. But when I began to work in an architectural office, this vision of simplicity and optimization was given way to the reality I experienced in the industry: highly complex workflows with a lot of repetitive processes. The introduction of BIM resulted in most projects creating heavy files containing a lot of geometrical information, which not even one team member had a complete overview of. My motivation for choosing the Master Track of Building Technology at TU Delft was to learn the tools and techniques to simplify and optimize these architectural workflows. The course Earthy, taught by Dr.Ir. Pirouz Nourian and Ir. Shervin Azadi introduced me to the concepts of generative design. As a result, I started to think of architecture not anymore in geometries but in spaces and related information. The ideas provided in Earthy, together with the research of the Genesis Lab in Participatory Design and Polygonization provided the foundation for this thesis.

Acknowledgements

This thesis would not be successful without other people's support, guidance and motivation. Firstly, I want to thank my mentor team Pirouz Nourian, Shervin Azadi, Tillmann Klein and Hans Hoogenboom for their guidance, patience and valuable input.

I would like to thank Pirouz Nourian for helping me through the methodological and theoretical part of this thesis and for the many hours of consultations in which he guided me in the right direction and provided valuable references as my main mentor till his departure from TU Delft.

A big thanks to Shervin Azadi, for showing me the ocean of possibilities in applying python programming to architecture. I like to thank him for his great explanation of highly complex content, the impressive pre-work he did for this thesis, and also his patience with my technical abilities.

Thank you to Tillmann Klein, who helped me to understand the concerns of the construction industry. He helped me to enrich a theoretical workflow with application possibilities and context.

I would like to thank Hans Hoogenboom for stepping in as my main mentor and bringing a new perspective to the project. He helped me to shape a plethora of ideas into a successful thesis.

Thanks to Nadia Remmerswaal and Tim Castelijm for helping me to evaluate the system's potential in the building industry context.

Thank you to the TU Delft and especially the organisers and teachers of the Building Technology Master for providing great coursework, which is an excellent foundation for my professional future. Furthermore, I would like to thank for the opportunity to spend a semester abroad at the University of Melbourne during my master, which enlightened me both on an academic and personal level.

Many thanks also to my parents, Andrea and Christoph, and my grandparents, Christine and Hans, who gave me the opportunity to do this master's degree at TU Delft.

Content

0 INTRODUCTION	6
0.0 Context	6
0.1 Problem Statement	7
0.2 Research Question	7
0.3 Research Objective	8
0.4 Methodology	8
0.5 Scope and Delimitations	9
1 LITERATURE REVIEW	11
1.0 Building Information Modelling	11
1.1 Modular Building Systems	13
1.2 Encoding Design	18
1.3 Conclusions	23
2 DESIGN METHODOLOGY	24
2.0 Design Terminology	24
2.1 Units and Sequences	25
2.2 Dimensioning System	28
2.3 Stakeholders	29
2.4 Computational Framework	29
3 DEVELOPMENT	32
3.0 Deconstruct the Marching Squares Algorithm	32
3.1 Building the Boolean Marching Cubes Algorithm	34
3.2 Testing the Wave Function Collapse Algorithm	37
3.3 Building Component Development	39
3.4 Conclusions	42
4 DESIGN	43
4.0 Design Tool Workflow	43
4.1 Creating a topological design	44
4.2 Tile Creation	49
4.3 Calculating Tile Options	56
4.4. Tile Placement	58
4.5 Postprocessing	61
4.6 Design Strategies	63
4.7 Workflow Reflection	66

5 EVALUATION	67
5.0 Verification	67
5.1 Validation	78
6 CONCLUSIONS	82
6.0 Research Questions	82
6.1 Method-specific Conclusions	83
6.2 Discussion	84
7 REFLECTION	85
8 APPENDIX	89
9 BIBLIOGRAPHY	91
9.0 Literature	91
9.1 Figures	94

0 Introduction

0.0 Context

The housing market in the Netherlands is pressured. From 2014 until 2020, nationwide house prices rose by 52%. In comparison to that, residential construction activity is weak. With an annual growth of 0,9% of the building stock in 2020, the level is below the construction volume of 2009 (Global Property Guide, 2022). Therefore, there is an urgent demand for creating affordable housing solutions as soon as possible. Affordable housing solutions can be provided through standardization and prefabrication of buildings, which often results in repetitive and unpersonalized designs. A concept to solve this issue is mass customization. Mass customization is a strategy that describes the creation of user-adapted solutions at the same or lower costs than a standardized production (Bock & Linner, 2010). While the concept is applied in other industries, such as the automotive industry with individualizable car options, the construction industry still lacks a broad application of mass customization. Through successfully improving this application rate, it is essential to establish an intense and continuous relationship between the end-user and the housing producers to generate value for all involved parties and improve the environmental impact (Bock & Linner, 2010). The basis of this relationship needs to be the end-user's participation in the design process at as many stages as possible. To enhance the relationship, the design workflow needs to be simplified in a gamified manner so that an unskilled end-user is able to participate in the process without contradicting the efficiency of a systemized architecture. Furthermore, it needs to be ensured to provide a huge range of design options for the end-user, what needs necessarily to result in a collaborative system, involving multiple manufacturers, designers and producers.

0.0.0 Societal Relevance

Buying an own house is for most people one of the most substantial decisions of their entire life. A tense housing market and high prices often lead to the choice of standardized housing solutions with no possibilities to adapt or individualize.

But in the society of the 21st century, family structures are becoming more complex, and the need for individualization of living spaces is vast. Changes in the size of a single family over the years requires the adaptability of space. An individualized building construction based on discrete components can solve this issue. Discrete components can be prefabricated; therefore, the final building is much more affordable than on-site construction. Furthermore, the discretization of buildings allows the development of design tools that can simplify the design process and integrate the end-user from an early stage. In prospect, with the combination of discrete elements with gamification, tenants can configure their living spaces with reduced professional support.

0.0.1 Scientific Relevance

Using a discrete approach for creating architecture might hypothesize that the creative possibilities are limited. However, the scope of combinatorial creativity allows countless design options. An example is the piano, containing a limited set of keys, with limitless possibilities for creating new pieces of music. A building system based on voxel-based components allows new possibilities towards information management within the building process. Establishing a discrete BIM-system allows component tracking from the early stages of the design process. Analysis data and design decisions can influence specific building components. Discretization is the basis for establishing algorithms in the BIM environment that can simplify the design process. In the scope of this thesis, the potential of a discrete BIM approach and its potential for the building industry is elaborated.

0.1 Problem Statement

Digital Design Platforms like The New Makers Builder offer a collaborative system in which a modular house can be designed from discrete components. The components are demountable and therefore allow adaptations and individualized solutions (The New Makers, 2022).

However, these systems work only with one specific building product universe developed by one company and therefore exclude millions of building products developed for the construction market. The reason for this is the lack of standards and the high complexity of the components, which hinders the collaboration of multiple manufacturers.

Building Information Modelling software solutions like Revit can integrate various products into a design workflow. However, these kinds of software are developed to handle planning projects of almost any size and complexity. Therefore, building elements are processed to create continuous geometries, resulting in highly parametric geometries and adaptable data. This structure allows a lot of flexibility, but it leads to complex relationships between components that require a lot of technical skills and hours of work to implement, structure and track components.

This research will tackle these two challenges and combines the two approaches to create an open-source collaborative building system that solves the dependencies and adjacencies for components developed in various environments. Furthermore, the research will elaborate on such a system's potential data integration, automatization and output generation. The roles of the architect, the end-user and the building product manufacturer is explored towards their participation potential. The methodology will be demonstrated on an infill system.

0.2 Research Question

Referring to the problem statement, the main research question of this elaboration is the following:

How to create a computational design tool that encodes building products into a participatory and discrete architectural workflow?

The research question implies that the main focus of this thesis is the creation of an integrated tool considering the aspects of translating building products of various manufacturers into discrete building components. Participation in this context reflects the influence of the main stakeholders on the design.

Sub-Questions

The following sub-questions are leading the path towards the main question:

How to implement several kinds of building components with various dimensions into a grid-based system?

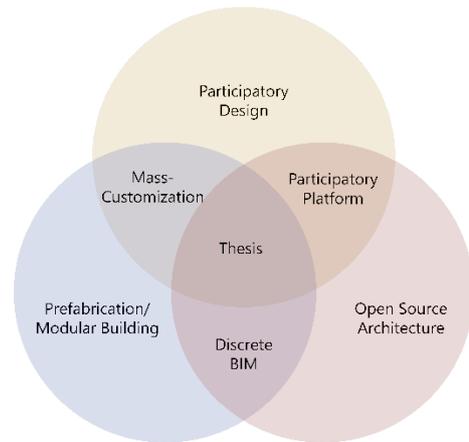
The current building industry produces building products in various shapes and dimensions. Therefore, an essential component of a holistic building system is to examine to which degree existing building products with different dimensions can be integrated in the system.

What are the participation possibilities in such a system for the different stakeholders in a building project?

To evaluate the potential of the workflow for the application in the context of the AEC industry, it is important to examine the influence of the workflow on different stakeholders in the building process. This elaboration examines the potential for the end-user, the architect and the building product designer.

0.3 Research Objective

This research is embedded in the research areas of open-source architecture, prefabrication and participatory design. This thesis's first aim is to develop a framework for integrating a discrete BIM workflow. This framework should be implemented as a design tool for a voxel-based building component infill system. The concept of Mass-customization is explored by introducing configuration options for the discrete components. Finally, the application of the design tool as a participatory platform is reflected. This thesis is embedded in the context of the Dutch housing market and is therefore focussing on products, services and systems that are available in the Netherlands.



0.3.0 Expected Product

The thesis product should consist of a computational workflow and an infill system applying the workflow. The computational workflow incorporates a design tool in which a professional user can design BIM-compatible tile sets based on building products. Additionally, it is aimed for a process in which these tile sets can be placed according to a coloured set of voxels and a tile preference by an unskilled user. Finally, the infill system should be incorporated into the workflow.

0.4 Methodology

0.4.0 Research Methodology

The elaboration follows a rationalistic-atomistic approach. This means that the research problem is investigated based on its smallest sub-units of reason and logic. Therefore, the methodological approach chosen for this framework is design science research. This research methodology has a scope on creating an innovative product with focussing on the process of making choices rather than existing knowledge. The design science research methodology distinguishes five steps. After the identification of the problem, objectives for the solution are defined. Based on that, design and development is performed. The results are first implemented to demonstrate the solution and finally evaluated through verification and validation.

This research is embedded in the generative design framework of GoDesign. GoDesign provides a complete framework for the mass-customization and optimization in architectural design. This thesis focuses mainly on the shaping phase and is therefore compatible with methods and processes developed in the scope of the other phases of GoDesign (Azadi & Nourian, 2021).

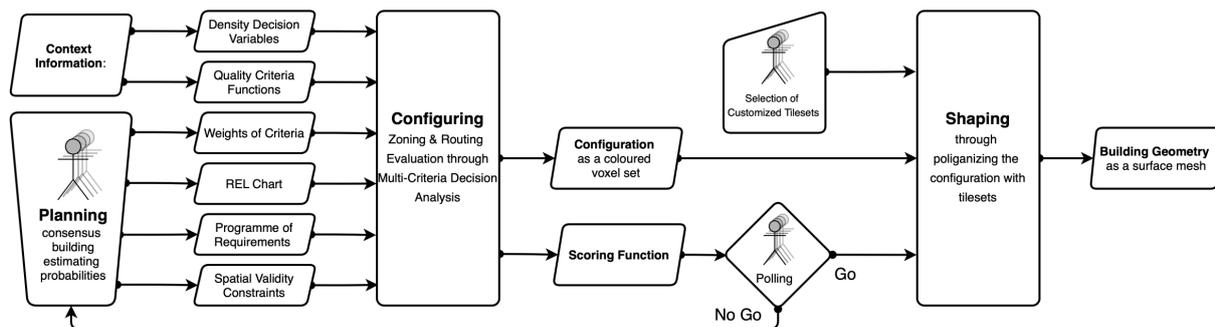


Figure 1: GoDesign Framework (Azadi & Nourian, 2021, p. 288)

0.4.1 Proposed Methodology

Following the design science research methodology, the following methods were deduced.

1. Literature Review

For creating a valid design tool, it is necessary to incorporate the current state of research. Therefore, a literature review in the three research areas of the thesis, Building Information Modelling, modular building systems, and strategies for encoding designs, is conducted. In the area of BIM, current workflows and data storage and processing methods are examined. After that, current industry strategies of prefabrication are set in the context of open building and discrete architecture. Finally, the technical foundations for implementing a discrete computational workflow are elaborated.

2. Design Methodology

It is essential to define a ruleset and evaluation criteria, to guide the developments according to a theoretical framework. Therefore, a ruleset is established and set into a framework after specifying the design terminology.

3. Development

In several studies, the BMC and WFC algorithm's potential is extracted, and their application's possibility is evaluated. Furthermore, several aspects of the infill system are designed and tested.

4. Design

In the design phase, the computational workflow is created and tested. Finally, the tile set database, the algorithmic placement, and the participatory framework are combined.

5. Evaluation

The system workflow is evaluated through validation and verification. For validation, the workflow is implemented on a rowhouse typology, and the functionality is elaborated. For validation, the value of the workflow for the industry is elaborated by implementing systems and products from the industry. Furthermore, interviews with experts from the housing prefab industry are conducted to evaluate the thesis product.

6. Conclusion

The results of the evaluation are assessed, and further potentials are outlined.

0.5 Scope and Delimitations

The scope of this elaboration is the intersection between architecture, computational design, and product design. The main aim of this research is the creation of a design workflow for discrete buildings. For demonstration, an infill system is developed and implemented in the workflow. A detailed and producible set of components is not part of this study.

The developed system could be applied in larger housing projects with various stakeholders, but an individual house owner is addressed for this study. The development of a structural framework, as same as the participation of more complex ownership situations, is not considered. The system's performance is set in the context of the construction industry.

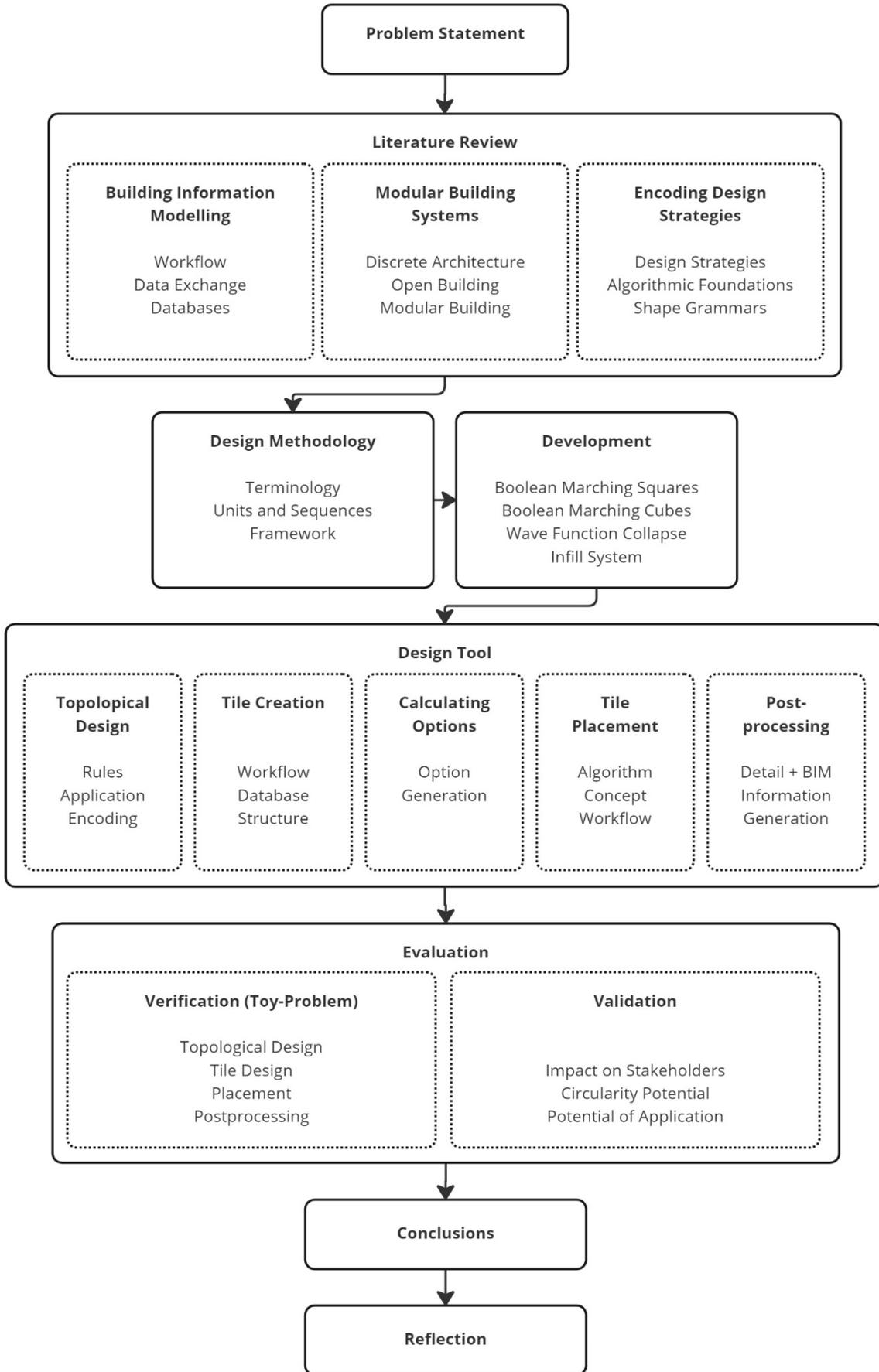


Figure 2: Thematic organisation of the research (Author)

1 Literature Review

1.0 Building Information Modelling

In the scope of this research, it is relevant to get an overview of the common principles, data structures and advances in Building Information Modelling. The second step examines the possibilities of file- and data exchange in BIM.

1.0.0 Workflow with BIM

Before Building Information Modelling was introduced, the construction industry did not work as a whole. The individual disciplines looked for their best interest, and the quality of the whole project suffered. In 2007 in the United States, the waste created by this inefficiency was estimated at \$500 billion (Rex Miller, 2009). With BIM, however, it is possible to overcome the communication deficits in the industry by building a structure beforehand virtually. BIM allows the extension of model information by additional data, such as scheduling, safety and costs (Hardin et. al., 2015, p. 2). BIM is relevant for construction, as same as for the design. The link between these disciplines is done by linking geometry information to real-time databases. The crucial areas of information stored in the databases are construction documentation, visualization, material information, cost estimations, 4d construction sequencing and scheduling, and fabrication data (Garber, 2014).

1.0.1 Geometries, Components and Libraries

While the shift from a physical drawing to CAD just changed the medium, the shift from CAD to BIM changed the focus from a class-oriented to an object-orientated design environment. While in CAD-Software, geometries are grouped in classes, for example, curves and solids, the information modelling groups objects into libraries. These libraries can contain additional information, such as materiality or adjacency. This object-oriented design approach allows simple geometrical data, such as lines, to represent more complex geometries by adding information modelling packages and three-dimensional modelling packages. Today, many building suppliers offer libraries of specific products, such as doors, windows, and screws. The selection of components from a library provides a streamlined design workflow, but it reduces the architectural design to the selection of pre-configured choices (Garber, 2014).

1.0.2 Data Exchange

Fluent communication between a construction project's stakeholders is essential in the BIM environment. However, at the beginning of the development of BIM applications, there was no standard file format, and different programs' interactions were impossible without data loss.

With enhancing the use of BIM, several data exchange methods were developed. Lou et al. (2020) described the three exchange categories: file-based exchange, cloud-based exchange or local data exchange. A file-based exchange is a one-way data transfer from one platform or collaborator to another through a standardized file format, for example, the Industry Foundation Classes. A disadvantage is that manipulations can only be performed on the file, not object level. This issue can be solved with a cloud-based exchange. However, this exchange method lacks open standards and privacy strategies. A third option is the local data exchange, which can be performed through blockchain technology. This technology is still in development and lacks compatibility and conflict-resolving mechanisms (Lou et al., 2020).

Speckle is an example of a cloud-based data transfer. It is an open-source data platform to connect 3D and BIM data across the AEC Industry. Users can install speckle connectors in their working programs, such as AutoCad, Rhinoceros, Excel or even Python. Speckle allows real-time data transfer and versioning of files (AEC Systems, 2022). Speckle offers access and visualization of 3D data online.

The "Industry Foundation Classes" or in short "IFC" is a standardized file format to allow data transfer between the architecture, engineering, and construction industry. The file format contains information about the identity of objects, characteristics, relationships, and processes. Additionally, information about stakeholders, like owners and designers, as well as abstract concepts like performance, can be codified (buildingSMART-International). The development of the format started back in 1995. Therefore, the format is based on EXPRESS, a data formatting language available at the time of development. Twenty-five years later, IFC became the standard for data exchange in BIM, but most developers have no or little experience anymore with the work in EXPRESS. Therefore, a task force was developed to modernize the ifc format. Independency of the data structure from a specific programming language should be achieved. Furthermore, the task force plans to modularize the format, separate responsibilities, and eventually allow independent release cycles (Berlo et al., 2021). These aspired improvements would enhance the advantage of the IFC standard over other exchange methods.

1.0.3 Classifications

The IFC format introduced a universal classification system for data exchange in BIM. However, in the construction industry, the ISO 12006 framework is established. ISO 12006, titled "Organization of information about construction works" (International Organisation for Standardization, 2015) is applied worldwide as a basis for national classification systems for construction. While the objectives of these two classifications are comparable, they have considerable differences in internal semantics and structure (Ekholm, 2005). For the actual compliance of IFC to the national classification systems, a conversion is required. The developer of the IFC standard, buildingSMART International, developed a Data Dictionary that can convert the ifc classification to national classifications (buildingSMART-International, n.d.).

1.0.4 Databases

In the scope of this research, it is relevant to access databases that provide data about building products and detailing that can be implemented in the workflow. While countless websites offer building product information, three examples are presented to show the potential of different kinds of databases relevant to this thesis.

Bimobject

Bimobject.com is an online database to create and retrieve building products in various file formats. It is the leading database in that sector, with 31 million files downloaded annually. The platform offers technical specifications and the BIM-related classification of the offered products. Products can be downloaded and seamlessly integrated with a BIM workflow in common software (bimobject, 2022).

NBS Source

The National Building Specification (NBS) is a company in the United Kingdom that provides a classification system but also offers a tool to implement building products. NBS source is a tool to find building products and download specifications and BIM-files, such as in Revit or IFC format. An advanced workflow is created through the integration of classification and specification (NBS Source, 2022).

KVT (Kwaliteit van houten gevelementen) Online is a platform created by the NBvT (Nederlandse Branchenvereniging voor de Timmerindustrie). It provides a guideline for creating wooden facade elements, providing information about the workflow, and integrating materials, assembly and detailing. Compared with the other two platforms, the KVT does not provide specific building products from manufacturers. However, the end products described on the platform fulfill the Dutch building certification BRL 0801 (NBvT, 2014).

1.1 Modular Building Systems

1.1.0 Discrete Architecture

In Mathematics, a discrete variable is countable in a finite amount of time. A continuous variable, however, would take forever to count. Therefore, a continuous variable is measured, while a discrete variable corresponds to a value in the natural numbers (Abramowitz, 2012, pp. 927). Also, in technology, this phenomenon is present. An analogue signal is a continuous signal, while digital information is encoded in discrete bytes.

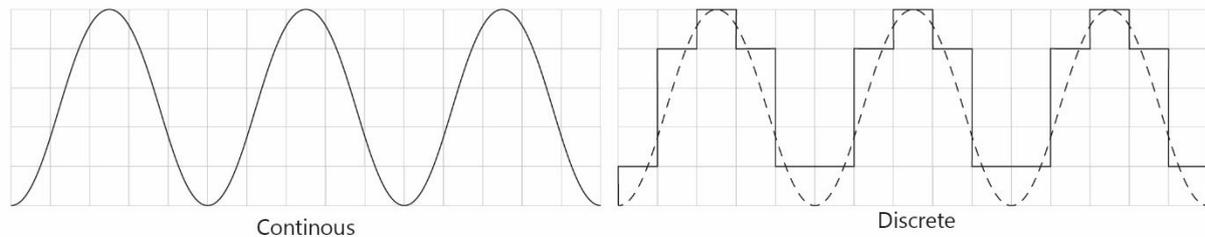


Figure 3: The Continuous and the Discrete (Author)

The concept of discrete architecture is an alternative perspective towards modular buildings. Gilles Retsin describes discrete architecture as the following: “the Discrete is an emerging body of work that seeks to redefine the entire production chain of architecture by accelerating the notion of discreteness in computation and the physical assembly of buildings” (Retsin, 2019, p. 7). The concept is based on a part-to-whole approach. This means creating buildings out of parts instead of modelling them continuously, such as a concrete shell. Since computation is essentially based on discrete data, combining these parts with a digital assembly workflow is possible. Constructing a building from parts means a loss of a certain resolution, but it is traded for the possibility of scaling (Retsin, 2019).



Figure 4: Assembled Discrete Building Blocks, Photo by James Harris (McAndrew, 2022)

1.1.1 Open Building Concept

The Open Building Concept was developed by the Dutch architect John Habraken and aims to improve the interaction between the building industry and customers in the built environment. The main concept is the subdivision of a building into several levels of decision-making. Open Building differentiates between the urban fabric "tissue", the containing base "support" and the infill. The distinction between infill and support systems on the building level creates separate entities with different life cycles and can be supplied independently. The support system is setting the limits for the infill system, while the infill system determines the requirements of the support system (Cuperus, 2001).

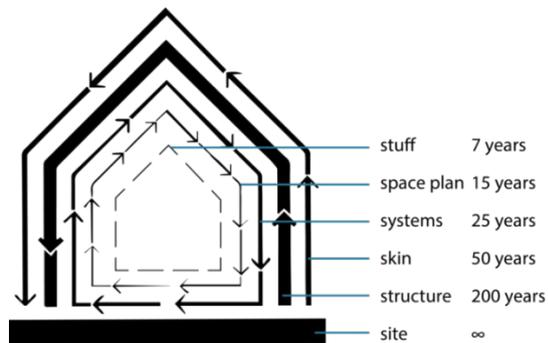


Figure 5: Shearing Layers of Change (Stewart Brand, 1994)

Coordination

The main challenge in Open Building is the coordination of dimensions and connection between the individual components. Therefore, a classification called *formal description* was introduced to give every element in a building a unique identifier. The formal description contains two parts. The abstract description gives specific information about the position of the element in the building. The product description states the adaptability of an element. It distinguishes between prefabricated and therefore unadaptable elements, standardized elements like plates and elements that are produced on-site (Cuperus, 2001). In the scope of this thesis, a new approach towards digitalizing these kinds of classification in the framework of Open Building is elaborated.

Matura Infill System

A mentionable system incorporating the principles of Open Building is the Matura Infill System. The Matura Infill system was developed by the Ahrend group in the Netherlands in 1986, focusing on modular coordination and incorporating cables and conduits (Kendall, 2015).

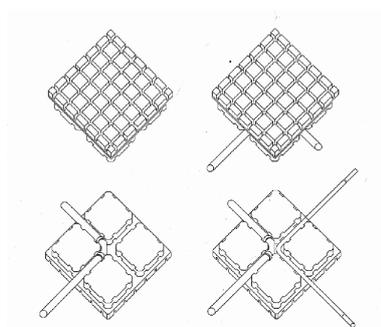


Figure 6: Matura Infill System (Gan, S., Zhang, H., 2021)

The system introduces the matrix tile, a component that rests on the floor and contains piping. This matrix tile can be covered thanks to its LEGO-like connection points with flooring elements and

baseboard channels that can contain further cables. The system can contain all necessary piping for a room and allows the assembly of “off-the-shelf-products” (Cuperus, 2001).

Furthermore, Matura developed a new logistics strategy for dwelling projects. Instead of that, all producers of infill parts for dwellings deliver their products to the building site independently, Matura offered a product bundle service. All products required for one dwelling unit were packed together and loaded into one container in the reverse order in which there are used on-site. This allowed a complete installation of a fit-out system of a 1000 m² unit in only two weeks (Kendall, 2003).

1.1.2 Modular Architecture

To categorize modular architecture, three models can be distinguished as described by (Klein, 2013): Slot modularity, bus modularity and sectional modularity.

Slot modularity describes the use of one connection interface, that allows the placement of several different discrete elements. An example of that is the car radio. A few years ago, a car owner could fit the radio of almost every other car in her connection slot.

Bus modularity describes the concept of an independent connection interface that can connect two or more objects. An example of that is the USB Port, connecting several electronic devices.

The third option is sectional modularity. In sectional modularity, a part shares the same connection interface with other parts with different properties. An example of this modularity is the Lego system, with interlocking notches and grooves (Klein, 2013).

1.1.3 Design for Deconstruction

While having the primary goal of reconfigurability, it is essential to address the demountability of components. Guy and Ciarimboli (2005) developed guidelines for the design for deconstruction that are worth mentioning. Firstly, materials with reuse and recycling potential should be preferred to use. The materials need to be labelled for efficient deconstruction. Connections need to be accessible. Non-permanent connection methods, such as screws and bolts, should be preferred. In general, components should have a straightforward structure and interchangeability should be achieved through standardization and modularity (Guy and Ciarimboli, 2005).

1.1.6 Case Studies for modular Building Systems

In this section, several Building Systems for the dwelling market are presented. They are all adopting the Open Building concept, and they are applicable to rowhouses or similar typologies. The features of the systems are categorized and compared.

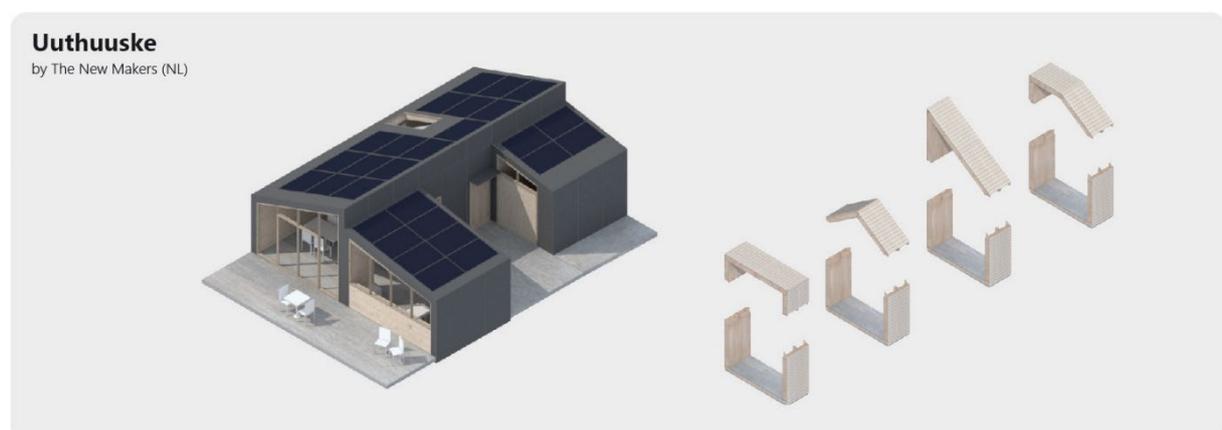


Figure 7: The Uuthuske System (The New Makers, 2022)

The Uuthuuske is a biobased modular system developed by the New Makers in Delft. With the help of digital fabrication, an IKEA-like kit is produced in the factory. The kit is easily mountable, and subcomponents, as interior walls, can be clicked together by the house user. (Oorschot & Asselbergs, 2021) The system is based on sectional modules of 1.2m, which are mounted together. All components and service systems can be demounted, so there are a lot of configurations to choose from for the user. The comfort cabin is a prefabricated block that is inserted in the house, containing a kitchen, bathroom and toilet. Currently, the user can customize a building based on various material options and element typologies with assistance through an architect of the New Makers (*The New Makers*, 2022). The Uuthuuske is a very advanced modular housing system, offering a lot of configuration possibilities for users. The demountable wooden elements are the base for a comprehensive circularity concept. However, the fixed width of the building limits the spatial choices for the user a lot. Furthermore, through the closed product universe of this system, the variety of modules is limited.

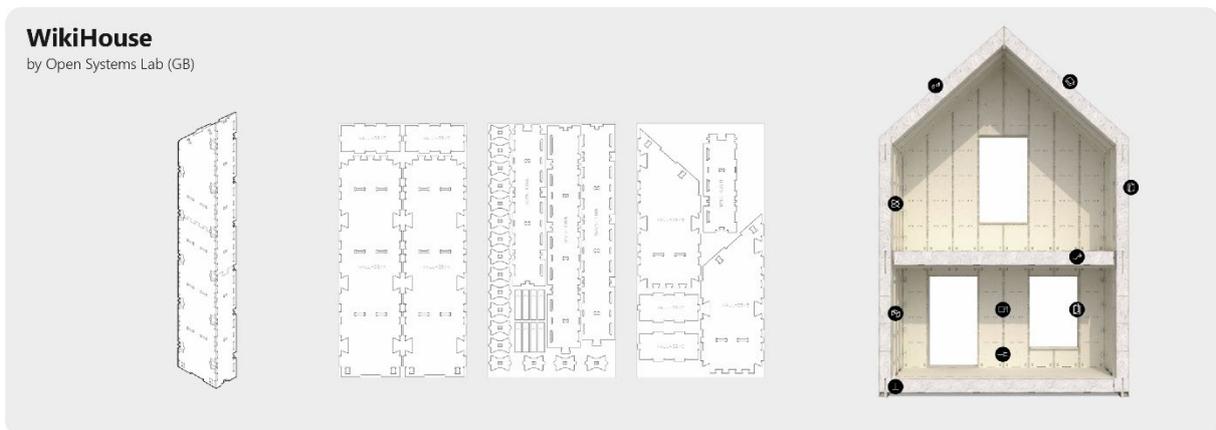


Figure 8: Skylark System (Open Systems Lab, 2022)

Another system that is referred to translate an IKEA method to building scale is the WikiHouse project. WikiHouse is an open-source and digitally-manufactured building system for small and cost-effective housing solutions. Users can download a housing design, modify it in SketchUp, and produce it then themselves with a CNC machine and plywood sheets. The connections are solved with wooden joinery, screwing is mostly avoided. This way, the houses are completely demountable ("www.wikihouse.cc," 2022).

The WikiHouse is highly participatory since it transfers the complete production and assembly process to the end-user. Thanks to the wooden joinery and the use of wood, it is very circular. On the other side, WikiHouse requires high skills. Users must be able to fluently work with SketchUp and a CNC machine, furthermore they need basic construction skills.

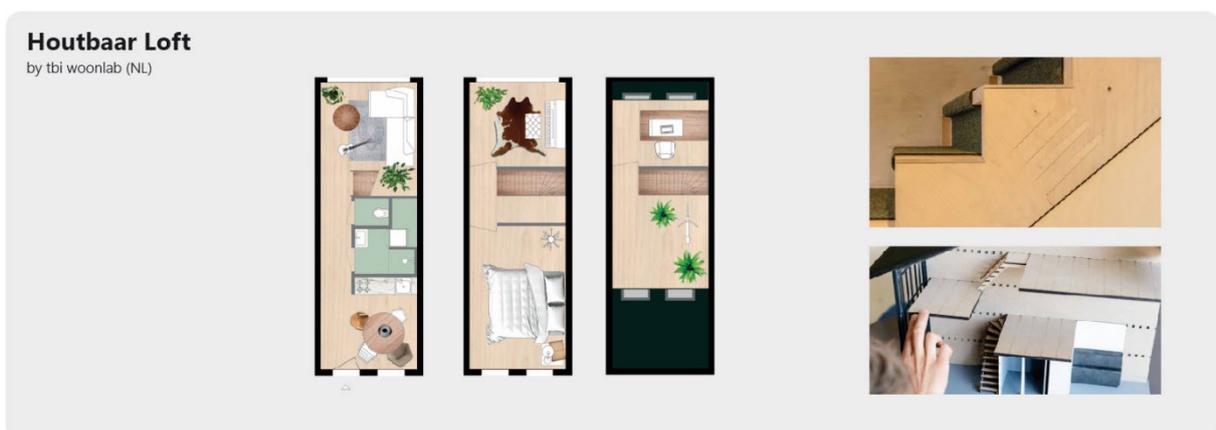


Figure 9: Houtbaar Loft. TBI Woonlab (2022, 01)

Houtbaar Loft is a modular building system for rowhouses, developed by TBI Woontab in collaboration with The New Makers and Sustainer Homes. Houtbaar Loft has a standard hull that contains two wooden structural side walls and two facades with windows for the front and the back. As claimed on the product website, the system is fully demountable, completely produced out of CLT panels, and can be installed on location within one day.

When the hull is installed, end-users can choose their preferred infill from a module collection. The assembly is managed through clicking-connectors, which are placed on the structural sidewalls. Toilet, bathroom and kitchen are integrated into a fixed service core in the middle of the house. At 13.01.2022, the Houtbaar Loft is still in the prototyping phase and the pricing structure is not yet known (TBI Woonlab, 2022, 01).

The strength of Houtbaar Loft is the combination of a fixed structural hull with a flexible infill system. The reconfigurability of the infill elements offers inhabitants a wide range of choices for a loft building. However, a disadvantage is the strong design limitation through the fixed envelope, restricting also the possibilities for indoor spaces.

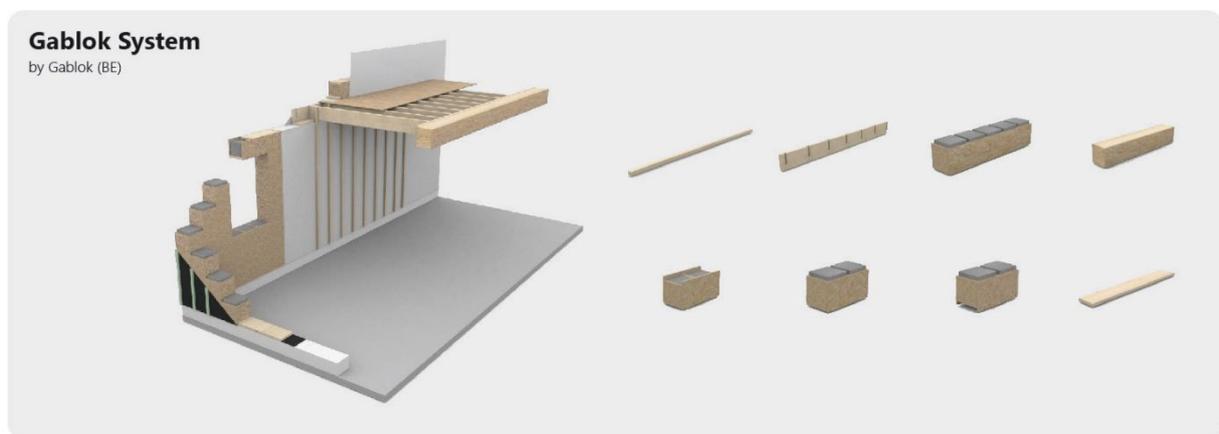


Figure 10: Gablok System (Gablok, 2022)

The idea of Gablok evolved from interlocking building block toys. Founded in Belgium in 2019, its goal is to allow self-builders to construct their own home. The concept is the stacking of interlocking isolated wooden blocks towards the construction of a whole building. With only 8 different elements the assembly of a shell construction is possible. The company converts the building designs by an architect into the Gablok system and delivers the blocks to the site. The assembly of the Blocks is then in conducted by the end-user (Gablok, 2022).

The Gablok system is translating the building principles of the modular toy LEGO to the building level. The modular wooden construction is demountable and is has therefore a high circularity potential. As a self-build-system, it requires however high personal effort by the end-user.

1.2 Encoding Design

In the scope of this thesis is the development of a design tool that simplifies the architectural design process and offers the opportunity for non-experts to participate in it. For that, several tools for generating designs are presented and evaluated. Following that, technical foundations for the implementation of the most promising tools are elaborated.

1.2.0 Strategies for encoding designs

Savov et. al (2020) defined five strategies for encoding architectural design into interactive modelling platforms: implicit modelling, parametric configuration, part assembly, procedural generation and assisted sculpting. The strategies are evaluated from the perspective of an end-user who wants to create an individualized design, and a professional designer who assists the end-user and provides initial input for the strategies.

Implicit Modeling

Implicit Modeling describes the creation of designs in the commonly available CAD softwares or 3D modelling software like McNeel Rhinoceros, Autodesk AutoCad or Blender. While these programs allow the highest degree of design freedom, this freedom implies a complex modelling process not suitable for an unskilled user.

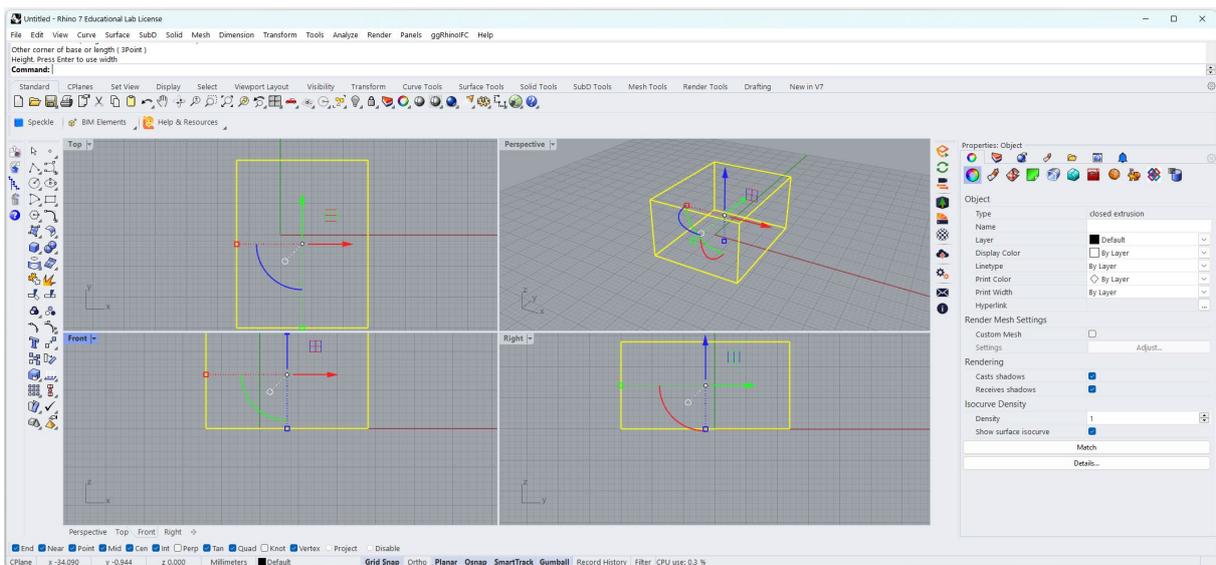
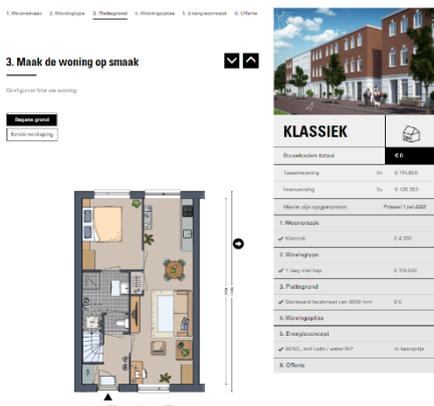


Figure 11: Implicit Modeling with Rhino (Author)



Parametric Configuration

Parametric Configurators are widely available online to customize specific, predefined designs. These configurators allow a non-expert to participate in the design process by selecting a specific combination of predefined options. However, while they are easy to use and can provide direct design feedback, such as the final price, the design freedom for the end-user is very limited.

Figure 12: Parametric Configurator of the beterBASISHuis (TBI Woonlab, 2022 10 11)

Part Configuration

The Part Configuration is the extended version of a parametric configurator. Instead of just selecting predefined options, the end-user can freely configure parts following predefined rules towards an individualized assembly. While the end-user has a high degree of freedom, the expert's influence on the final design is minimal.

An example of a part configurator is the IKEA Kitchen Planner. The planner is accessible via a desktop computer or mobile device in any web browser. Firstly, the user needs to enter data about the general type of kitchen she prefers. Then, the kitchen room layout is designed by simply entering the dimensions of selected walls. In the third step, an example kitchen is generated and positioned in the individualized kitchen room. All kitchen components can be moved, exchanged and modified to users' preference. The user is able to position elements freely in space. Feedback is given firstly by the price indication of the whole kitchen configuration and secondly by detecting falsely placed elements. As seen in the figure below, items that are not placed according to prespecified design rules are pointed out to the user.

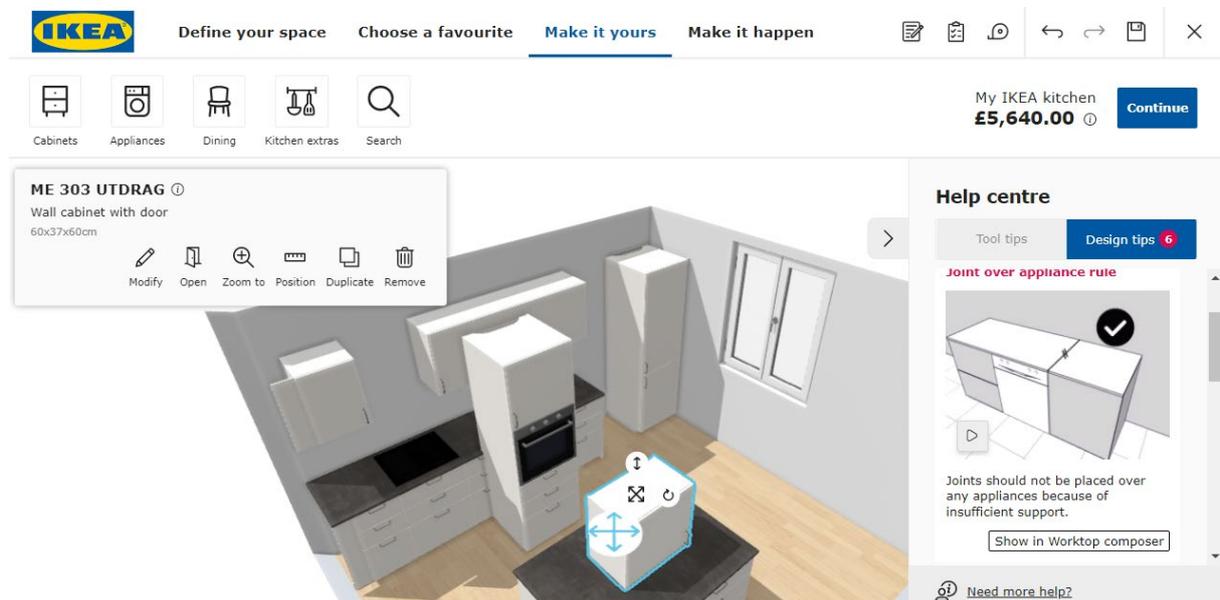


Figure 13: IKEA Kitchen Planner (IKEA, 2022)

The initiative Block by Block is a different approach as a part configurator. Block by Block is a joint program between Mojang, the developer of Minecraft and UN-Habitat. They have the novel approach of using the computer game Minecraft to let communities participate in urban planning projects. Established in 2012, the foundation has funded public space projects in more than 35 countries, often in developing countries, impacting the life of more than 2.3 million people. Minecraft is a widely known game and is easy to use, especially for kids. In workshops involving people of all ages, urban spaces are designed in Minecraft. The design process is done in Multiplayer mode, involving several players working together. The designs are then transferred to an architect, who converts them to design drawings. After that, the project can be funded and built (Block by Block, 2022).



Figure 14: Rebuilding Tradition In The Kathmandu Valley- Minecraft Model and Reality (Block by Block, 2022)

Procedural Generation

With Procedural Generation, designs are shaped through additive or discrete modelling according to predefined rules. The rules are embedded in algorithms that can run without different degrees of user-interaction. While keeping control over the design workflow for both professionals and end-users, the algorithms can be implemented in intuitive design games.

An example of Procedural Generation is the video game Townscaper, developed by Oskar Stålberg and based on the Wave Function Collapse Algorithm. The game is a tool to let the user express combinatorial creativity. The user has only two possible actions to perform in the game. A click on the left mouse button will add a block, and a click on the right mouse button will remove a block. This simplicity in the functionality lets the user immediately understand the game's principles and generate design options. Without specific user instruction, decisions to place blocks at specific locations in the model, balconies, staircases, roofs and other building elements are generated (Stålberg, 2022).

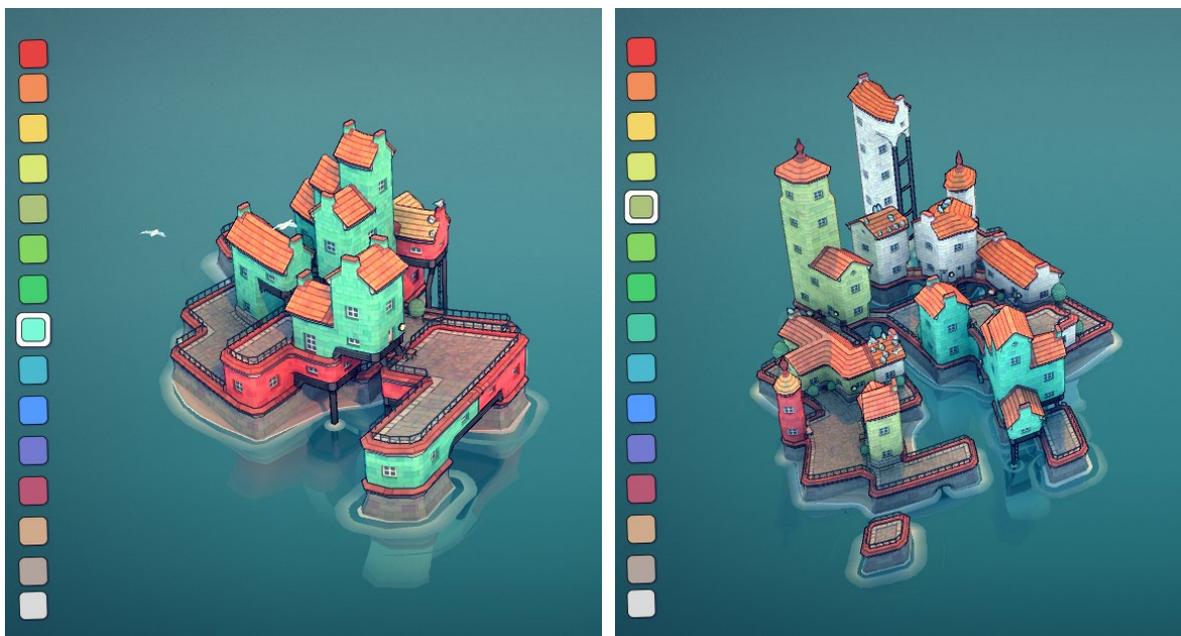


Figure 15: Two Designs created in seconds by the author in Townscaper (Stålberg, 2022)

Assisted Sculpting

The last encoding strategy is assisted sculpting, as developed by Savov et. al 2020. To create a unique configuration, the user can sculpt a set of voxels according to a predefined grid. The set of voxels is converted into geometries through an algorithm, that selects and places predefined tiles according to the configuration of the voxel set. The user keeps a high control of the output, while the usability and the influence of the expert remain acceptable.

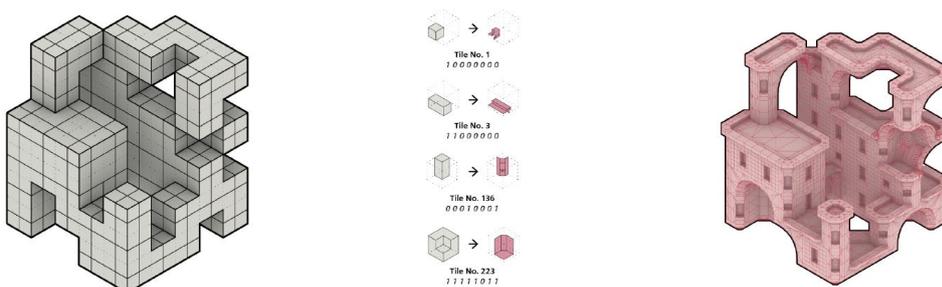


Figure 16: Assisted Sculpting (Savov et. al, 2020)

Evaluation

In the figure below, the strategies for encoding designs are evaluated. This thesis aims for the creation of a design tool that can be used by unskilled end-users but allows professionals to generate design guidelines at the same level. The influence of these two players is therefore essential, as same as the easiness of use. Furthermore, it is aimed to generate design feedback to provide the end-user feedback for the design decisions that are made. Applying these factors, assisted sculpting and procedural generation strategies are performing best. Therefore, in the following chapters are the technical foundations of these strategies evolved.

Case Study	Implicit Modeling	Parametric Configuration	Part Configuration	Procedural Generation	Assisted Sculpting
application example	Rhino AutoCAD Sketchup	IKEA Kitchen Planner	IKEA Kitchen Planner	Townscaper Chasioti (2020)	Savov (2020) Park (2022)
experts influence on design	✔ high	⚖ medium	✘ low	⚖ medium	⚖ medium
users influence on desing	✔ high	✘ low	✔ high	⚖ medium	✔ high
difficulty of use	✘ very high	✔ low	⚖ medium	✔ low	⚖ medium
feedback to design decisions	✘ no	✔ yes	✔ yes	✔ yes	✔ yes
rating (✔=3, ⚖= 2, ✘= 1)	6	7	7	8	8

Figure 17: Comparison of strategies for encoding designs (Author)

1.2.1 Constraint Solving

For assisted sculpting, as for procedural generation, design rules need to be integrated that set constraints to the design space of the user. To integrate these constraints, such as the need for a door, the placement algorithm must be able to solve constraints. The theory behind this is the constraint satisfaction problem. A constraint satisfaction problem is a mathematical problem that contains a finite set of variables. Each variable has a non-empty and finite domain of possible values. Furthermore, it contains a set of unary and binary constraints. Unary constraints involve only one variable, while binary constraints involve two variables.

An example of a constraint satisfaction problem is the map-colouring problem. In figure 4 is shown how a map colouring problem can be solved (Tsang, 1993).

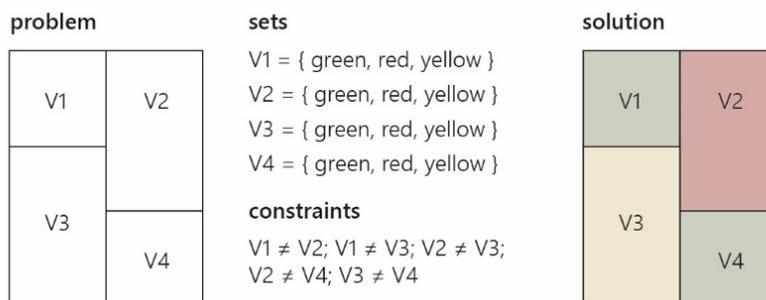


Figure 18: Map-coloring (author)

The tree search method is the simplest method to solve the constraint satisfaction problem. With this method, variables are instantiated one after the other. First, a value from the variable's domain is assigned, and the constraint check is performed. If the constraints are satisfied, the next variable is instantiated. Backtracking is executed if there is no opportunity to assign a value without violating the constraints. This means that the algorithm is revisiting the latest assigned variable and trying to assign a new value for it (Liu, 1998).

1.2.2 The Wave Function Collapse Algorithm

The Wave Function Collapse algorithm was developed by Max Gumin in 2016. It is based on the Model Synthesis Algorithm created by Paul Merrell in 2007 (Merrell, 2022). The name “Wave Function Collapse” refers to a phenomenon from quantum mechanics. It describes the abstract probability of finding a specific particle somewhere. When a particle is unobserved, it may be in several possible states. Only observing the particle's position sets the probabilities of all other options to zero (Giacosa, 2014). This principle of collapsing possible positions for elements is also the principle of the WFC Algorithm.

The algorithm is an example-driven procedural content generation algorithm using constraint solving and was initially created for 2D pixel image generations. However, it is also applicable on a 3D tileset to create architectural models. Marian Kleinberg developed an infinite virtual city from a simple tileset containing bridges, roofs, walls and streets (Kleineberg, 2022). Eleni Chasioti went even further and developed a tool to create models with the WFC-Algorithm directly in the 3D-modelling software Rhinoceros, with the help of a Grasshopper-Plugin. The plugin allows the creation of design options based on predefined tilesets for the early design process (Chasioti, 2020). Through the support of the implementation of design constraints, the algorithm can be combined with other algorithms or manual inputs (Gumin, 2022).

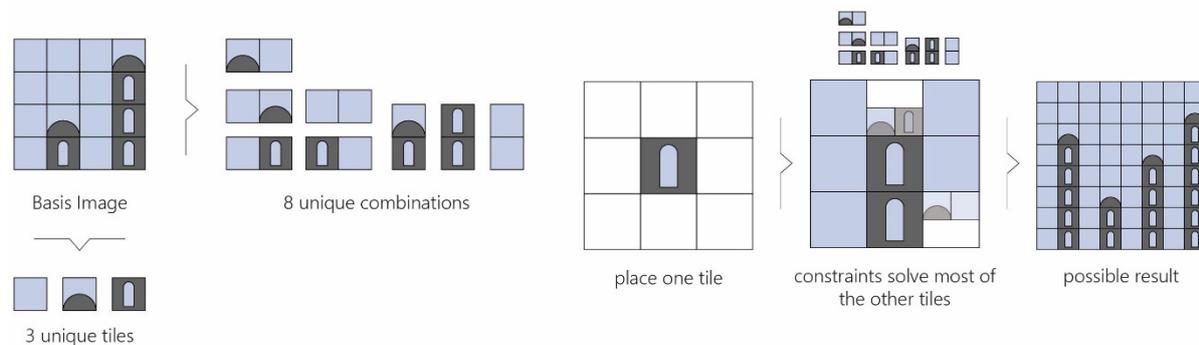


Figure 19: Workflow of the Wave Function Collapse Algorithm (Author)

1.2.3 The Marching Cubes Algorithm

The Marching Cubes Algorithm was developed by William Lorensen and Harvey Cline in 1987. It was initially developed to create triangular vertices using linear interpolation of medical data, such as MRT scans. The basis is a scalar field of voxels, and the algorithm checks the neighbouring conditions of the eight corners of the voxel. Based on the 15 unique cube configurations, a polygonal mesh is extracted (Lorensen & Cline, 1987).

The python library topoGenesis was developed by Pirouz Nourian and Sherin Azadi at TU Delft University since 2020 (Azadi & Nourian, 2021). It contains topological structures and functions that can be used in generative design. It contains the Boolean Marching Cubes algorithm, which is an optimization of the initial Marching Cubes algorithm for generative design in architecture (Nourian & Azadi, 2022). 24 cube configurations can incorporate all possible neighbouring conditions for one voxel. For each of them, a specific tile is defined, forming a tileset. When the algorithm is running through a voxelized shape, it places the tile with the matching neighbouring conditions on the position of the analysed voxel.

1.2.4 Shape Grammars

To produce algorithmic designs that are technical valid and aesthetically pleasant, multiple design rules need to be defined. A set of these rules tackling three-dimensional shapes can be called a shape grammar. Sass (2005) defines two subcategories of shape grammars relevant for housing design: design

grammars and construction grammars. Design grammars can produce architectural designs but do not consider materials, assembly, or construction methods. The main aim of design grammars is the production of a variety of design options. Construction grammars on the other hand are translating a design model into construction components. For example, a construction grammar can transform a specific three-dimensional design into a set of two-dimensional line drawings that can be used for CNC fabrication (Sass, 2005).

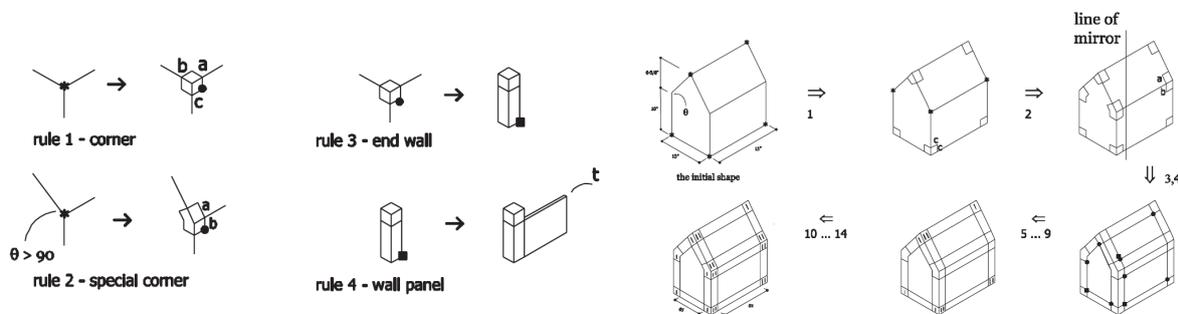


Figure 20: A shape grammar and grammar derivation (Sass, 2005)

1.3 Conclusions

The results from the literature review are the foundation for the development of evolving the methodology of a design tool for discrete building information modelling. The following conclusions can be drawn:

Advanced databases for obtaining building products and guidelines are existent and integrated into the workflows of the industry. A novel system should integrate connectors to the BIM environment, both on the input side through building products on the output side with data exchange. The IFC format was identified as the most suitable data format to integrate into the workflow.

A modular system is dependent on a dimensional system. In most presented cases, the dimensioning system is dependent on the width of the building, as in the Houtbaar Loft. Sometimes, it is the minimum element size, as at Gablok. The careful choice of a dimensioning system is an essential factor regarding the limitations of the system. Design for deconstruction should be considered through a simple structure with non-permanent connections as a basis for the infill system.

The WikiHouse is the only presented system that allows the user to modify existing modules and create new ones. In general, it is interesting that almost all building systems have a closed product universe supplied by one company. Opening the system while still keeping the demands towards the user's skills low is a crucial component towards the success of the thesis product.

The principle of converting architectural building elements into tilesets and creating a plethora of spaces with them seems to have significant potential for application in the scope of this thesis. While the Boolean Marching cubes algorithm can identify specific spatial situations, the Wave Function Collapse Algorithm allows an end-user to select particular elements and arrangement patterns. Combining these two algorithms towards a user-controlled placement system for system elements can bridge the gap between participation and standardization.

2 Design Methodology

2.0 Design Terminology

In this project's scope, there is a computational and a geometrical approach to structuring data. This duality allows building components to be placed according to simplified computational rules. The following presents the terminology to differentiate between the two data structures.

Voxel

A voxel is a hexahedron representing a specific geometrical module of a specific size in space. Multiple stacked voxels form a voxel array, which represents connected spaces with the size of multiple geometrical modules. In addition, a voxel can contain information about a specific colour, allowing a set of voxels to have multiple colours and display different spaces. A coloured set of voxels is the initial user input towards the spatial configuration of the building.

Cube

A cube is a geometrical module with the same dimension as a voxel. The vertices of a cube are the centerpoints of eight adjacent voxels, and the centerpoint of the cube is a vertex of each of those voxels. Therefore, the voxel and the cube have a relationship of duality. The cube's dimensioning shapes the overall grid for the building system.

Sub-Voxel

The sub-voxel represents the region of the overlapping of a cube with a voxel. Since a cube overlaps with eight voxels, the zone is an octant of the voxel, as same as an octant of the cube. In a cube configuration, the sub-voxel is represented by one digit. This digit can be binary (0 or 1), but it can also represent the colour of the parent voxel as an integer. The sub-voxel, as a geometrical unit, is used to generate the basic tileset, forming the basis of all other tilesets.

Computational Tile

The computational tile is a geometry assigned to a specific cube. In its most simple form, the tile is represented by a surface, separating occupied sub-voxels from unoccupied sub-voxels. The geometry is always related to the occupation (or not-occupation) of the eight sub-voxels forming one cube. The size of the computational tile is always relative to the voxel/cube sizing. Additional to the eight vertices, a tile can be identified through the six faces of the assigned cube and the specific Tile ID.

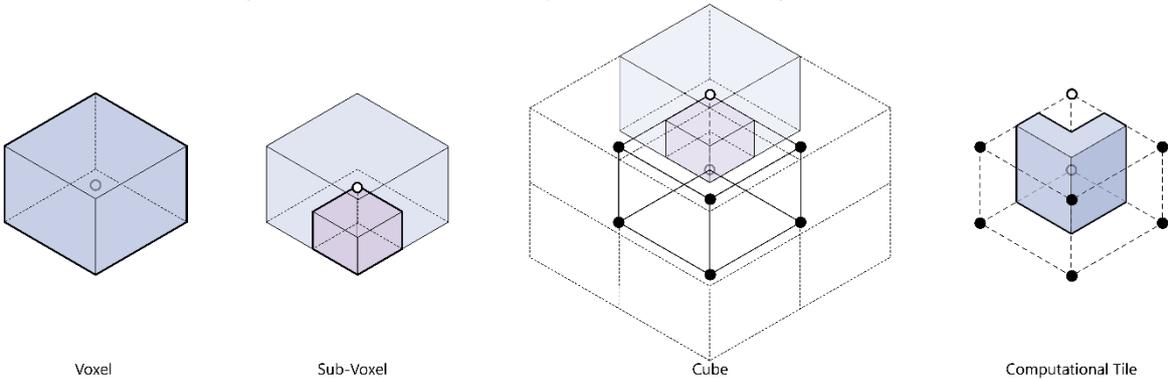


Figure 21: Visualization of Design Objects (Author)

Tile Set

Multiple tiles representing different cube configurations but offering compatibility are forming a tile set. These tilesets are the main geometrical input for both BMC and WFC algorithm. A complete tile set includes 26 tiles, all assigned to different voxel configurations.

Family

A family is a concept of an extended tileset. A family contains all tiles that are possible to connect with each other. This includes the basic tileset of 26 unique cubes but also instances of them containing other components, for example windows and doors instead of plain walls.

Building Product

A building product refers to a specific product available on the building market. The building product is linked to a unique EAN-number. The EAN13 or GTIN number is the international standard to encode an article number. The number allows to trace a product and get information about supply chain and manufacturer (Weber, 2020).

Part

A part is a modified or non-modified building product initialized in the computational workflow. A part contains exactly one building product available on the market (for example an OSB-plate), and contains additional information about the treatment of this building product. A treatment can be the resizing of certain measurements, or the drilling of holes.

Component

A component is a building entity, as it can be mounted on-site. A component contains one or more parts. Connection systems, like screws and nails are included in the information of one component. Components also display different steps of assembly within a tile, for example by separating the three components door, frame and wall.

Building Tile

A building tile is the complete entity of a discrete building component compatible with the voxel grid. It is the most important object in the workflow, as it combines information on the virtual objects with information of building components. Building tiles function as a translator between a physical building component and a virtual design environment.

2.1 Units and Sequences

Within the workflow, there is data transfer between several interfaces, algorithms and programs necessary. To ensure that no data is lost during transitions, it is essential that there is a clear distinction of directions, planes and vertices to be able to identify the specific positions of all objects during all stages of the workflow. In the following, several elementary counting sequences and units are defined. As a boundary condition, a cubic grid system is assumed.

2.1.0 Coordinate System

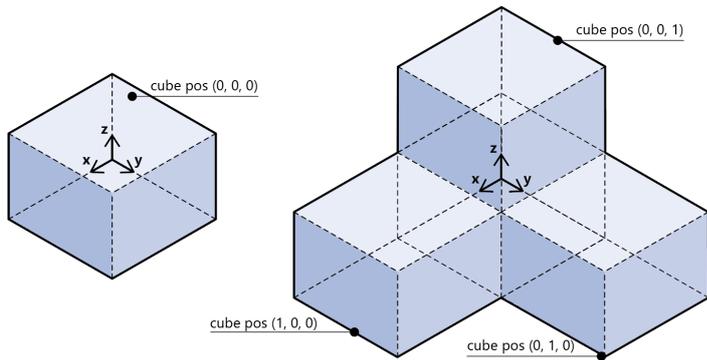


Figure 22: Coordinate System (Author)

In the coordinate system, x is defined as direction 0, y as direction 1 and z as direction 2, while assuming that the xy-plane is the ground plane. Since most of the geometries discussed in this thesis are produced with McNeel Rhinoceros, the coordinate system is set according to this software, resulting in a plain xy-plane and an orthogonal xz plane. This clarification is the basis for all further numbering of elements.

2.1.1 Cube Faces Counting Sequence

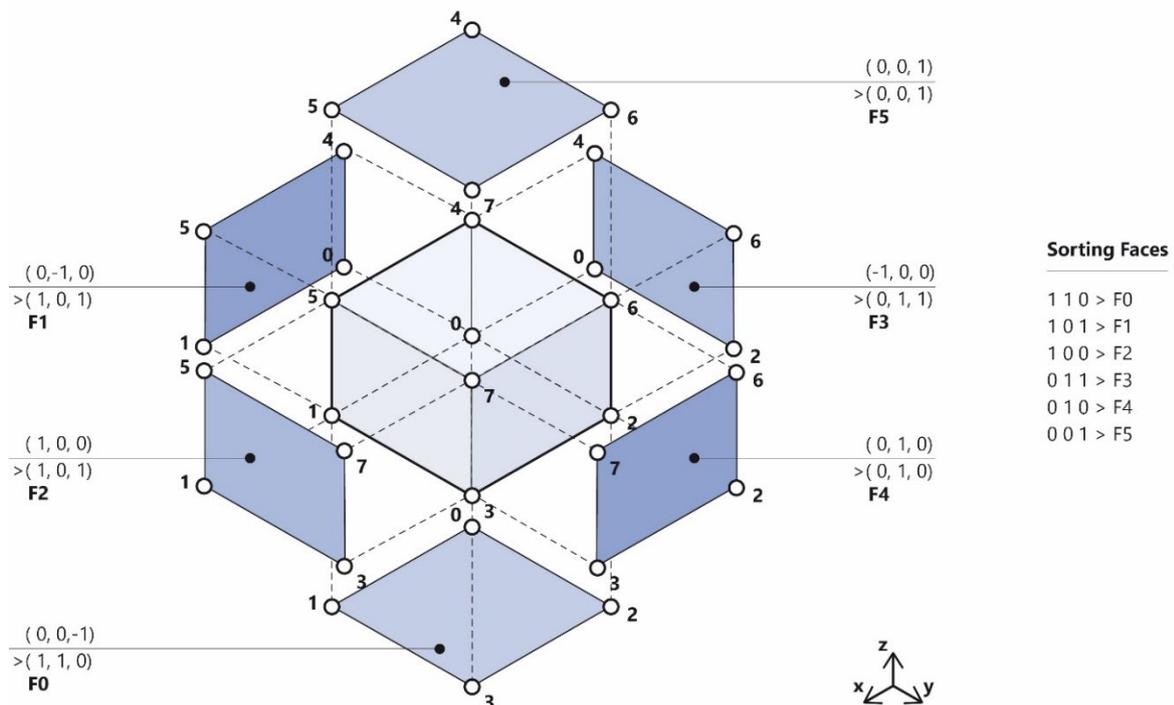


Figure 23: Derivation of the counting sequence of the cube faces (Author)

The cube face counting sequence can be derived from the coordinate system. For each of the six cube faces, the direction is defined. It is assumed, that the origin of the coordinate system is the geometric center of the cube. The six resulting vectors are now translated into a binary format through inverting negative vectors. The resulting list of binary vectors is sorted according to value of all combined digits. For example, the vector direction (0, -1, 0) is converted to (1, 0, 1) and sorted according to („101“). All six faces are now assigned a face number, used for adjacencies between different cubes and tiles in later steps.

2.1.2 Cube Vertices Counting Sequence

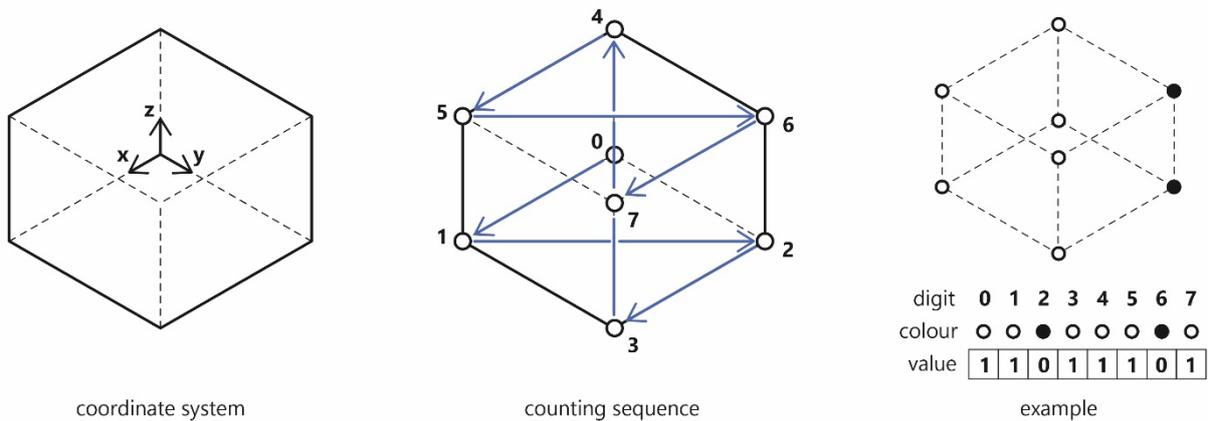


Figure 24: Counting Sequence (Author)

To be able to codify the cubes, all vertices need to be identified too. The counting is performed from the lowest sum of the coordinate of all points (-1, -1, -1) to the largest sum (1, 1, 1). From the counting sequence, a specific vertex code, checking the corner conditions of the cube, can be defined. As seen in the example, a cube with four corners defined as filled, and two defined as empty, can be translated into a binary code.

2.1.3 Establishing a Tileset of 26 unique cubes

One cube with its eight vertices can have 256 possible configurations, with the assumption of using only binary values. For the context of this thesis, it is aimed to limit the amount of configurations as much as possible to simplify the design process. Solkyu Park developed a method to reduce the 256 configurations down to 26 configurations relevant to the architectural context (Park, 2022). In a first step, point-to-point and edge-to-edge connections are eliminated since they do not have relevance in the architecture industry. These types of connections are not able to produce spaces that are usable for humans.

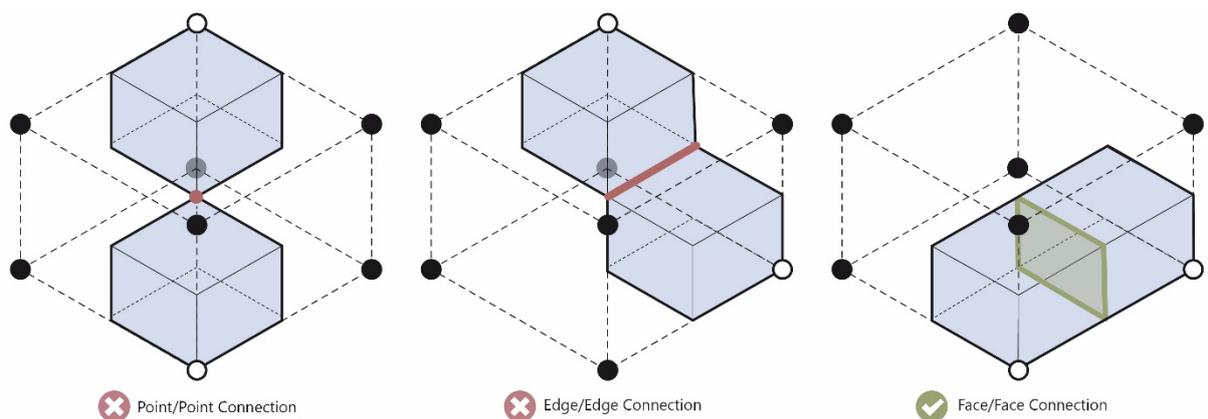


Figure 25: Connection types of sub-voxels (Author)

Only surface-to-surface connections remain, which are usable to create building parts. The stack of cube configurations can already be limited to 126. In the second step, cube configurations with the same shape but a different rotation on the xy-plane of the cube are grouped together. This reduced the stack to 33 unique cube configurations, each representing one shape, having four instances rotated around the centerpoint of the cube.

From these remaining cube configurations, seven configurations are mirrored along a horizontal axis, equivalent to seven other configurations from the stack. These configurations are grouped together as well, resulting in a total stock of 26 unique cube configurations with seven of them having eight instances, 17 having four instances and two having one instance only (Park, 2022).

These 26 unique cube configurations will be the basis for the tile design and tile structuring in the following chapters. However, for the fabrication phase, 33 cube configurations are considered since a detailed geometry might have a different setup than a mirrored instance of the same.

2.2 Dimensioning System

The dimensioning system must be chosen accordingly to human needs and human measurements. It is important to distinguish between a tatami module and a micro module. The tatami module will be the basis for the size of the building components. The tatami module can be divided into micro modules. These micro modules are the smallest unit of the system, defining connection interfaces and material changes.

To simplify the generation of the tilesets, a cuboid rectangle with a squared base was selected as the primary voxel geometry. The stairs and the toilet have a strong dependency on human measurements. Therefore, the dimensioning system is based on these two building units. According to the Dutch housing regulation, the minimum size for a toilet is 0.90m on 1.20m (van Overveld et al., 2011). Based on that, 0.90m is assumed as the minimal usable width of a building element. To this minimum dimension of the usable space, a minimum wall thickness needs to be added. As a reference, the thickness of a standard gypsum cardboard partition wall is presented. According to the Rigips system MW12RBRH, the wall has an acceptable thickness of 100mm (*Verarbeitungsrichtlinien Trockenbau*, 2021). Therefore, half of the size of the wall thickness is added to each edge of the base surface, creating a base surface of 1.00m x 1.00m. The height of the module is dependent on the stair configuration. Based on the length of 1.00m, a 25cm run on an 18 cm rise was chosen. This results in a total of 4 steps per module, leading to a height of 0.72m.

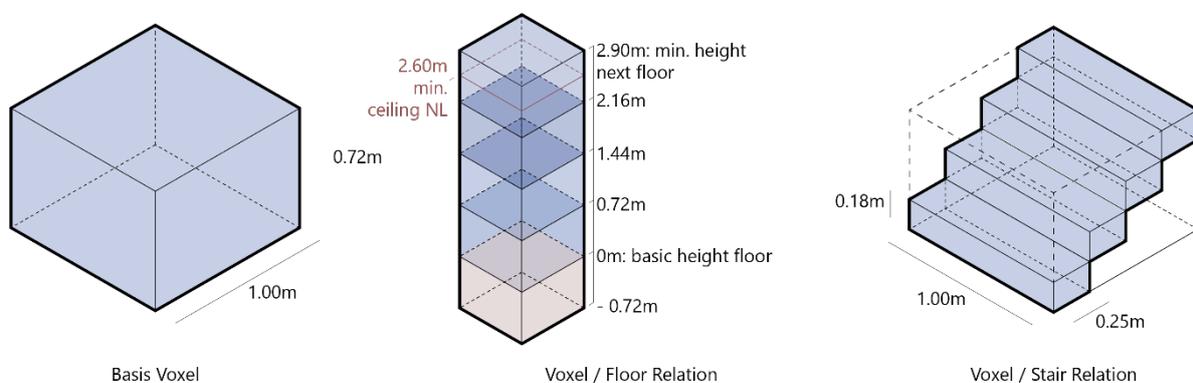


Figure 26: Dimensioning System (Author)

The stacking of these modules defines the minimum height of the second floor. The stacking of four modules achieves a height of 2.90 and is therefore sufficient for the minimum ceiling height (2.60m is required in the Netherlands (van Overveld et al., 2011)). The stacking of four modules is therefore sufficient to reach the next level with a construction height of the ceiling of 30 cm. Stacking five modules to reach the next level is required when thicker ceiling construction is used.

It is important to notice that these dimension proposals will be necessary for the creation of the tilesets in this thesis. However, they do not influence the system workflow and could be replaced with other dimensions later in the process.

2.3 Stakeholders

As the objective of this thesis is a collaborative design tool, it is important to define the stakeholders included and excluded from the design workflow.

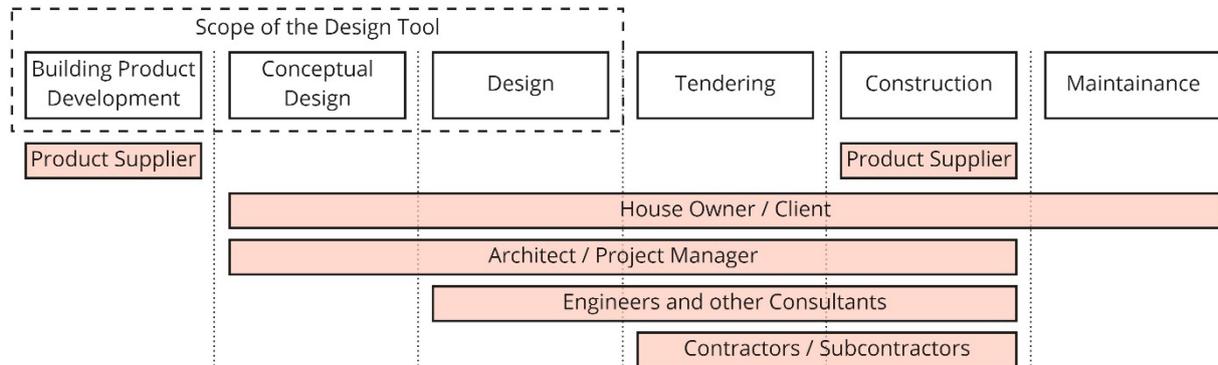


Figure 27: Stakeholders involved in different stages of the planning process (Author)

Building design and construction involve several stakeholders in various phases. The design tool focuses on the phases of product development, conceptual design, and design. The essential involved stakeholders in these phases are the building product supplier, the client, the architect and the engineer. The design tool needs to have the availability to be compatible with standard BIM-programs, such as Revit, so that in prospect also stakeholders in a later stage of the process can profit from the system. For simplification, the stakeholders addressed by the design tool are reduced to three parties: the building product supplier, the client (house owner) and the architect. The material supplier is included in the extended function of a building product designer. The house owner is in this simplified model equivalent to the end user. The architect is seen as a designer and manager, combining responsibilities of engineers, architects and project managers. These simplifications are done to reduce the focus of the addressed persons for this thesis.

2.4 Computational Framework

Based on the results of the literature research, a methodology for a design tool can be created. It was chosen to break down the design tool into several sub-steps. This approach offered the possibility to conduct experiments in each step separately without letting other parts of the tool compromise the results.

2.4.0 Topological Design

With a topological design, it is possible to open the complex spatial planning process for participation. Like in Minecraft, blocks or voxels can be placed to create spaces. The difference is, however, that the placed voxels represent space directly and do not represent built mass. Therefore, as an input, specific voxel sizes and kinds are required. A set of voxels arranged by the user is produced as an output.

2.4.1 Tile Creation

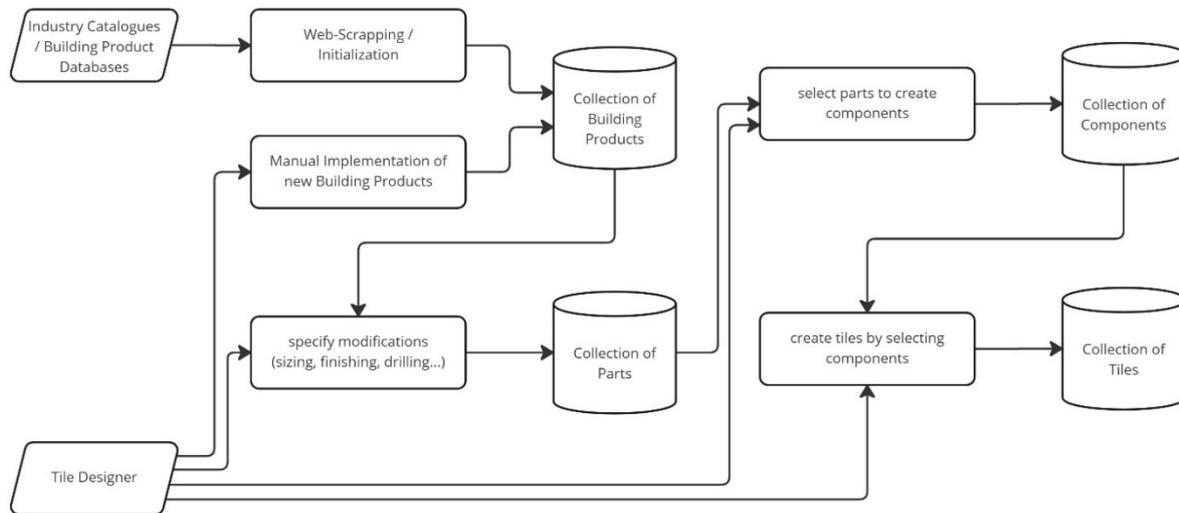


Figure 28: Flowchart of the Tile Database Structure

For the creation of the tiles, a database is required. Following the data storage approaches in Building Information Modelling software described in the literature research, a hierarchic system is introduced. The hierarchic system has the advantage of using multiple instances of objects that only need to be created once. As seen in the flowchart above, it is aimed to create for the design objects individual tables, that can be filled with instances. The lowest layer in the hierarchy is the building products table, offering the implementation of building products from the industry. Parts, Components and Tiles are design objects that relate to the IFC class `IfcElement`. Through nesting, one tile can incorporate multiple components, and one component can incorporate multiple parts. The geometrical requirements towards the database are examined in the next chapter through the development of an infill system.

The main database is aimed to be set up in the Microsoft Access format. This format has the advantage that it is an industry-standard program, and it is accessible with the widely used SQL language. The modification, input and output commands are executed with python through the pyodbc library, which can access the database.

2.4.2 Tile Options

The next step of the design tool needs to be the definition of the solution space for the tile placement. For each voxel of the topological design, cubes need to be extracted, and matching tiles from the tile database need to be mapped onto those cubes. This step is a technical necessity and does not include any manual modifications.

2.4.3 Tile Placement

The tile placement is the most essential step of the design tool, as it requires the combination of participatory design and building information modelling. The following chapter tests the Wave Function Collapse algorithm and the Boolean Marching Cubes algorithm to extract the best possible method for this step.

2.4.4 Postprocessing

For the Postprocessing, it is required to transfer the data generated in the design tool to other software or stakeholders. Strategies need to be tested, which data is relevant to export. Regarding the data transfer, the IFC format as an industry standard is chosen.

2.4.5 Aimed Level of Detail

Working with Building Information Modelling offers the possibility to design representative mass models as same as detailed fabrication-ready models. To set a standard for communication between different stakeholders working on several stages of detail, the American Institute of Architects established 2008 five levels of development, the LODs (Daga, 2021). As described in the AIA E202 contract, the following levels are defined based on (*Level of Detail* | GSA, n.d.):

	LOD 100	LOD 200	LOD 300	LOD 400	LOD 500
Title	Conceptual	Approximate Geometry	Precise Geometry	Precise Geometry	As build
Model Properties	Primitive model Roughly sized	Approximate quantity, size, location, and relationship of most objects	Object dimensions, capacities and connections are defined	Model Data ready for fabrication	A combined model of LOD 300 and 400 All building elements
Additional Data	Energy analysis (optional)	Preliminary Information for each object	Data of all objects are filled	Additional fabrication-related data	Objects contain cost, purchase and commissioning data
Implemented Layer in Design Tool	Topological Design	Interactive Tile Placement	Tile Creation Postprocessing	-	-

The last row shows the level of detail that is aimed to be implemented in the individual parts of the design tool. The topological design provides a conceptual design of spaces. The design environment for the tile placement is simplified to LOD 200. The Tile Database should contain information for at least LOD 300.

3 Development

3.0 Deconstruct the Marching Squares Algorithm

3.0.0 Applying the Marching Squares Algorithm

The Boolean Marching Cube algorithm, as explained in the last section, is an algorithm for the polygonization of an architectural configuration. To point out the functionality of the BMC algorithm, the Marching Squares Algorithm is presented first. The Marching Squares Algorithm is the two-dimensional version of the Marching Cubes Algorithm. The goal of the Marching Squares Algorithm is the generation of contour for a two-dimensional scalar field (Komura, 2008).

In the architectural context, the basis for the process is a mesh surface, for example, the floorplan of a room, in figure BMC_01 displayed through the projection of voxels. In the first step, a grid of cells is applied to the surface (figure BMC_02). Cells corners covering the surface are filled, while cell corners covering no surface are blank (figure BMC_03). This binary image is examined by the algorithm square by square, resulting in its name, the Marching Squares Algorithm.

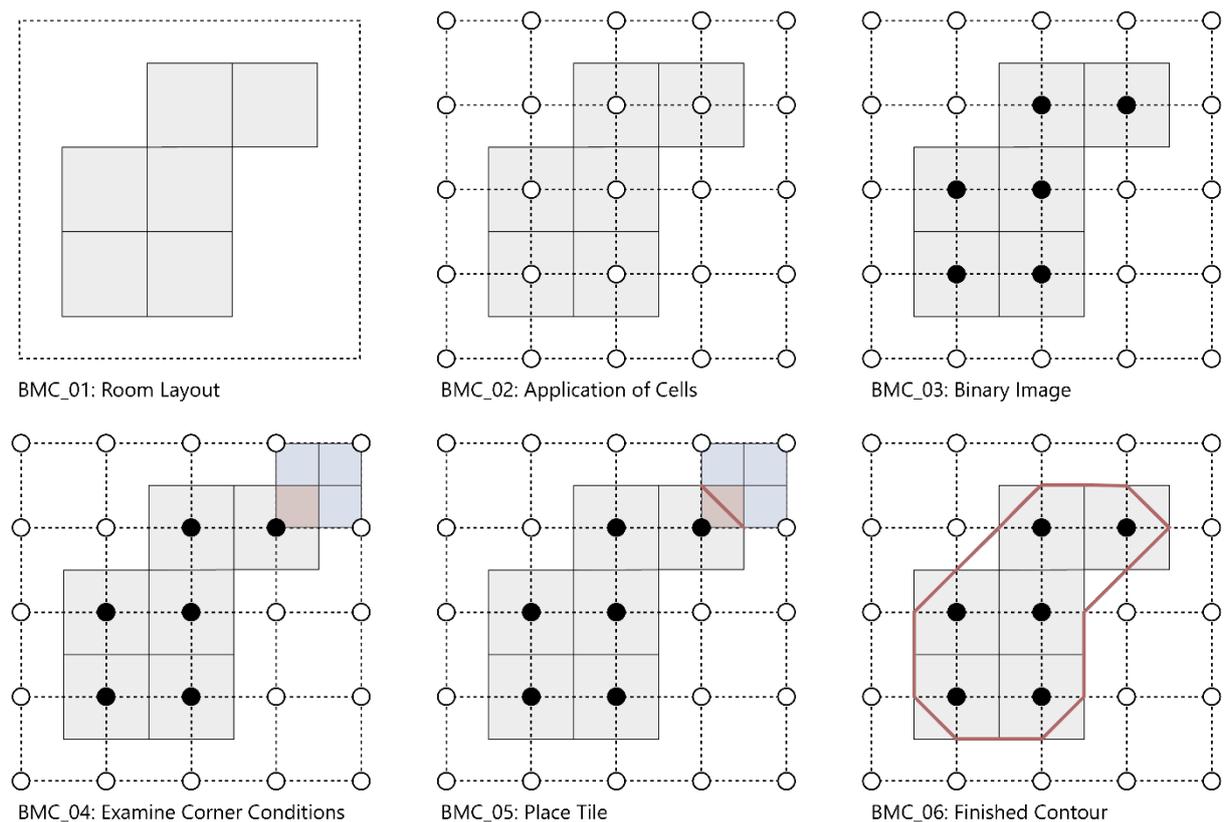


Figure 29: Applying the Marching Squares Algorithm (Author)

As seen in the figure below, for each square are 16 different options of corner configurations possible. Some of the 16 options are just rotations of another configuration, reducing the amount of unique configurations to six. For each of those configurations, a geometry is defined. In this example, it is a simple line separating blank and filled corner points with the shortest distance. The algorithm examines all possible squares in the base image and places the geometry from the fitting corner configuration for each square.

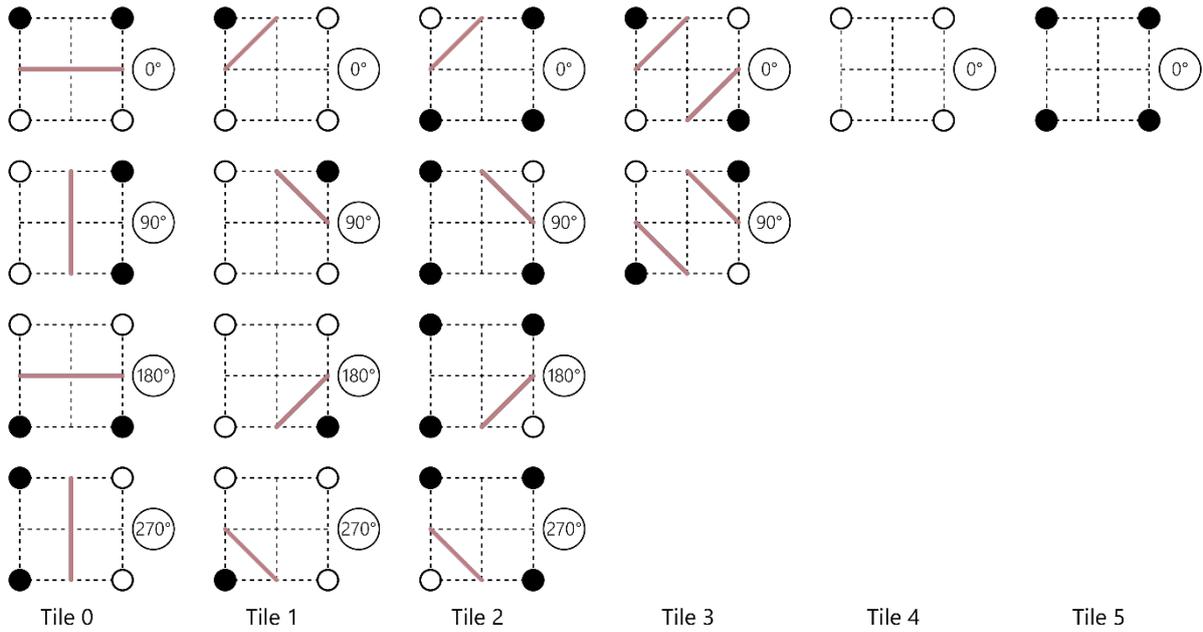


Figure 30: Marching Squares Tileset (Author)

3.0.1 Embedding the Marching Squares Algorithm into architectural context

For the modular system, there is the need to produce more than just one contour line to contain all necessary information. There is the need to generate an outer envelope, providing water tightness and protection, and individual infill systems for different rooms. The algorithm needs to run multiple times with different cell configurations to achieve that.

As seen in the figure below, the starting point is the definition of multiple rooms. Now, this information is translated to a two-dimensional lattice. Each corner point of the lattice obtains a different value, depending on if the space referring to is "room 1", "room 2" or empty. After that, a threshold is applied to the lattice to transfer it to a binary image. In the first case, the values within room 1 are set to "1" and all other values are set to "0". This way, a specific interior tileset for room 1 can be placed. The same process is repeated with room 2. For the envelope, the values for rooms 1 and 2 are both set to "1", and all other values are set to "0". This way, the envelope can cover the whole building.

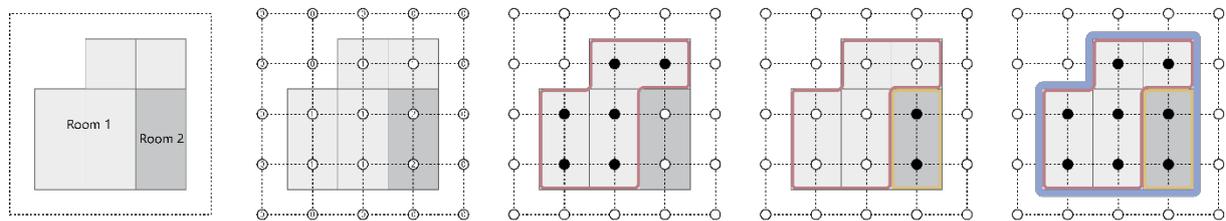


Figure 31: Applying the Marching Squares Algorithm on different fields (Author)

A detailed set of components can be designed based on the logic described above. Components are in the scope of the Marching Squares Algorithm identical to tiles. The figure below shows how an envelope and an interior tileset are applied to a building corner. To ensure compatibility of these two tilesets, the interior tileset is always placed within the voxel, while the exterior tileset is placed outside of the voxel. Both tilesets are adjacent to the faces of the voxel.

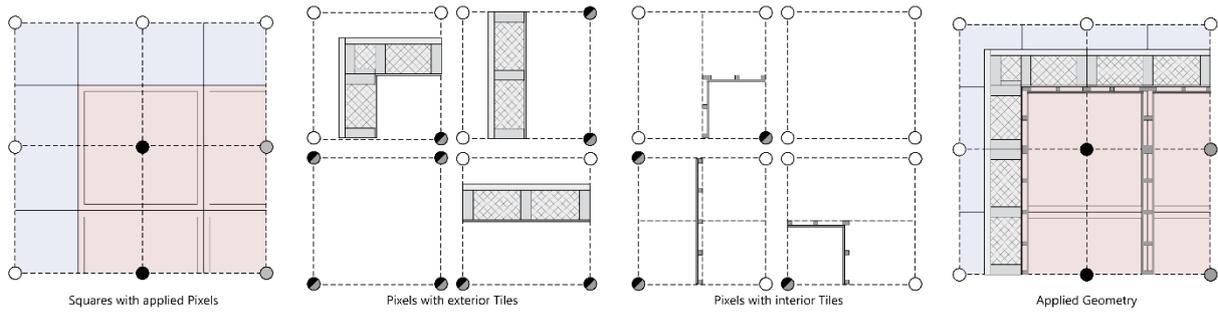


Figure 32: Applying two tilesets with Marching Squares (Author)

3.1 Building the Boolean Marching Cubes Algorithm

The Boolean Marching Cubes algorithm is implemented in the topogenesis library in Python by Shervin Azadi and Pirouz Nourian 2021. Solkyu Park developed an implementation in Grasshopper 2022. Since the individual tiles used in the workflow should be created in Rhinoceros, a new design workflow in Grasshopper was created to fit the individual needs of this project.

3.1.0 Automatizing the creation of tilesets

The first step of working with the Boolean Marching Cubes Algorithm is the generation of a basic tileset. The tileset is based on the 26 unique cube configurations mentioned in the previous chapter, so in total 26 tiles needs to be designed. In the first step, the 26 unique cube configurations are initialized as binary code, structured in the four categories "Wall Tiles", "Floor Tiles", "Roof Tiles" and "Special Tiles" as described by Park (2022).

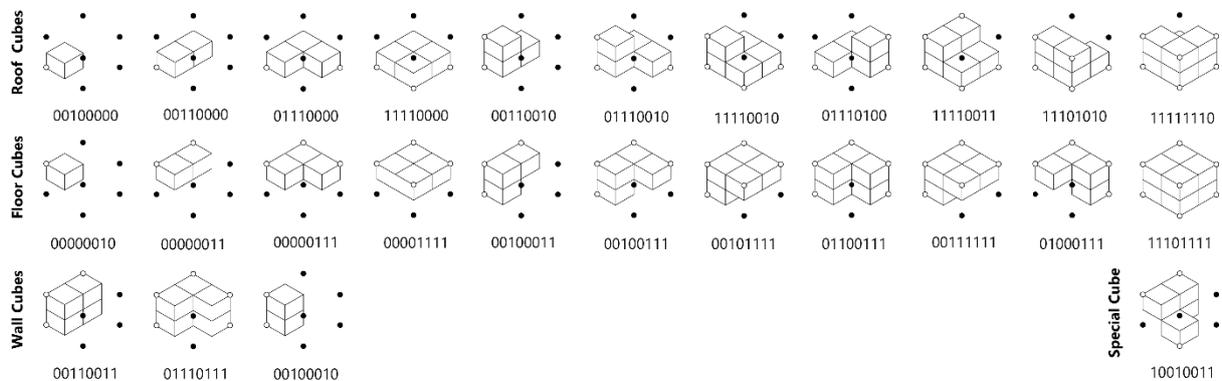


Figure 33: 26 Unique Cube Configurations (Author)

The cube configurations are then displayed on a regular grid, materializing the filled sub-voxels of each cube configuration, as seen in the figure above. The tile geometry is created by offsetting the filled- and unfilled subvoxels and intersecting the two sets. The result is a set of 26 unique tile geometries, as seen in the figure below.

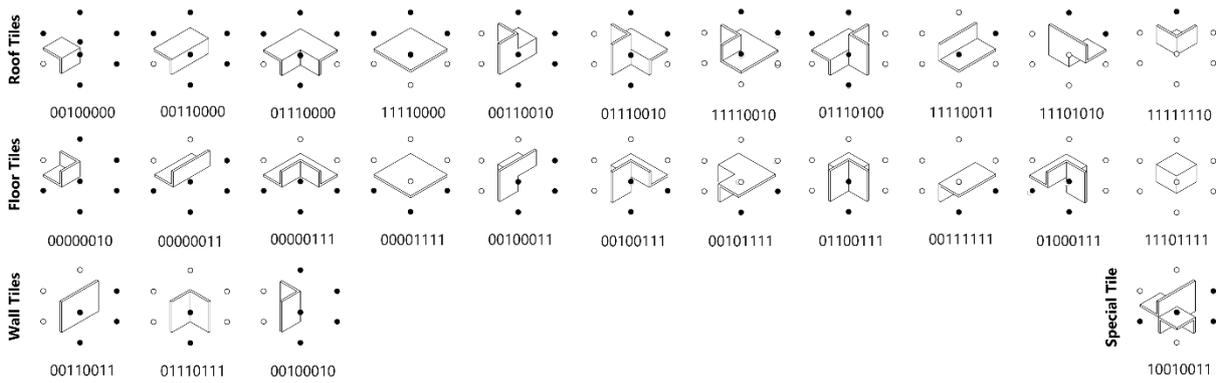


Figure 34: 26 Unique Tiles generated with Grasshopper (Author)

After the generation of these 26 tiles, it is necessary to fill all 126 cube configurations they represent. At a maximum, one unique tile represents four rotated tiles and four mirrored versions of these rotated tiles. The referred rotations are stored in relation to the tile geometry of the unique cube, as seen in the figure below.

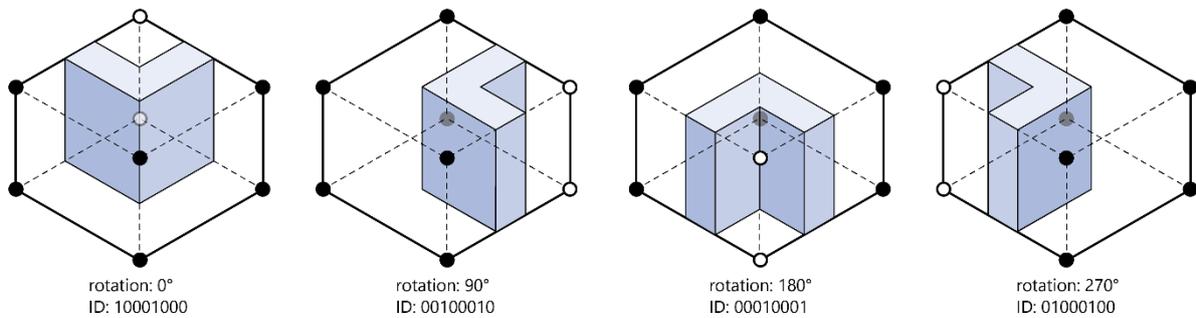


Figure 35: Calculating all possible tiles from one unique tile (Author)

An algorithm calculates the IDs of the rotated and mirrored tiles and appends these to a list with all possible tiles. The kind of rotation and mirroring of the initial geometry is dependent on the position of one specific ID in the list.

3.1.1 Applying the Algorithm

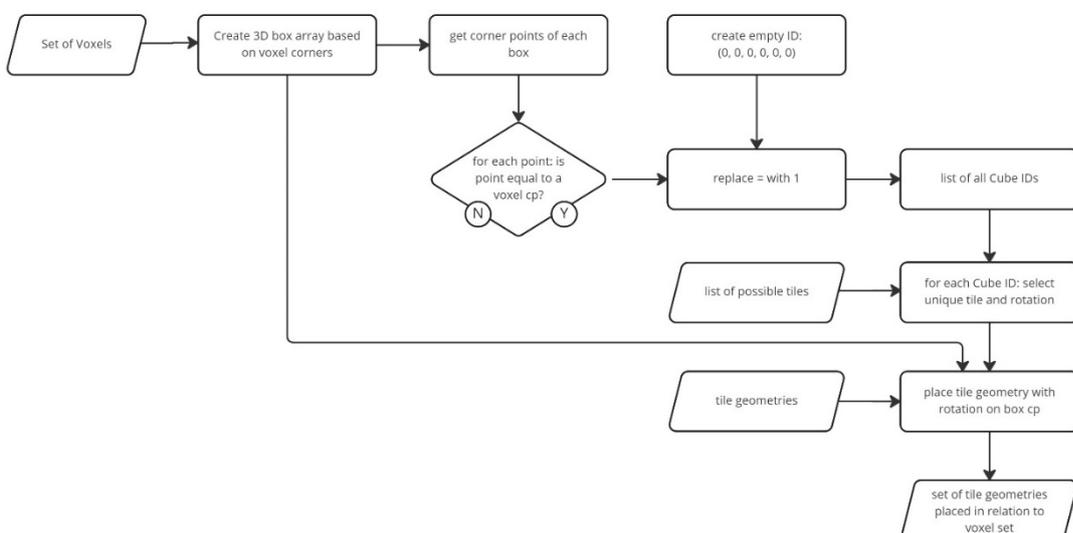


Figure 36: Flowchart of the Boolean Marching Cube Algorithm (Author)

After a tileset is created, the algorithm itself can be applied. The basis for the algorithm is a coloured set of voxels, as seen in the figure above. Each corner point of the voxel serves as the centerpoint of a newly generated cube. Now, for the corner points of each cube is identified, if it is inside or outside of a voxel. That way, an ID for all cubes can be generated. This IDs are now compared with the IDs of the input tileset, as described in the previous chapter. Each tile geometry can now be placed at the right position in the cube lattice with the required rotation.

3.1.2 Creating new Tilesets

Following the first successful application of a tileset with the algorithm, new tilesets can now be generated based on the basic tileset of the 26 unique tiles. In a first approach, a tileset with tilted tiles for the roof are generated within minutes. The workflow speeds up the creation of new tilesets and allows for more complex tile options.

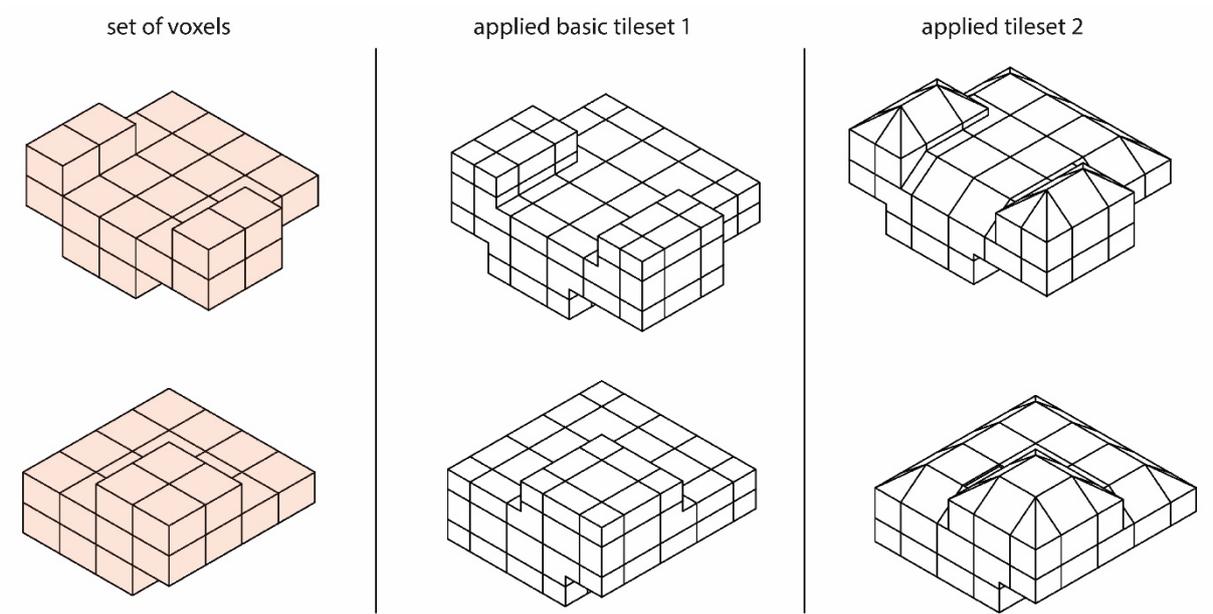


Figure 37: Applying two tilesets each on two different set of voxels (Author)

3.2 Testing the Wave Function Collapse Algorithm

Approaches to apply the Wave Function Collapse Algorithm

The WFC algorithm, as described in the literature review, can process different input strategies. The first strategy, as implemented in the original algorithm by May Gumin, is based on a sample picture, from which the relations between the individual tiles are deduced (Gumin, 2022).

A version of the original algorithm by Mateusz Bugaj is applied in Python (Bugaj, 2020). The starting point is the definition of an input image, in this case a 4x4 matrix of either black or white pixels. The rules applied in this example image are applied on a 20x20 matrix and create a new image (figure 26)

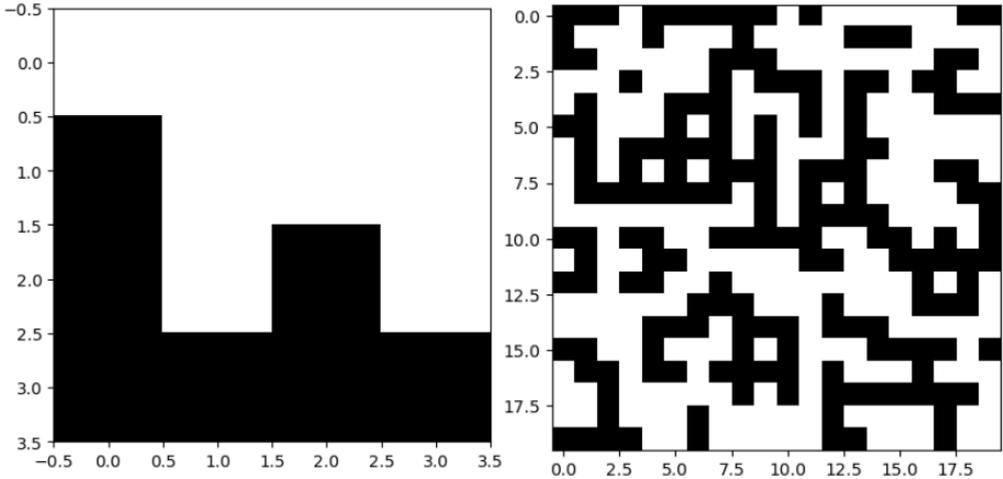


Figure 34: Input Image (4x4) for the Wave Function Collapse Algorithm, Output on a 20x20 matrix (Author)

The advantage of this input strategy is the visual representation and similarity of the output. A disadvantage is however the imprecision of the translation of pixel connections from input to output. Therefore, a second approach is tested as described by (Chasioti, 2020). For this input method, the 6 faces of a tile get assigned a specific connection profile, dependent on their geometry. Assigning different faces of multiple tiles the same connection profile allows the placement of these tiles next to each other. In the figures below, two examples of the application of an tile-based input are demonstrated.

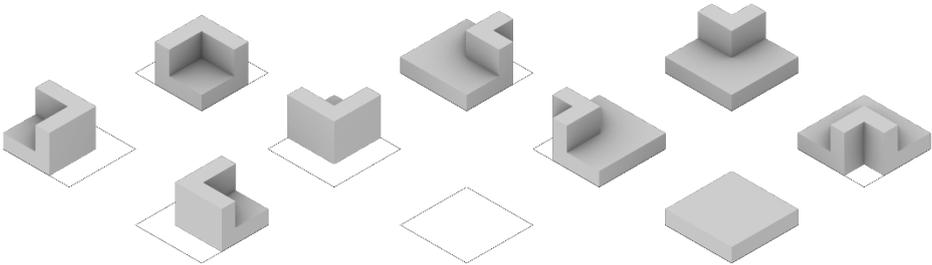


Figure 38: Tile Set to test the WFC Algorithm (Author)

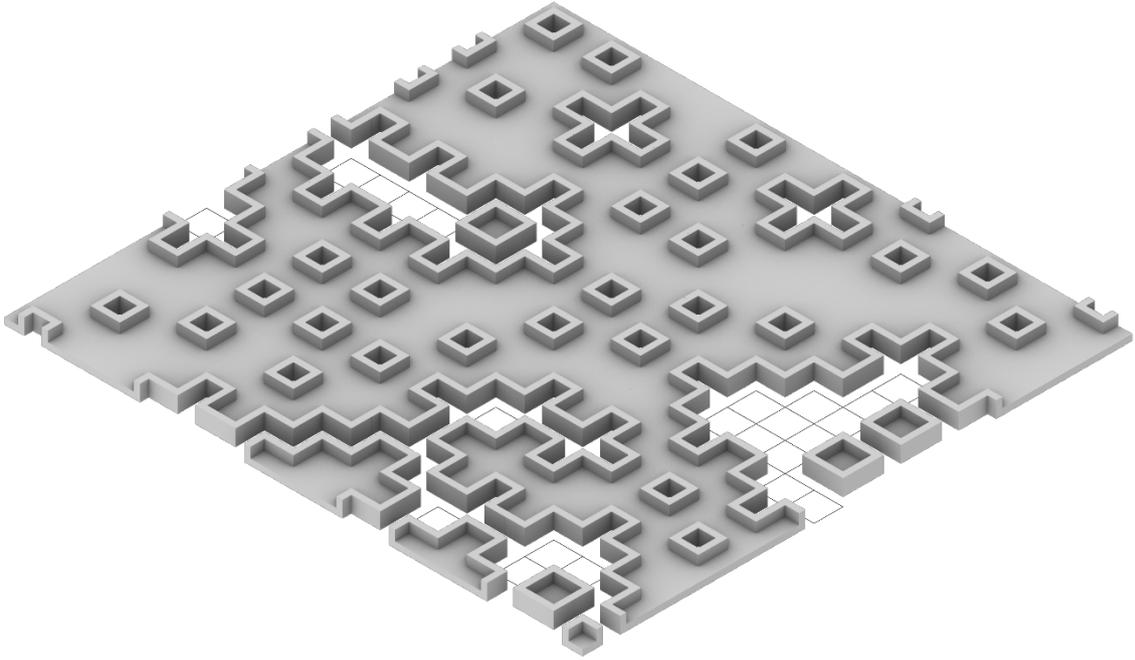


Figure 39: Applied tileset on a 20x20 matrix (Author)

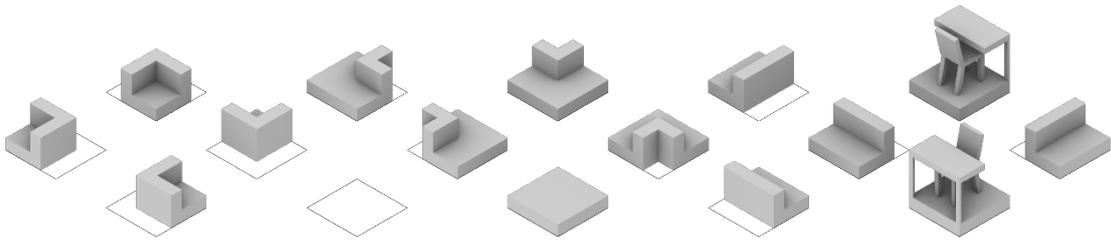


Figure 40: extended tileset with furniture (Author)

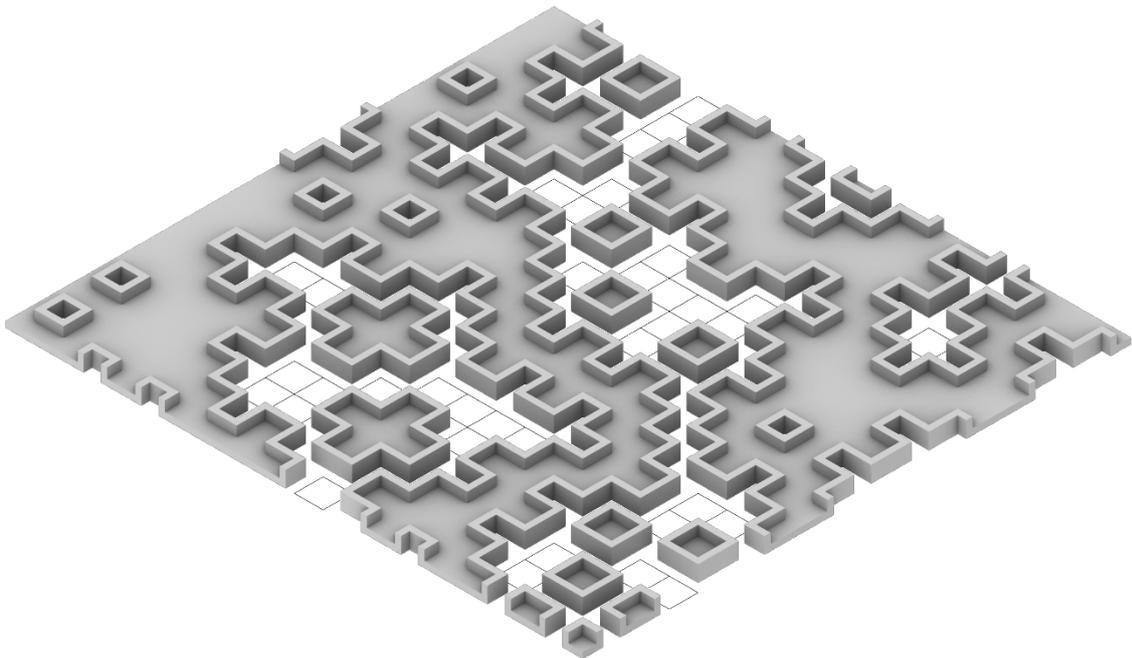


Figure 41: Application of the extended tileset on a 20x20 matrix (Author)

3.3 Building Component Development

While developing the algorithmic components, it is essential to relate to the creation of building components. As described in the Methodology, the aim is to develop an interior infill system to show the possible advantages of the discrete BIM workflow. Since reconfigurability and reusability of components is a large advantage of such a workflow, the reconfigurability of components is the central aim of the system. Furthermore, the system needs to react on tolerances and movements, and consider ergonomic handling at the construction site. Finally, the system needs to be compatible with the proposed discrete design workflow. This means that the sizing and arrangement of the components must reflect the 26 cubes described in the Design Methodology Section.

3.3.0 References

As shown in the table below, three building systems were analysed based on the established criteria. Creating a basic frame as a structural basis seems to be the most successful approach for establishing reconfigurability. Connections can be made with at least one face not mounted to when the component is finally assembled, the connections to other elements are hidden, which improves the design quality. The advantages of all three systems are combined for the design of the basic module.

System Name	Studio Bark UBuild	AUAR Home Office (ALIS Building Blocks)	Knauf W12 (wooden stud wall)
Prefabrication	✔ prefabricated frames	✔ complete prefabricated elements	✘ no prefabrication
Connection Type	In Element: Screws Element/ Element: Bolts and Nuts	In Element: Screws Element/ Element: Bolts and Nuts	Wood Screws
Materiality	18 mm plywood frame + cover	plywood frame, reinforcement + cover	sqared timber profiles (ca. 60x60) + gypsum cardboard cover
Discrete Elements	✔	✔	✘

Figure 42: Comparison of infill systems relevant for discrete design (Author)

3.3.1 Basic Module

In the figure below, the most basic infill system unit is shown. It is a simple wooden frame construction, bolted with countersunk screws. Stability against torsion forces is added though the installation of panels on both sides of the frame. In the most standard case, OSB plates are installed with each 6 screws, but also different plates (as gypsum cardboard) are possible. Assembled, the unit is 1000x720x130 mm. The weight is 11,55kg with one OSB plates mounted, and 18,3kg with two OSB plates mounted, therefore easily transportable by one person from an ergonomic perspective. The height and width were selected based on the voxel size, as described in the Dimensioning section. Materials, such as the selection of the screws, are estimated according to examples, they do not guarantee to be the best choice from a carpenter's point of view. It is necessary from the implementation of the component into the workflow, that a specific building product is linked.

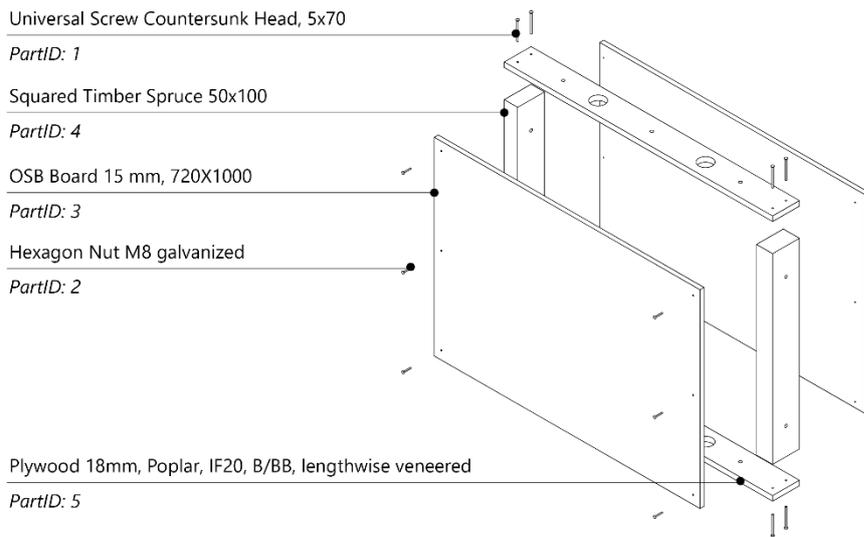


Figure 43: Basic Infill Module (Author)

3.3.2 Further Modules

Following the development of the basic module, more system elements are developed to Below, the system is showed applied in context. Based on references, transition modules to ceiling, floor, windows and doors are developed.

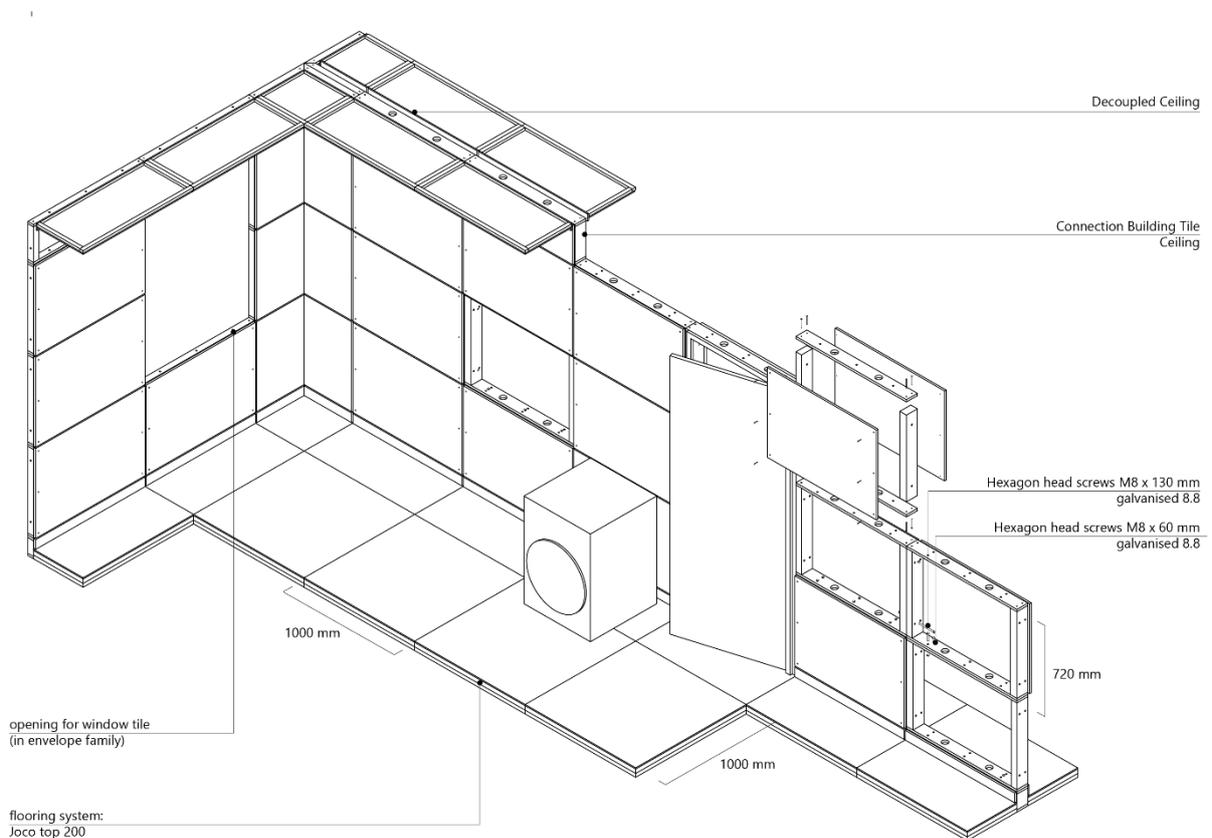


Figure 44: Extended Infill System (Author)

3.3.3 Tolerances and Movements

To tackle tolerances and movements, several strategies were developed. Firstly, through decoupling the ceiling, the interior walls can react to irregularities of the ceiling. Since the whole system neglects permanent connections such as glue or nails, the components are fully demountable. To tackle irregularities on the component level, the frame and the cladding plates can be installed in different steps. Therefore, the positioning of the cladding plates in relation to the frame can be slightly modified to guarantee aesthetic joints.

3.3.4 Integration of Services

The aim of the building system is the integration of several services. The possibility of integrating lighting, water and electricity is shown in the graphic below. Furthermore, the integrated floor is based on the Joco Floor Heating System. This system is a dry installation and completely reconfigurable.

3.3.5 Assembly

Regarding the need to integrate services and transition modules to other building elements, the strict assembly based on the tiling proposed in the Design Methodology cannot be performed in all cases. For example, the floor heating system must be installed as a whole, reaching over multiple cube modules. Furthermore, doors will be hooked in after the installation of the frame. Therefore, these operations are beyond the limitations of the theoretical system and must be considered by an architect.

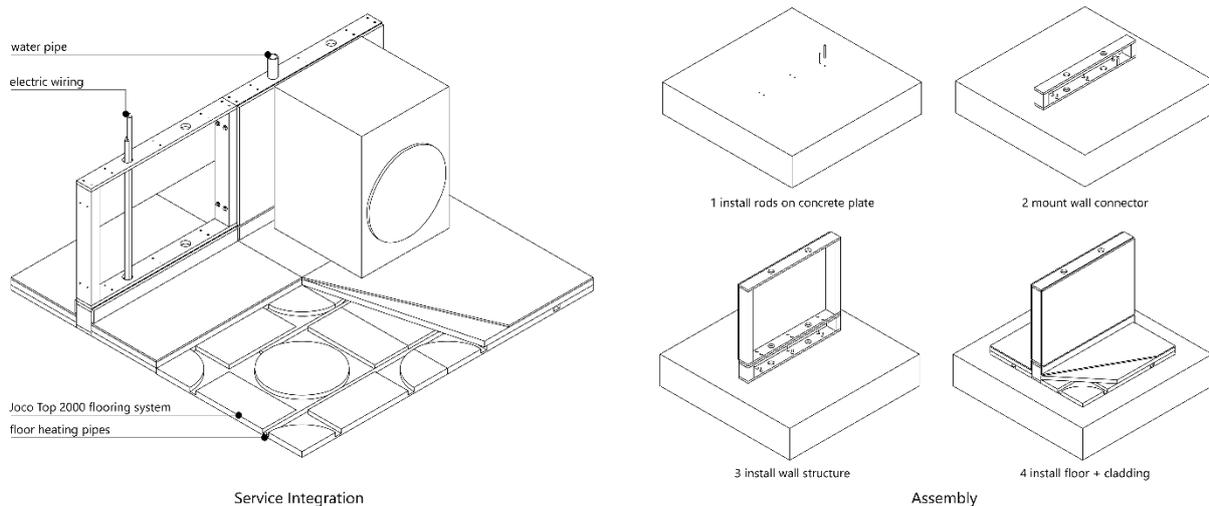


Figure 45: Services and Assembly (Author)

3.4 Conclusions

The experimentation with the Boolean Marching Square Algorithm offers the possibility of separating the envelope from the interior system to create two different tilesets. This approach reduces the complexity of rules that need to be defined for the tile placement.

It is possible to automate the tile production with a Grasshopper script. Simple tilesets for conceptual designs can be created quickly. However, severe modifications need to be performed for a more detailed approach. Therefore, the automatic generation of tilesets can only be a template for creating a qualitative tileset of building components.

The application of the Wave Function Collapse Algorithm gave new insights into the possibilities but also for the limitations for the use in an architectural context. The three possible directions of further research are connectivity exploration, information lattices and participation. The WFC Algorithm can solve connections between possible tiles and elements that are out of the scope of the BMC algorithm. For example, the WFC algorithm can solve the connection of various parts in a flat wall. It is possible to assign certain voxels values on solar incidence, noise level, and much more data through information lattices. With these lattices, the probability matrix of the WFC can be modified, that for example, sun protection elements are installed in a wall with high solar incidence. The input for the Wave Function Collapse algorithm can be the basis for a participatory system, including the user of a house or building project in the design process. A user can choose tiles and possible combinations; based on that, the whole infill can be generated.

4 Design

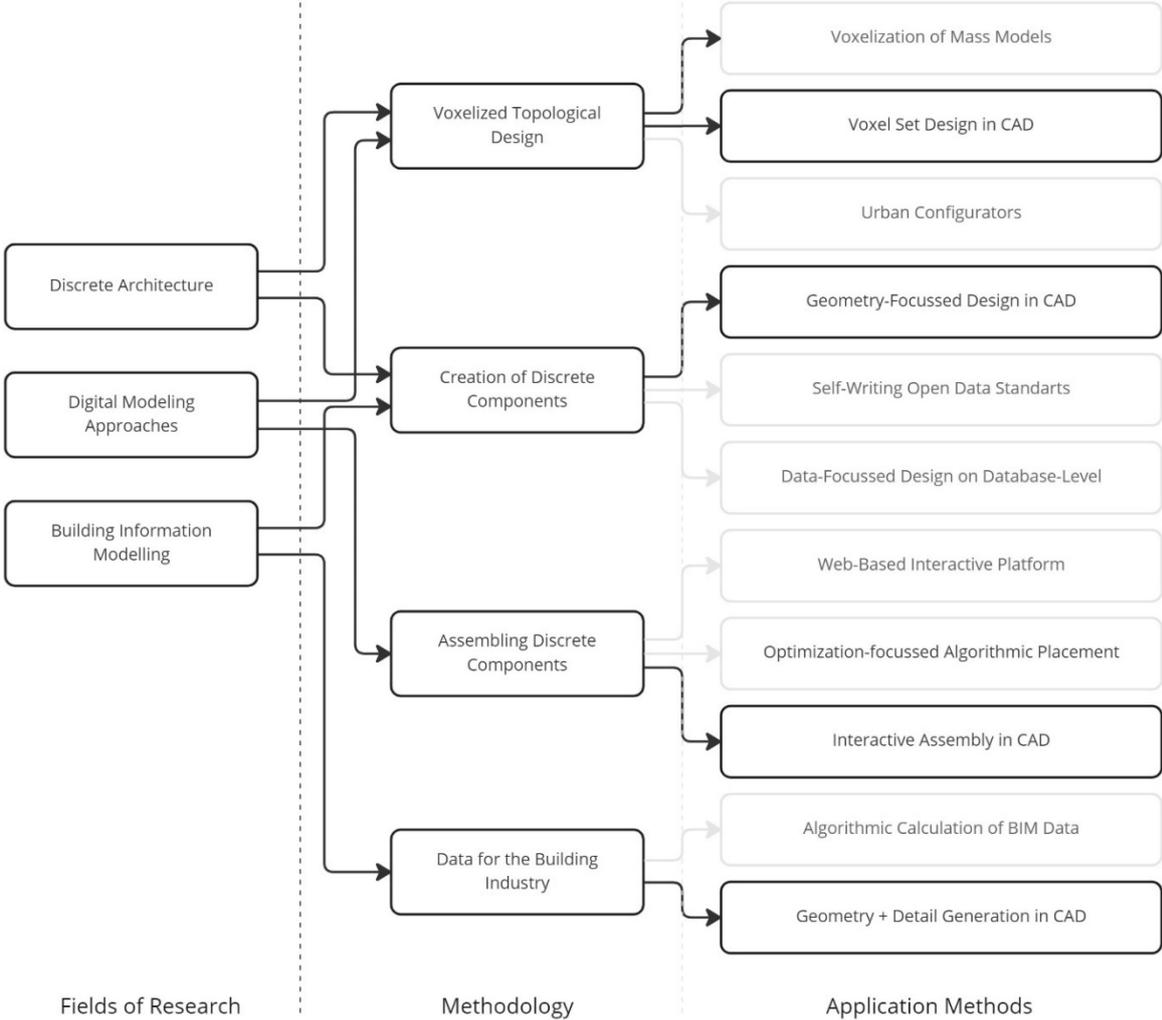


Figure 46: Application Methods for the Design Tool in context (Author)

In the Literature Research, the fields of research were examined, and in the Design Methodology Section, the framework for the topological design, creation of components, component assembly and data generation were elaborated. In the design section, some of the possible application methods for the Methodology are presented in the shape of a design tool. As presented in the figure above, for each methodology is one method elaborated, but there is potential for future research through exploiting other application methods.

4.0 Design Tool Workflow

The workflow elaborated in the following sections contains five steps, leading from initial design and tile development till export of the produced data for the building industry. Firstly, the creation and encoding of a topological design is presented. In a second step, the creation of Building Elements, called "Building Tiles", is elaborated. With the data generated through these two sections, possible tiles for the different spatial conditions in the topological design are generated. For each position of the topological design, one tile needs to be selected. This can be done by the user directly through an interactive user interface

in Rhino. After specifying each tile for the design, output data can be produced. It is possible to extract geometries and detail drawings, but also an export of BIM-relevant data in the ifc format is possible.

The workflow is designed to reach a high integration of both the professional design as same as the end-user. While some tasks need to be performed by a professional, especially the component design, a topological building framework, and data generation, the user can be involved in the modification of the topological design, as same as the selection of tiles and styles for the building.

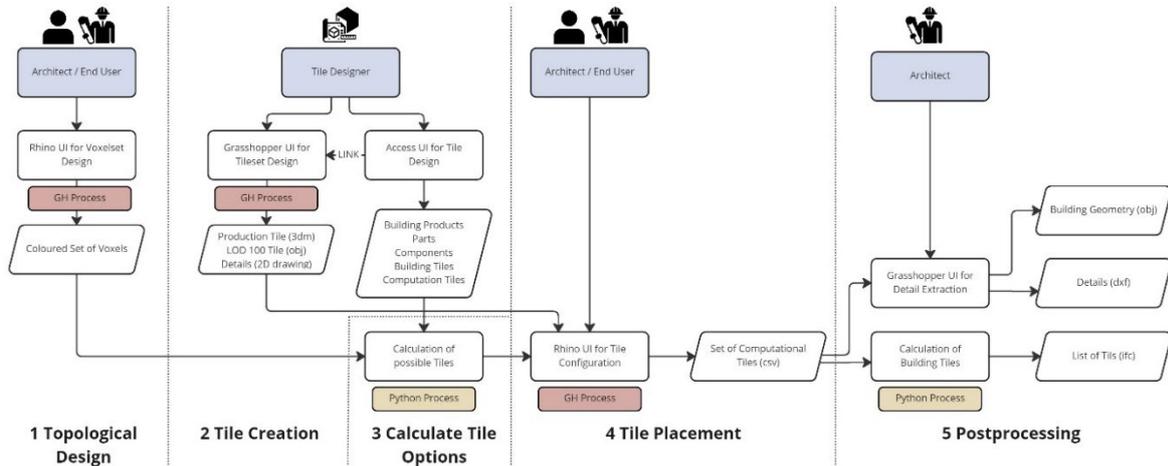


Figure 47: Flowchart of the Design Tool (Author)

4.1 Creating a topological design

The first step of the design workflow is to create a topological design of a building by an architect in collaboration with the client. The creation of a topological in comparison to a conventional design is abstracting the conceptual design to the space itself. The topological design is based on the spaces that are created within a house, not the materialization through walls, floors and ceilings. The design can therefore focus on the individual spatial needs of a user. The materialization can be performed in a later stage of the process. The final geometry can be re-translated to voxel, so a design loop between space, function and geometry evolves.

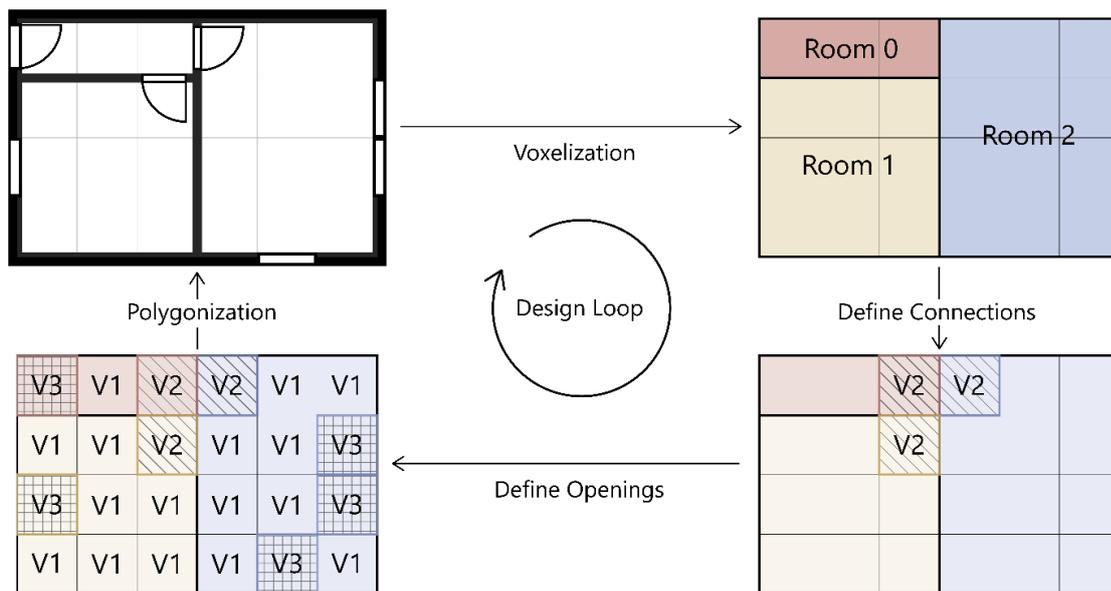


Figure 48: Materialized Design in exchange with a voxelized topological design (Author)

4.1.0 Kind of Voxels

The aim of a topological design based on voxel is to reduce the building to the most characteristic features of space that is created. The easiest approach to translate a building to a topological design is the encoding of rooms as individual units. Each room is translated to a set of voxels, that can be assigned to a specific function or design, for example a kitchen or a living room. However, housing spaces that do not have an openings cannot be accessed and are therefore obsolete. It is therefore necessary to introduce a new layer, openings. An opening is located at a specific location in space and can therefore also be represented by a voxel. Since an opening connects two spaces, either two rooms inside the house or one room with the outside space. There needs to be placed an opening voxel on both sides of the intended opening. To be able to express further features of a topological design, windows are introduced as openings allowing a visible connection, but not a passable connection for humans.

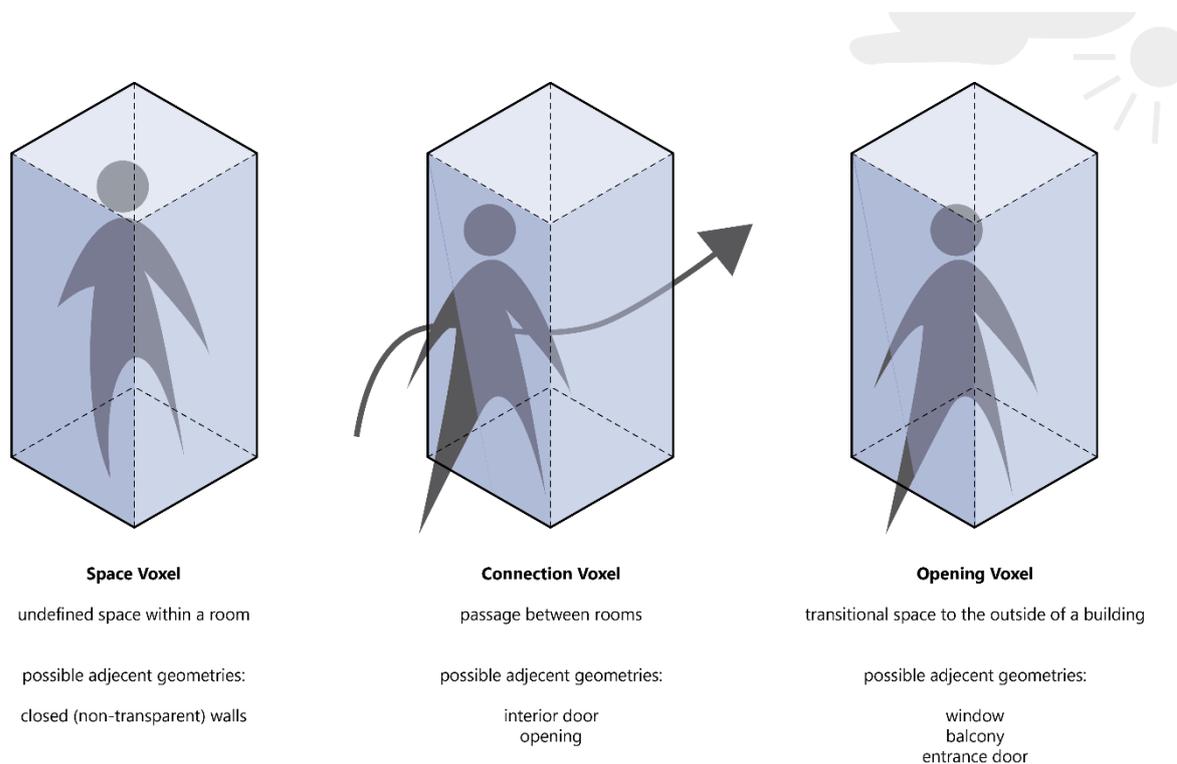


Figure 49: Kind of Voxels (Author)

4.1.1 Design Rules for the creation of voxelated topological designs

To create the voxelated topological design of a building, a few rules need to be followed to be compatible with the system.

Rule 0: Voxel Sizing

In every project, only one voxel size is valid. This means, a voxel size needs to be defined in advance of producing components, tiles and designs. All designs created in this thesis build up on the voxel size 1m x 1m x 0.72m. It is possible to choose a different voxel size, but then there is the need to create new tiles that are compliant with the adjusted voxel sizing.

Rule 1: Grid Restriction

All voxels used in one project need to connect at least with four vertices to four vertices of another voxel.

Rule 2: Openings

In the yx-plane, opening voxels need exactly one other opening voxel within a different room as direct neighbour.

Rule 3: Opening

Opening Voxels can only be placed to the outside of the building. This means that at least one face of an opening voxel is not adjacent to another voxel.

Rule 4: Double Assignments

All voxels are assigned to exactly one room and exactly one kind of voxel. This means, that every voxel needs to carry two statements.

Rule 5: Rooms

The smallest walkable space is the bathroom, with a size of at least 0.90m * 1.20m, with the minimum ceiling height of 2.60m. The box 0.90 * 1.20 * 2.60 is therefore the minimum space that needs to be incorporated in one set of voxel to count as a room. With the voxel size of 1m * 1m * 0.72m, at least 8 voxels are necessary to incapsulate this space and to create a valid room. Furthermore, voxels assigned to one specific room must be adjacent to at least one other voxel assigned to the same room.

4.1.2 Application of the rules in a CAD environment

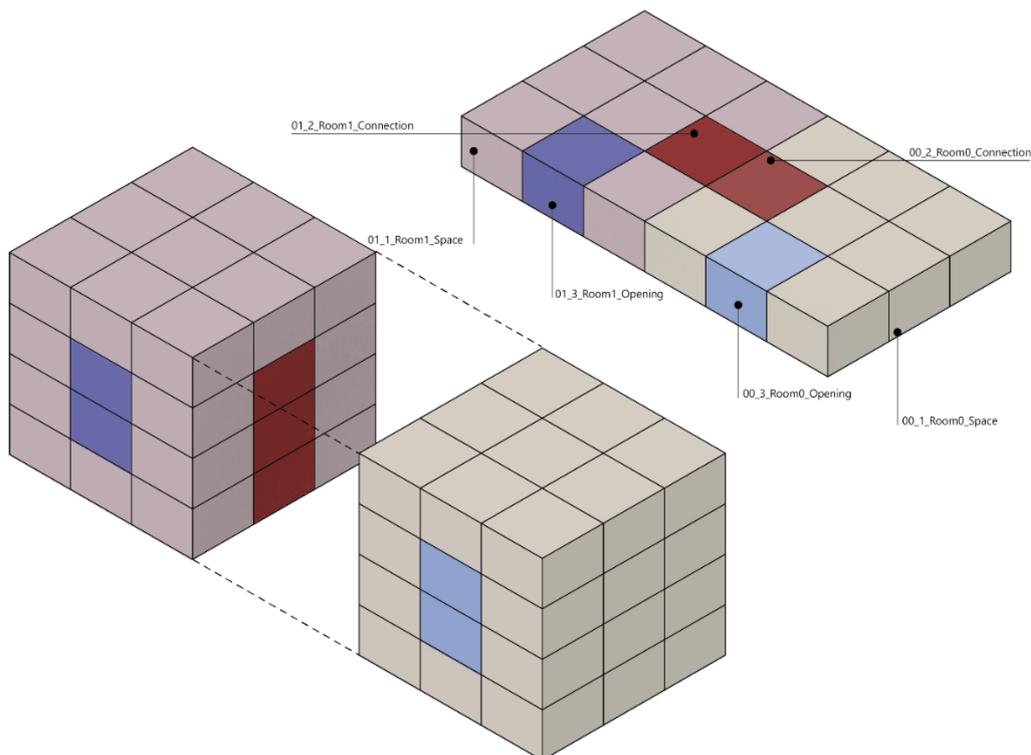
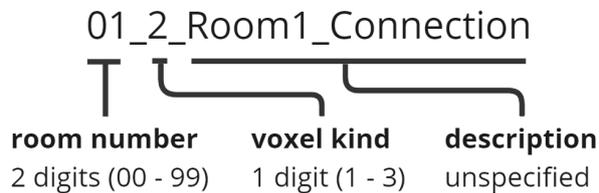


Figure 50: Topological Design and Application of Layers (Author)

The rules presented above need now to be applied in Rhino, to be able to be processed in Grasshopper in the following step. Voxels are produced as simple box geometries, but also Brep objects can be processed. All voxels must be placed on the right layers in Rhino. For that, a specific codification system is established:



When all voxels are placed and assigned to the wished layers, the topological design is concluded.

4.1.3 Encoding a topological design

Algorithm 1: Voxel Reader

```

Data: breps sorted on Rhino Layers (Voxel Room + Kind)
for every brep do:
    get Rhino layer name from brep
    extract room number and voxel kind from layer name
    append room number and voxel kind to list layer_list
end
for every brep do:
    get centerpoint from brep
    append center point to list centerpoints_list
end
create bounding box incapsulating all breps
get length, width and height from one brep as dimensions_one
get length, width and height from the bounding box as dimensions_all
for all values in dimensions_all do:
    divide value in dimensions_all with value in dimensions_one and add 1
    append result to list array_size
end
extract smallest point form the vertices of bounding box as starting_point
create array from starting_point with size array_size and geometry of one brep as cube_array
for every brep in cube_array do:
    get coordinates of the vertices as corners
    for every coordinate in corners do:
        if coordinate is equal to one value in centerpoint_list:
            append value form layer_list with the same index to required_cube
        if coordinate is equal to no value in centerpoint_list:
            append "000" to required_cube
    append required_cube to list_of_codified_cubes
    end
end
Result: (list_of_codified_cubes, array_size)

```

Figure 51: Pseudo-Code Voxel Reader (Author)

The topological design generated in the previous step needs to be translated to a processable format. The algorithm "Voxel Reader" performs this task through reading the topological design as a set of breps, and produces a list of cubes that are placed in duality to the voxels. At first, the voxel set is initialized, and

Voxel at Vertex	Encoding
① 02_1_Room2_Space	021
② empty	000
③ empty	000
④ 02_3_Room2_Opening	023
⑤ 02_1_Room2_Space	021
⑥ empty	000
⑦ empty	000

encoded cube (24-digit string):
021021000000023021000000

an array of cubes in duality to that is produced, as seen in the pseudo code. In the second step, the Cube Identifier is generated through scanning through all eight vertices of each cube in the array. As shown in the illustration below, the information of voxels that have their centerpoint at a vertex of a cube are translated to a 24-digit string. This code is the Cube Identifier, and it gets assigned to all cubes in the array. The Cube Identifier contains the information which room and kind of space is at all of its 8 vertices. Together with the array size (containing the spatial structure of the array set), the whole topological design can be stored in a csv-compatible list. An example array can be found in appendix 1.

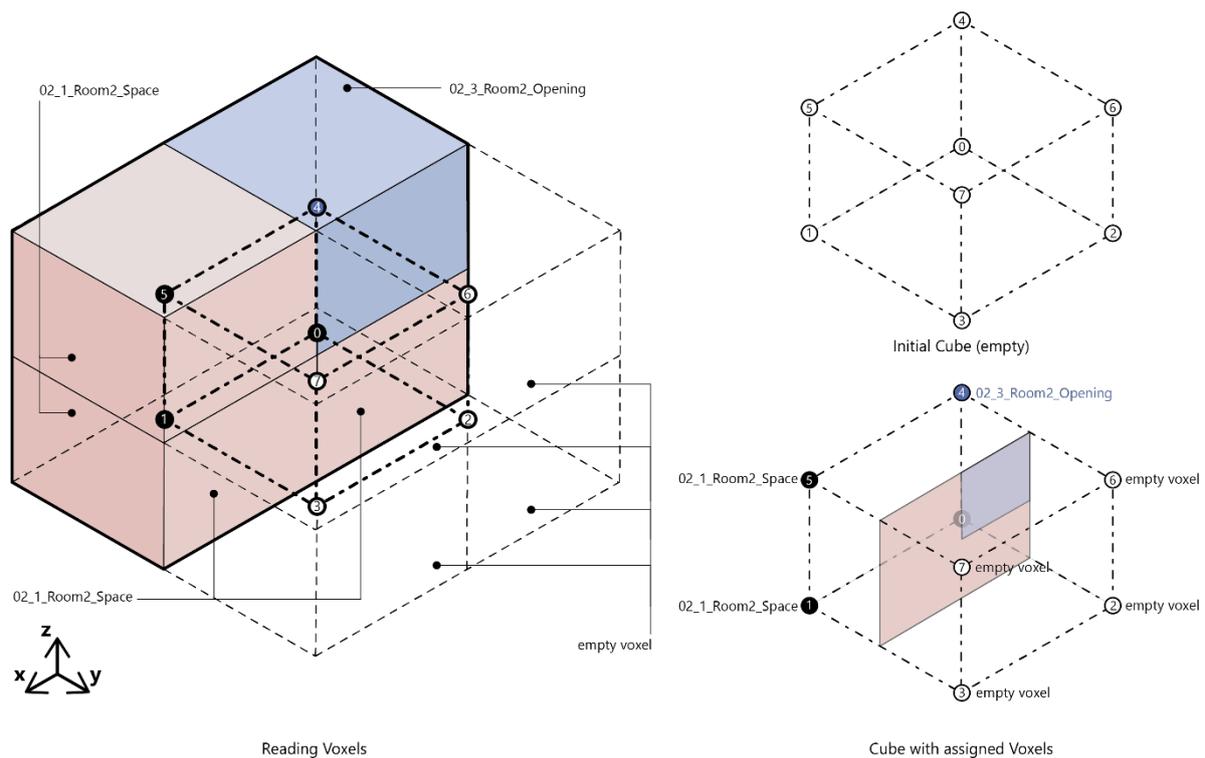


Figure 52: Encoding the Topological Design (Author)

4.1.4 Further Input Methods

Any design produced according to the presented rules can be used in the later stages of the workflow. A design can be created manually as a voxel-based design, but also other input methods can be considered. Referring to the GoDesign framework, approaches based on generative design can be implemented as long as the rules are considered. That way, the system is compatible with urbanistic and architectural configurators, such as GEN ARCH, as developed by Aditya Soman (Soman, 2021). Also, it is possible to convert a volume study adjusted to a specific site or design vision through voxelisation to a readable data structure. This can be done through topogenesis, a python library developed by Shervin Azadi and Pirouz Nourian (Azadi & Nourian, 2020).

4.2 Tile Creation

The next step in the workflow is tile creation. The process aims to design building components that can be mass-produced. The information about materiality, assembly and detail needs to be stored in a database. To be able to be compatible with the topological design developed in the previous step, the workflow aims to compatibility. The creation of building components follows a bottom-up-approach, meaning the configuration of the building component from building products.

4.2.0 Database Setup

To set up a database linking all information from building products to the final module installed on the construction site, tables are produced for all design objects. In each table, only design objects of one type are stored. Through connection of these tables, relations are instated. The figure below shows the Entity Relationship Diagram of the Databases used in this project. The database is accessed in several steps of the workflow; however, the database is only editable for the tile designer.

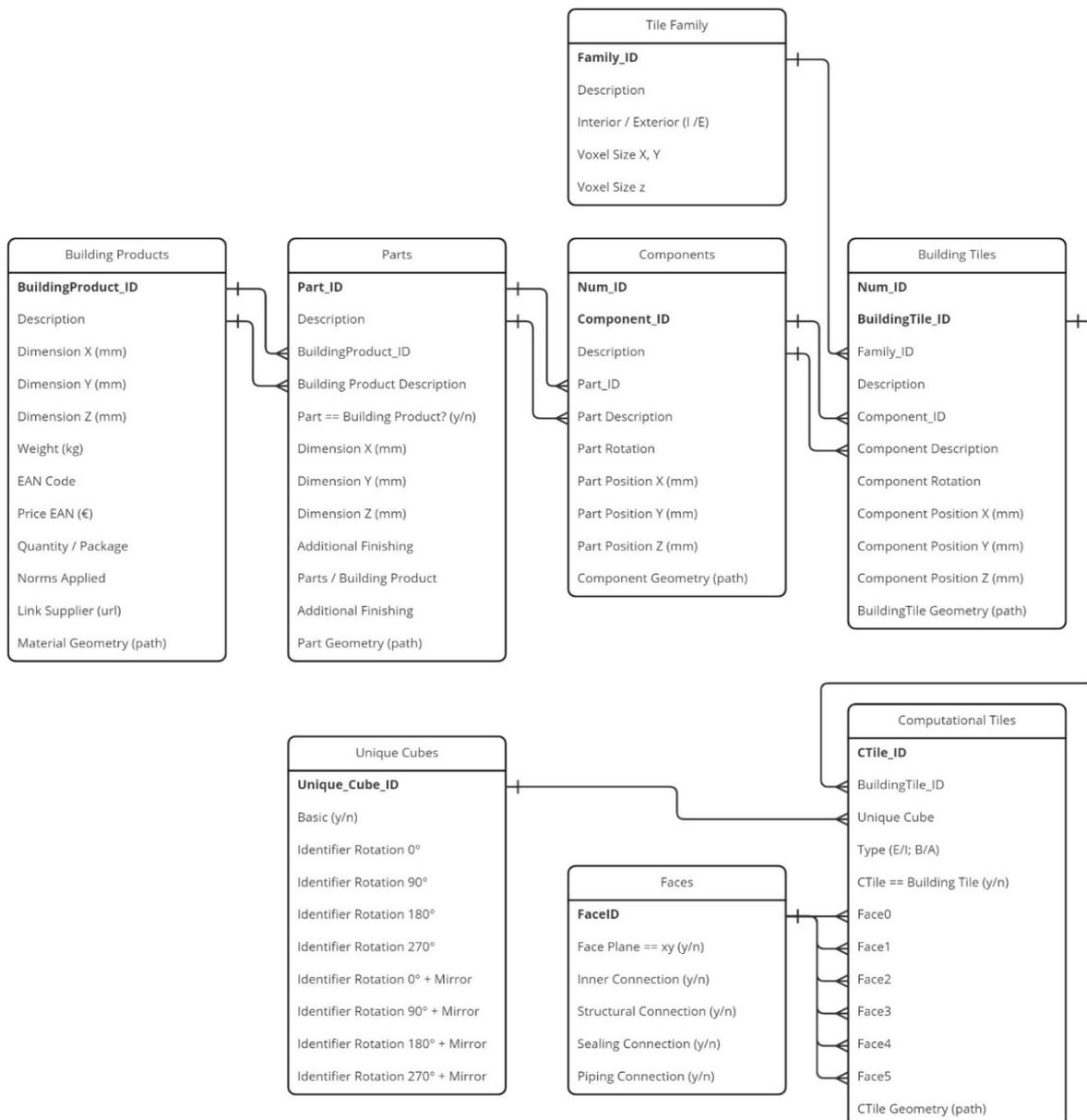


Figure 53: ERD Diagram of the Tile Database (Author)

The following steps explain the workflow by creating an interior infill wall element as building tile. All the steps, from Building Product initialization to the generation of Computational Tiles, are performed. The wall element is based on the basic module presented in the Development section and contains the same base materials.

4.2.1 Workflow

For creating Building Tiles, there are two workflows possible, dependant on the aim of the Tile Designer. The first approach assumes that the designer wants to create a Building Tile to solve a specific topological situation. In that case, the configuration of unique cubes representing the Building Tile topologically is the first step for the designer. Based on that, an appropriate family can be selected, and Building Tiles can be created with a bottom-up approach from the building product to the final Building Tile. The second workflow option assumes that the Tile Designer wants to implement one or many specific building products to work within the system. For that, building products are imported first, and based on that, a specific configuration of unique cubes is created, and a family is selected. From that point onwards, the Building Tile is designed similar in a bottom-up approach. Several steps of the process can be repeated or modified, to change the resulting Building Tile. Only once the Building Tile is approved, Faces and Computational Tiles need to be defined. The next chapters are described in the order of a topology-based design.

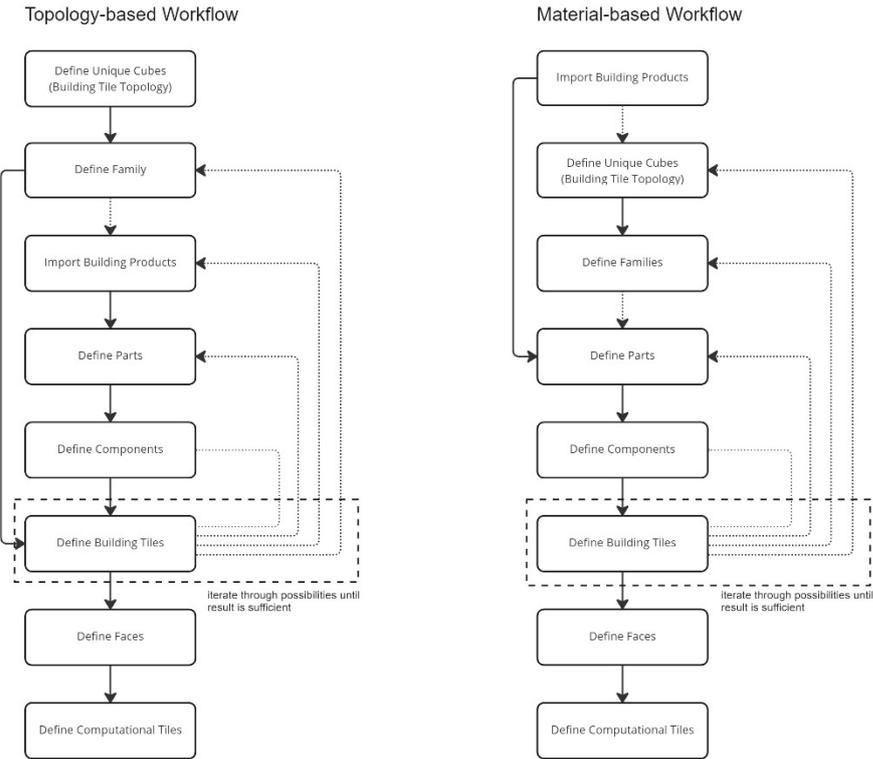
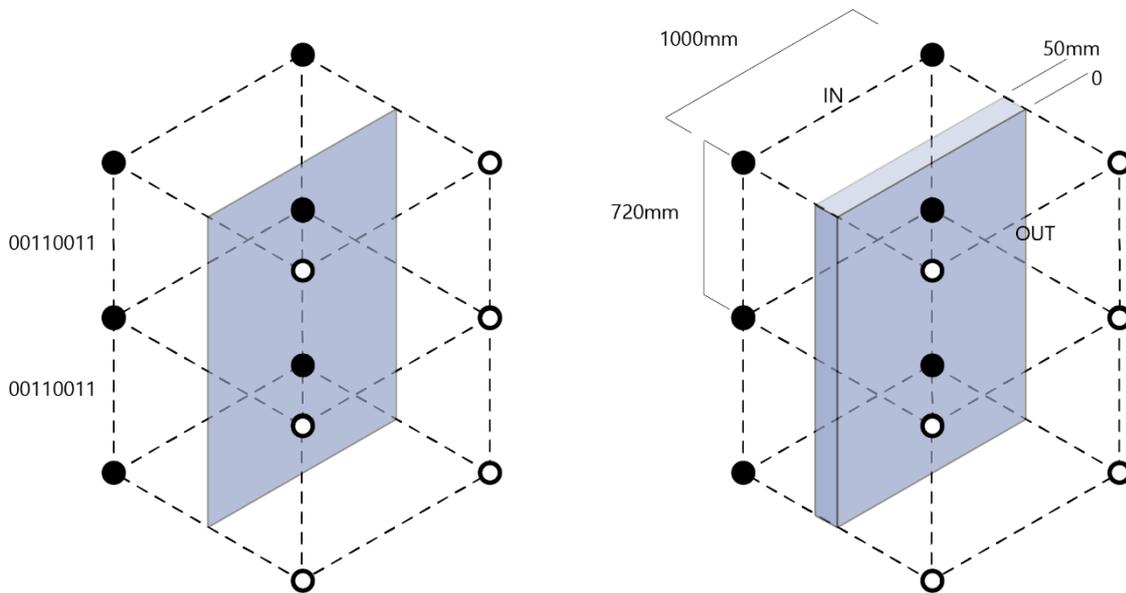


Figure 54: Two workflows to create Building Tiles (Author)

4.2.2 Unique Cubes and Families

A Unique Cubes refers to the spatial voxels, that have their geometrical center at the vertices of the cube. It contains information about a specific voxel configuration and its rotations and mirrored versions, as described in the Design Methodology. In the Microsoft Access table referring to the Unique Cubes, it is also additionally stored if the cube is a purely binary (basic) cube. Since the aimed Building Tile is a wall tile facing the outside wall, the UniqueCube_ID 1 (00110011) is assigned. However, since the tile shall be twice as high compared to the basic module, an additional UniqueCube_ID 1 is assigned. As seen in the figure below, the topological geometry is now defined.

Families are grouping several sets of Building Tiles together and are therefore essential to produce a successful tile set. To define a family, it needs to be specified if a set of interior or exterior tiles should be designed. Furthermore, the Voxel size is initialized, and the position of the approximate geometry is entered. Applied on the example, the Voxel Size 1x1x0.72m is entered, as it is the common voxel size assumed in this thesis. The tile is part of an interior infill system, assumed with the thickness of 50 mm.



Selection of Unique Cubes

Unique Cubes	Unique Cubes
Unique_Cube_ID	1
Basic (y/n)	y
Identifier Rotation 0°	00110011
Identifier Rotation 90°	01010101
Identifier Rotation 180°	11001100
Identifier Rotation 270°	10101010
Identifier Rotation 0° + Mirror	-
Identifier Rotation 90° + Mirror	-
Identifier Rotation 180° + Mirror	-
Identifier Rotation 270° + Mirror	-

Application of a Tile Family

Tile Family	Tile Family
Family_ID	1
Description	Interior_Infill
Interior / Exterior (I /E)	I
Voxel Size X, Y (mm)	1000
Voxel Size z (mm)	720
Offset Geometry (mm)	50

Figure 55: Unique Cubes and Families (Author)

4.2.3 Building Products

As described in the literature review, a broad stock of building material databases is already available online. From these databases, information for millions of building products can be extracted. Automated scrapping of this information would be the optimal information access, but in the scope of this thesis, the information is entered manually. For the demonstration tile, five different building products are needed. The figure below shows the insertion of a wood screw in the table "Building Products" as a new entry. For each new entry, an ID is generated automatically, which is used to identify a certain building product in the later process. Next to the dimensions, it is possible to register BIM-relevant data, such as weight, price and applied norms. The building product geometry can be imported directly from the manufacturer or created in a CAD-program. The location of the geometry file is linked to the table.

Building Products	87
BuildingProduct_ID	87
Description	Universal_Screw_Countersunk Head_5x70
Dimension X (mm)	70
Dimension Y (mm)	9.7
Dimension Z (mm)	9.7
Weight (kg)	-
EAN Code	4003530109560
Price EAN (€)	21.51
Quantity / Package	200
Norms Applied	DIN 18101
Link Supplier (url)	https://www.spax.com/en/p....-1191010500705/pid-814/
Building Product Geometry (path)	Geometry_Data/Materials/87_Universal_Screw_Countersunk Head_5x70.obj



Figure 56: Table: Demonstration of a Material Import (Author)

Figure 57: Countersunk Screw (Spax)

4.2.4 Parts

While building products display information about actual products from the industry only, parts extend this information by adding data about changed dimensions through cutting, modified shapes, or additional finishings. For the demonstration, an OSB plate available on the market of 2500 x 1200 x 15 mm is implemented. However, a dimension of 1430 x 990 x 15mm is required for a specific building element. This modification in size is expressed in a changed dimension in the Part table. Each Part is linked to exactly one BuildingProduct ID. However, it is possible to create multiple Parts referring to one building product, as for example a gypsum cardboard can be cut down to several dimensions. The geometry for the part can be re-used from the building product if the dimensions are the same, otherwise it can be designed in a CAD program and linked to the table.

Parts	Parts
Part_ID	2
Description	OSB_15_1430_990
Material_ID	32
Material Description	OSB_15_2500_1250
Part == Material (y/n)	y
Dimension X (mm)	1430
Dimension Y (mm)	990
Dimension Z (mm)	15
Additional Finishing	-
Parts / Material	1
Material Geometry (path)	Dataset/Geometry_Data/Parts/2_OSB_15_1430_990.obj

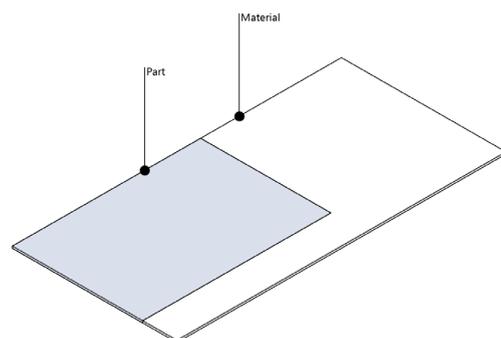


Figure 58: Demonstration of Part Implementation (Author)

4.2.5 Components

Components are functional units made from one or several parts. Therefore, compared to the previous categories, one component needs to eventually link to several parts and their position and rotation in relation to the component origin. This requires one entry for each Part Instance linked to a component. In the example below, one component is created from two parts, a gypsum cardboard plate and a screw. Since 11 part instances are used, there are 11 entries filled in the components table, linking all to the same Component ID.

Components	Components	Components	Components
Num_ID	54	55	56
Component_ID	3	3	3
Description	Cladding_OSB_DoubleTile	Cladding_OSB_DoubleTile	Cladding_OSB_DoubleTile
Part_ID	2	3	3
Part Description	OSB_15_1430_990	Universal_Screw_5x70	Universal_Screw_5x70
Part Rotation	Rotate (90.0°) about +X	Rotate (90.0°) about +X	Rotate (90.0°) about +X
Part Position X (mm)	0	-470	-470
Part Position Y (mm)	-40	0	0
Part Position Z (mm)	0	0	-360
Component Geometry (path)	3_Cladding_OSB_DoubleTile.obj	3_Cladding_OSB_DoubleTile.obj	3_Cladding_OSB_DoubleTile.obj

Figure 59: Assigning Components (Author)

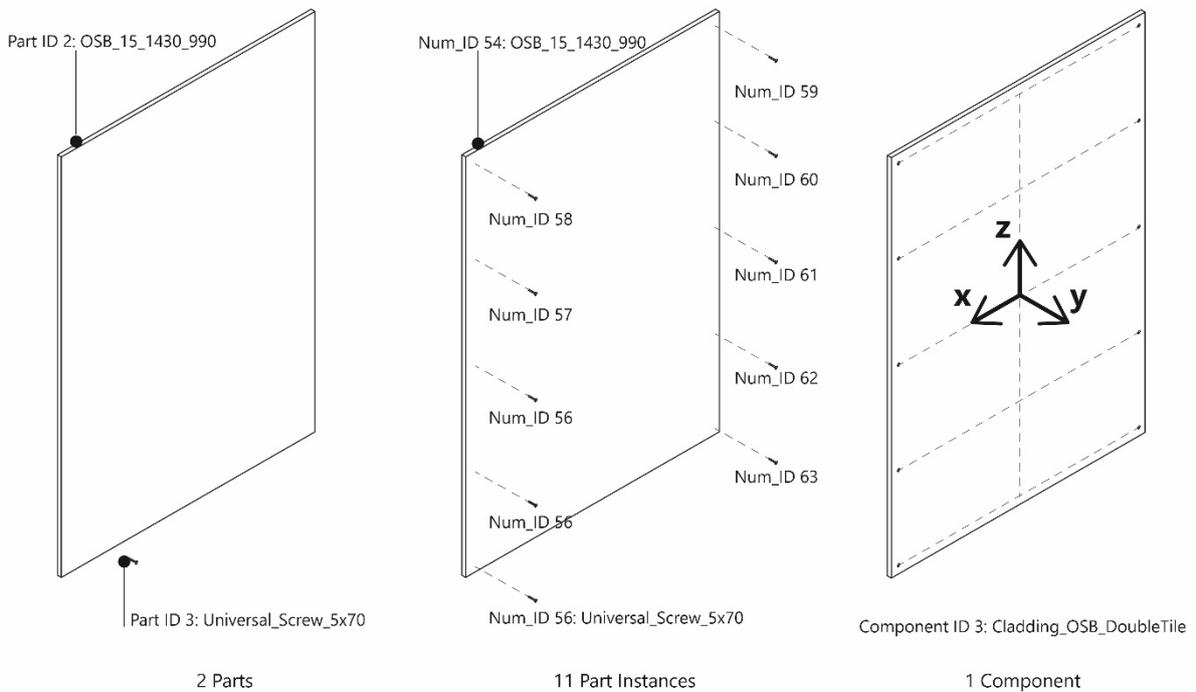


Figure 60: Representation of a component through Parts, Instances and as a whole (Author)

4.2.6 Building Tiles

In the next step, several components are grouped to a unit compatible with the topological design: The Building Tile. The setup of the data format is identical to the component table: One entry in the table represents the position and rotation of a component instance in relation to the Building Tile origin. In the example below, a Building Tile is created from two components. The assembled Building Tile fits in the spatial grid defined in the Family.

Building Tiles	Building Tiles	Building Tiles
Num_ID	16	17
BuildingTile_ID	2	2
Family_ID	1	1
Description	Wall_Double	Wall_Double
Component_ID	3	4
Component Description	Cladding_OSB_DoubleTile	Frame_DoubleTile
Component Rotation	Rotate (180.0°) about +Z	Rotate (180.0°) about +Z
Component Position X (mm)	0	0
Component Position Y (mm)	25	-7.5
Component Position Z (mm)	0	0
BuildingTile Geometry (path)	BuildingTile Geometry (path)	BuildingTile Geometry (path)

Figure 61: Assigning Building Tiles (Author)

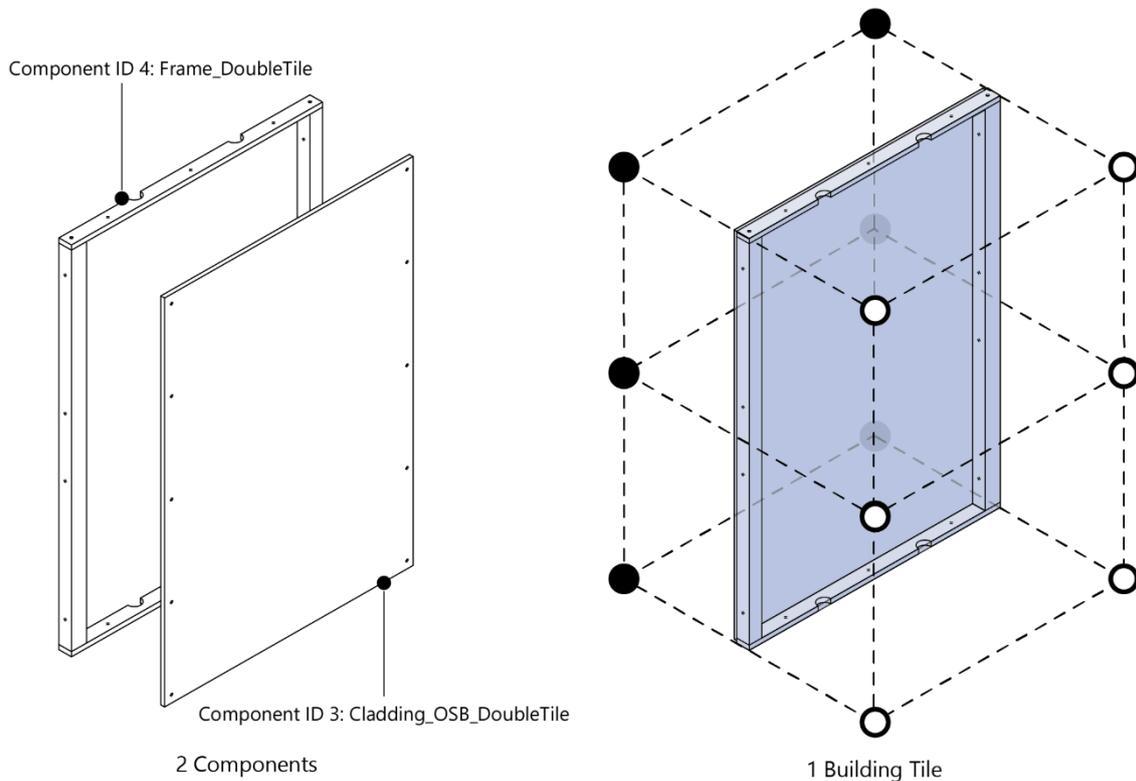
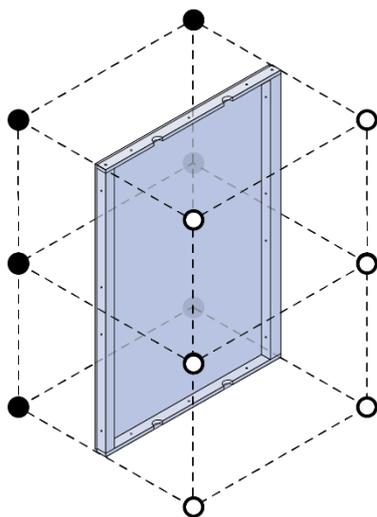


Figure 62: Assembly of a building tile (Author)

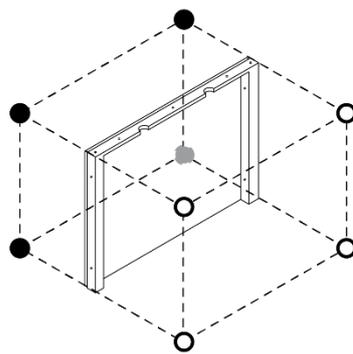
4.2.7 Computational Tiles

Once the Building Tiles are defined, they can be split into Computational Tiles so that they are processable with topological designs. This is done by slicing the geometry along the voxel grid defined in the Family table. As seen in the example below, the Computational Tile table contains information about the connectivity of all six faces and the referred Building Tile of each computational tile. The calculation of the faces is described in the next chapter.

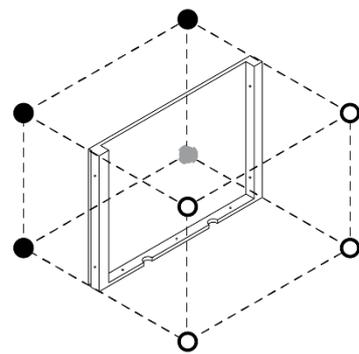
Computational Tiles	Computational Tiles	Computational Tiles
C Tile_ID	23	24
BuildingTile_ID	2	2
Unique Cube	00110011	00110011
Type (E/I; B/A)	I_A	I_A
C Tile == Building Tile (y/n)	n	n
Face0	INT1	EXT2
Face1	0	0
Face2	EXT1	EXT1
Face3	EXT1	EXT1
Face4	0	0
Face5	EXT2	INT1
C Tile Geometry (path)	I_A_23_00110011.obj	I_A_24_00110011.obj



BuildingTile_ID 2



Computational Tile ID: 23



Computational Tile ID: 24

Figure 63: Demonstration of the creation of Computational Tiles (Author)

4.2.8 Faces

Faces enrich the information given by the cube configuration to specify internal or external connections of Computational Tiles. The encoding of the faces is necessary, as otherwise Building Tiles would not be able to be constructed after an algorithm or a user modifies a spatial configuration. There are two types of connections, internal and external connections. Internal connections are referring to a fixed connection within one building tile and must therefore be complied. External connections can be a structural, sealing or piping connection, limiting the adjacent faces of other components. In the scope of this thesis, only internal connections were applied in the design tool, but in prospect, it would be an enrichment for the system to implement external connections. Below, an example is shown, in which all faces of the computational tiles are analysed.

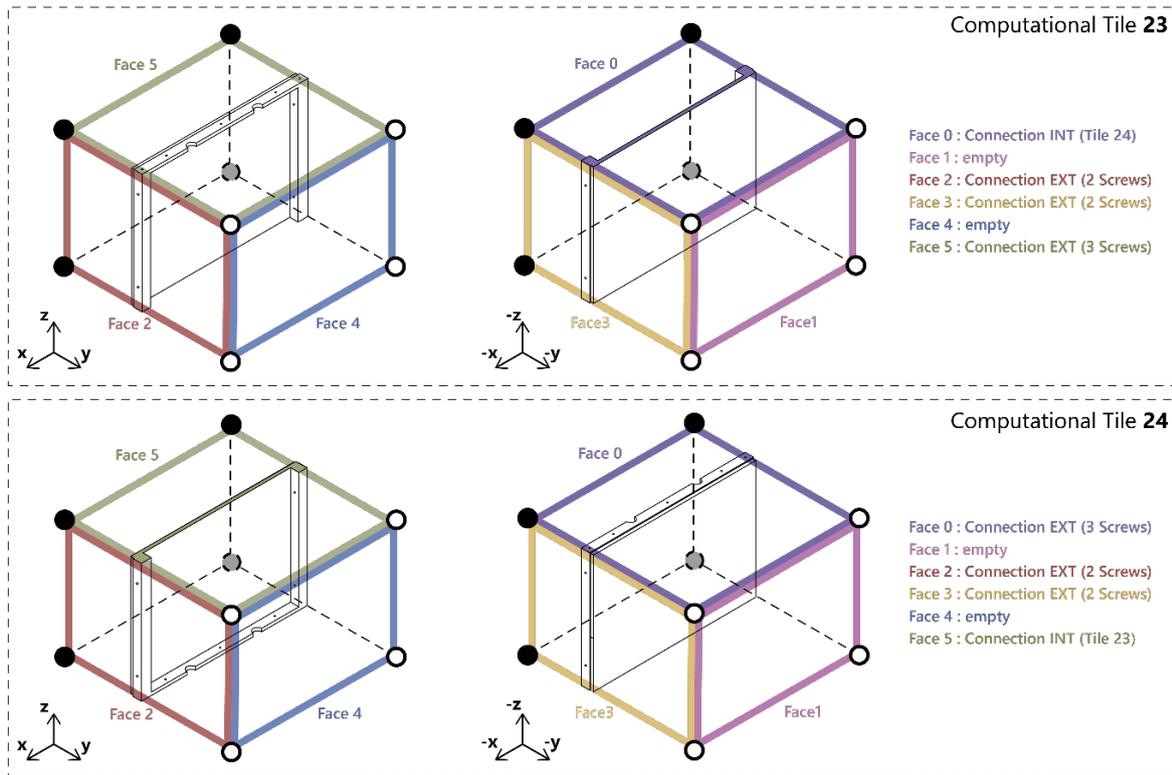


Figure 64: Encoding Faces (Author)

4.3 Calculating Tile Options

Based on a topological design and assuming a filled Tile Database, it is necessary to calculate the possible tile options for each position in the building. Following the concept of open building, the three systems structure, shell/membrane, and interior infill are separated, while the structural system is not in the scope of this elaboration. From the topological design, the building envelope and each individual room need to be extracted. Following that, for all individual instances, tile options are generated.

4.3.0 Generate Subsets

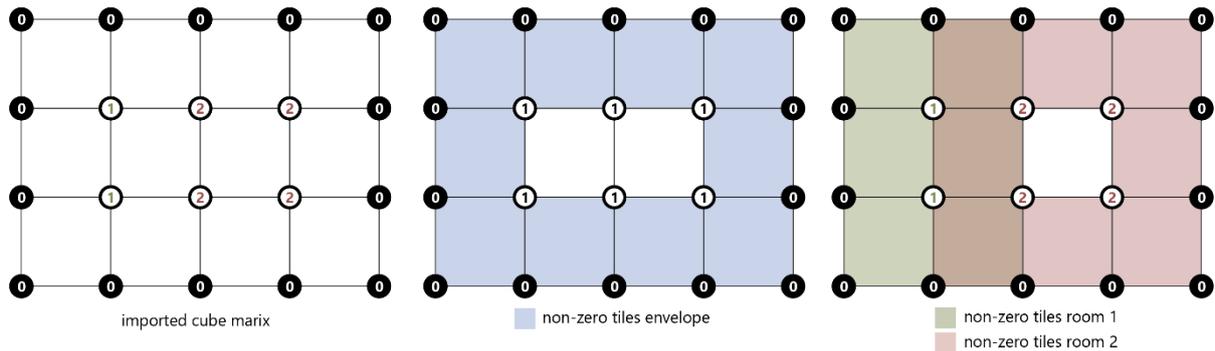


Figure 65: Converting an array of Cubes to Cube Subsets (Author)

Algorithm 2: Generate_Subsets

```

Data: list_of_codified_cubes, array_size
for item in codified_cubes do:
    remove delimiter "X" and room number
    if item does not contain "0":
        set item to "00000000"
    append item to envelope_cubes
identify number of rooms from list_of_codified_cubes
for room in all_rooms do:
    for item in codified_cubes do:
        for vertex in item do:
            if item does not contain "room":
                set vertex to "0"
        assemble vertex to item
        if item does not contain "0":
            set item to "00000000"
        append item to room_cubes
    append room_cubes to all_room_cubes
end
Result: (envelope_cubes, all_room_cubes)

```

Figure 66: Pseudo Code for creating cube subsets (Author)

4.3.1 Assigning Tiles to Subsets

When the cube subsets are created, Computational Tiles from the Database can be assigned to each item in all subsets. Firstly, the unique cube of all items in the subsets is identified, as seen in the pseudo-code below. In a second step, all Computational Tiles based on the same unique cube are appended to a list of possible tiles for each item. The Face Ids, stored also in the Tile Database, are appended according to the rotation the tile is placed. This algorithm creates the basis for a successful tile placement. It is also a part of the code that can be extended in prospect, as there is huge potential to add sensitivity to the selection of tiles though incorporating for example the performance of the tiles. Below, the algorithm is described in pseudo-code, an example of the output can be found in appendix 2.

Algorithm 3: Assigning_Tiles_to_Subsets

```

Data cube_subset, unique_cubes_table, computational_tiles_table
for item in envelope_cubes do:
    produce rotated and mirrored cube versions
    find matching entry in unique_cubes_table
    append UniqueCube_ID of entry to unique_cubes_subset
end

```

```

for item in unique_cubes_subset:
    find matching entry in computational_tiles_table
    for each matching entry:
        if cube_subset == envelope_cubes:
            if ctile_Type == exterior:
                append ctile_ID ctile_Faces to ctiles_item
        if cube_subset == room_cubes:
            if ctile_Type == interior:
                append ctile_ID and ctile_Faces to ctiles_item
    end
    append ctiles_item to ctiles_subset
end
Result: (ctiles_subset)

```

Figure 67: Pseudo Code for the Algorithm to assign Tiles to Cube Subsets

4.4. Tile Placement

In the scope of this thesis, two tile placement methods were developed. The first method is an interactive tile placement in Grasshopper, while the second method is an automated placement method based on the Wave Function Collapse Algorithm in python. While the first method could be implemented successfully, the second method is described in the section of future developments, as no results could be achieved from this method yet.

The interactive tile placement demonstrates one possible user integration in the design process. A user with a basic knowledge of Rhino and without technical knowledge of building components can individualize and modify a design according to personal preferences. As a first step, a geometrical representation of the building is created that the user can modify in a second step. For that, the geometry of the most basic set of Tiles from the tile options generated in the previous step, is applied on the topological design, as seen in the pseudo-code below.

The result is an array from tile geometries encapsulating the previously defined topological design. In the scope of this thesis, the geometries are represented through the obj and 3dm files produced in the tile creation section. In prospect of usability and performance, the geometrical information should be transported through an ifc file.

Algorithm 4: Compute_Basic_Placement

```

Data c_tiles_subset, array_size, voxel_size, c_tile_geometries
create 3D_array of points with array_size and voxel_size as cell as 3D_point_list
read names of all geometries as list_names_geometries
for item in c_tiles_array do:
    extract first tile in item as start_tile
    get rotation value from start tile as rotation_tile
    find start_tile in list_names_geometries
    get geometry related to start_tile
    if rotation_tile >= 4 do:
        rotation_tile = rotation_tile - 4
        mirror geometry
    rotate geometry for rotation_tile * 90°
    place geometry at first location first_item in 3D_point_list
    remove first_item from 3D_point_list
end
Result: (placed_basic_tile_geometries)

```

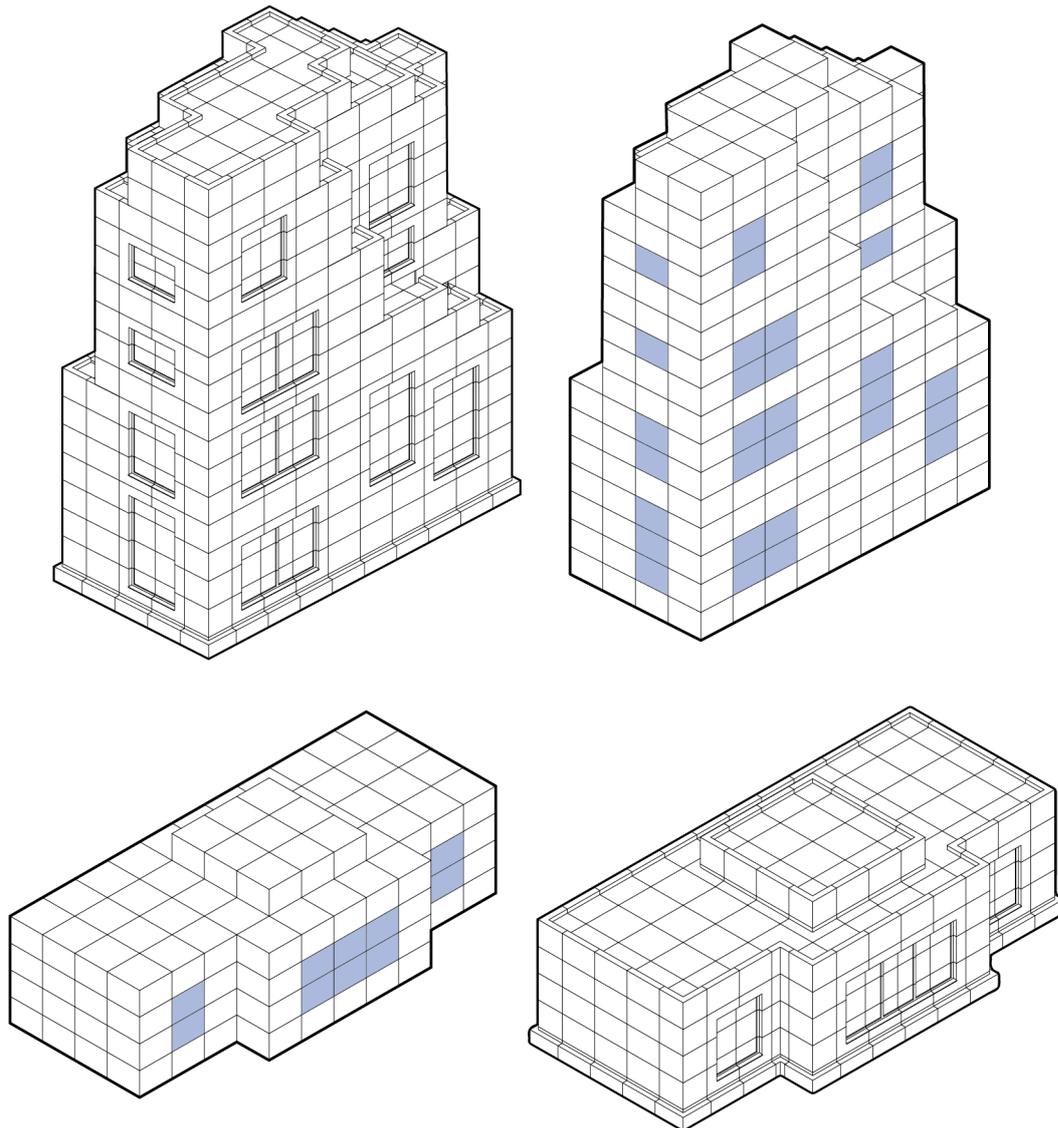


Figure 68: Several Building Assemblies as Algorithm Output (Author)

4.4.1 Interaction

Based on the initial placement of geometries, the user can now modify the arrangement. Conceptually, the user can replace and modify tiles in a gamified manner, offering an intuitive user interface to guide the user to advantageous decisions. For a proof of concept, the interactivity is implemented in the Grasshopper environment. Regarding an industry application, a gamified interaction would need to be programmed in a web-based 3D environment.

The implementation in Grasshopper is based on the activation of the tile geometries positioned in the previous step. The workflow is described in the user interaction diagram below, supported by screenshots of the workflow in the figure below. After the initialization and initial geometry placement, a clickable mesh is produced (Step 1). With the click on the left mouse button on one tile, options for replacements are shown (Step 2). The user can then navigate through the options, which are displayed simultaneously in the model (Step 3). When selecting the preferred option, the user can enter the alteration, and the model updates the modification. This step can be repeated for each tile as long as the user wants to

make changes. When the final version is achieved, the updated list of placed computational tiles including their rotations are saved and made available for postprocessing (Step 4).

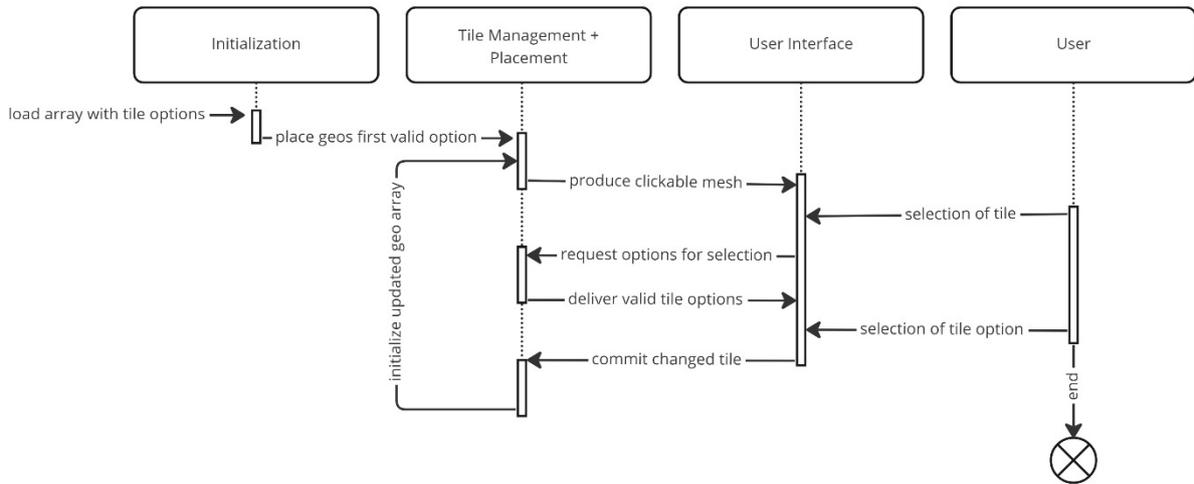


Figure 69: User Interaction Diagram (Author)

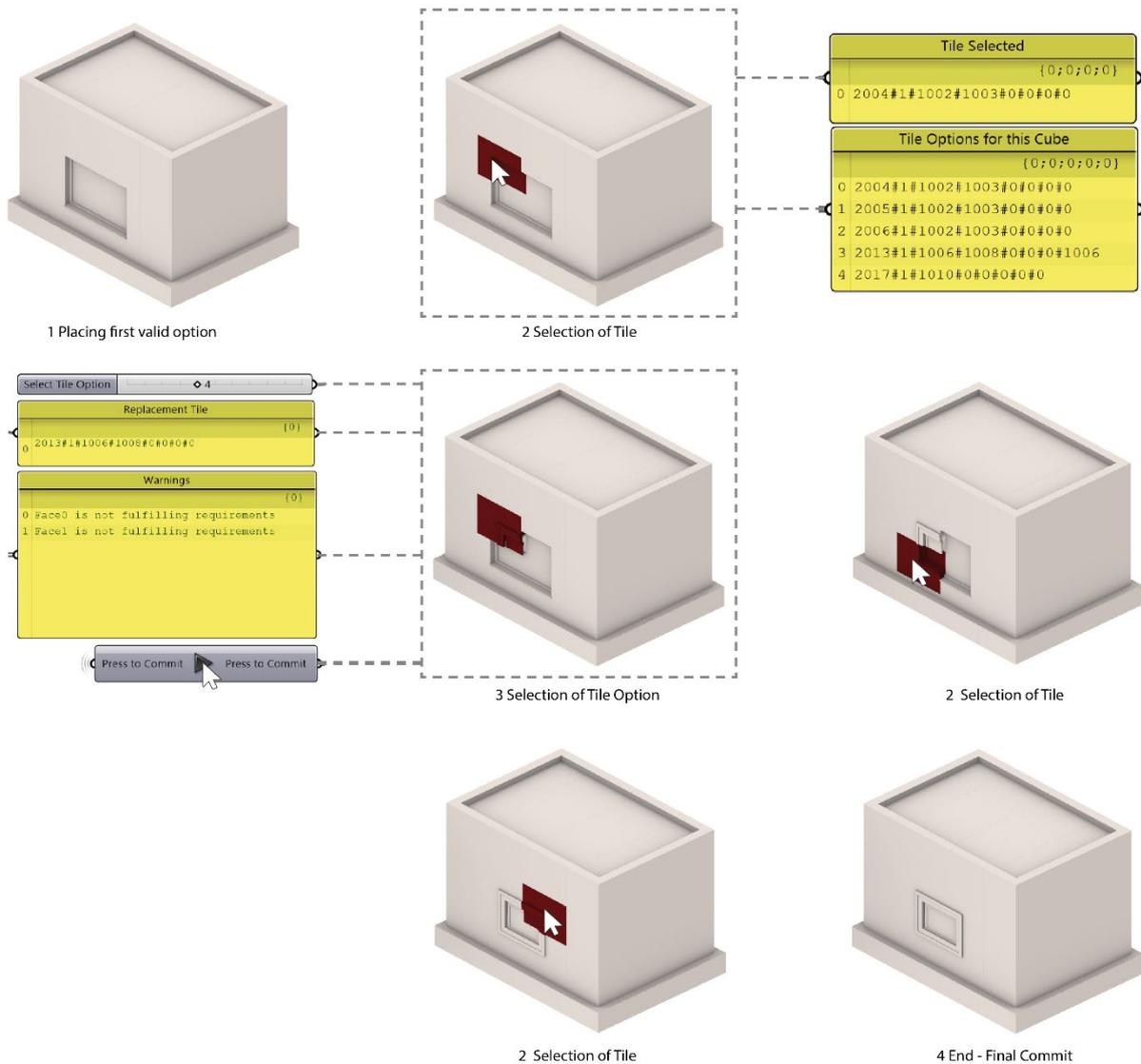


Figure 70: Interactive Tile Placement in Rhinoceros

4.4.2 Checking the Face Conditions

For the successful implementation of the user interface, the system's feedback regarding a tile placement decision is essential. Therefore, for each selected tile by the user, a stencil is applied to check the conditions of all six faces of the tile. In three-dimensional space, a stencil can represent neighbourhoods. The two common types of neighbourhoods are the von Neumann neighbourhood with six neighbours and the Moore neighbourhood with 26 neighbours (Azadi & Nourian, 2020). Since only the neighbourhood of the faces needs to be checked, the von Neumann stencil is sufficient. For each of the six neighbouring tiles, the Face_ID of the face adjacent to the examined tile is extracted. If the Face_ID is not identical to the adjacent one of the examined tile, a warning will show up in the Grasshopper Interface.

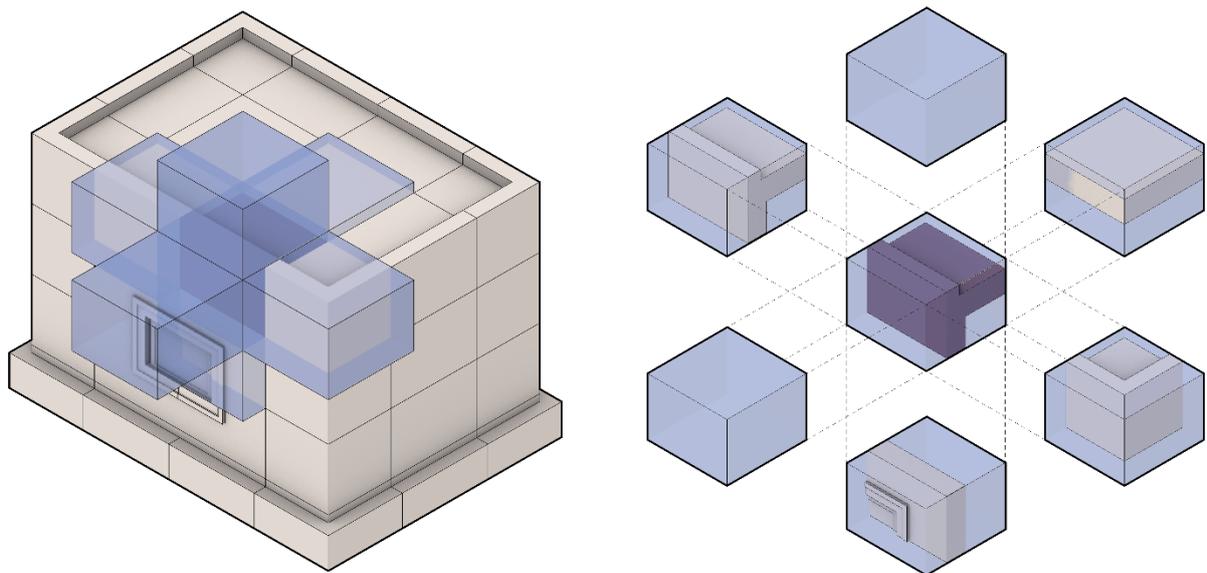


Figure 71: Van Neumann Stencil applied for the interactive tile placement (Author)

4.5 Postprocessing

The Postprocessing aims to translate the placement model (three-dimensional array of computational tiles) to information relevant to the building process, industry and construction. As described in the introduction to this chapter, multiple options of data export are considerable. The ifc format provides the opportunity to export geometrical data enriched with information about assembly, actors, parts and even more. Ideally, all information can be exported through this channel. To demonstrate the potential of the data export, two techniques were tested. First, there is the option of generating details of the important parts of the building geometry. In a second approach, the building tiles can be exported in the IFC format. Due to time reasons, the implementation of the ifc export does not contain geometrical information.

4.5.0 Detail Generation

To be able to use the algorithm for detail generation, computational tiles used in the model need to have a 2D drawing assigned, containing the detail for the specific geometric situation of the computational tile. Only details for situations that are relevant for planners, as corners, windows and other transitional situations need to be drawn as details. For example, no detail is needed for a computational tile representing a straight wall. The detail drawings can be produced in any CAD program and stored with the same file name as the computational tile itself, but with the addition of the plane, the detail is

produced on. For example, if a sectional detail in the plane XY is produced for computational tile E_B_11001100, the file is stored as E_B_11001100_XY.dxf. With this workaround, the produced details can be exported later into the model. The user of the tool needs to select a plane and the specific layer (in voxel steps) in which the detail drawing shall be generated. Now, for every tile placed on the sectional layer, a detail drawing is placed if existent, as seen in the Pseudo-Code below.

Algorithm 5: Detail_Generation

```

Data specific_c_tile_ids_layer, plane_select, layer_array, detail_geos
for item in c_tile_ids_layer do:
    in detail_geos find unit with name == item and plane == plane_select
    rotate + mirror unit according to rotation of item
    move unit to position if item on layer_array
end

```

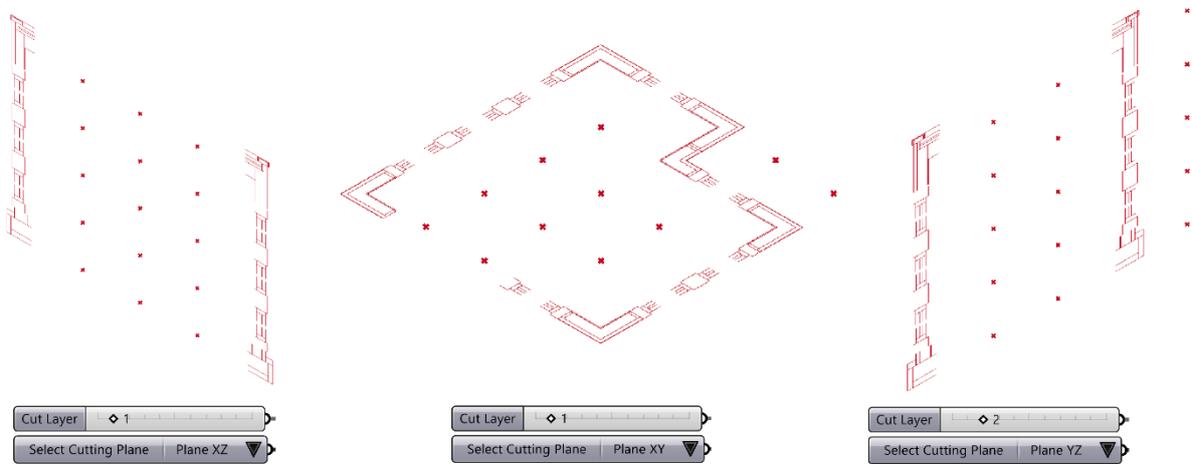


Figure 72: Detail Sections generated in Grasshopper (Author)

4.5.1 IFC Export

For the IFC-export the python library IfcOpenShell is used. IfcOpenShell provides an interface to read, create and modify IFC files in the programming languages C++ and Python (Norén & Gauthier, n.d.). After creating a new IFC file and project, each Building Tile used in the project is created as an IfcObject. Then, the included components and the tile quantities are created as IfcObjects and nested to the assigned Building Tile, as seen in the pseudo-code below.

Algorithm 6: IFC_Export

```

Data tile_database, used_building_tiles_ids
    create new IFC file
    create new IFC project
    for tile in used_building_tiles_ids do:
        import from tile_database all components assigned to item as components_list
        create new IfcObject with tile name
        create new IfcLabel with quantities of tile
        for each component in components_list do
            create new IfcObject with component name
            assign component to tile
        end
    end
Result: ifc_file_of_building_tiles

```

4.6 Design Strategies

With the methodical steps of the design tool defined in the previous chapters, conceptual considerations can be drawn that extend the scope of the infill system. Especially design strategies to solve structural joints, membranes, materiality, structures, and bespoke tiles need to be elaborated to extend the system to a whole building design.

MATERIALITY

The presented workflow allows the creation of a material passport for each building created with it. Through referencing to the database, the environmental impact and material composition can be derived. The building tiles, if created after the design for deconstruction principles, can be reused or recycled if the building is demounted. For the structure, it is recommended to use timber since it allows easy disassembly and reuse compared with concrete constructions.

STRUCTURAL JOINTS

When developing a structural system, tile developers can implement structural joints. To be compliant with the guidelines of design for deconstruction, it is essential to avoid permanent connectors, such as glue. It is preferred to use steel rods as introduced in the infill system, as seen in the figure below.

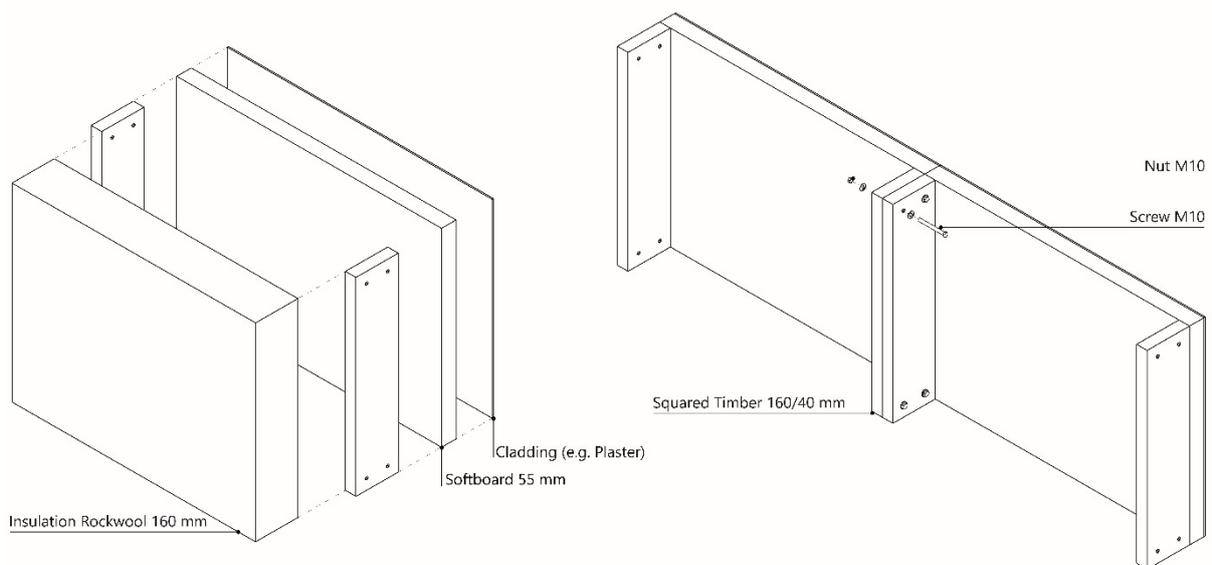


Figure 73: Structural Components (Author)

Membranes

Providing airtightness and water tightness is the weakness of a discrete building system. The discrete building systems of the WikiHouse and the Gablok system are excluding membranes and propose the installation on top of the assembled discrete structure. For the workflow proposed in this thesis, the same strategy is applied. Between the infill tiles and the envelope tiles, a vapor barrier can be installed. Additionally, a water-proof membrane can be mounted on the envelope tile before the application of the cladding.

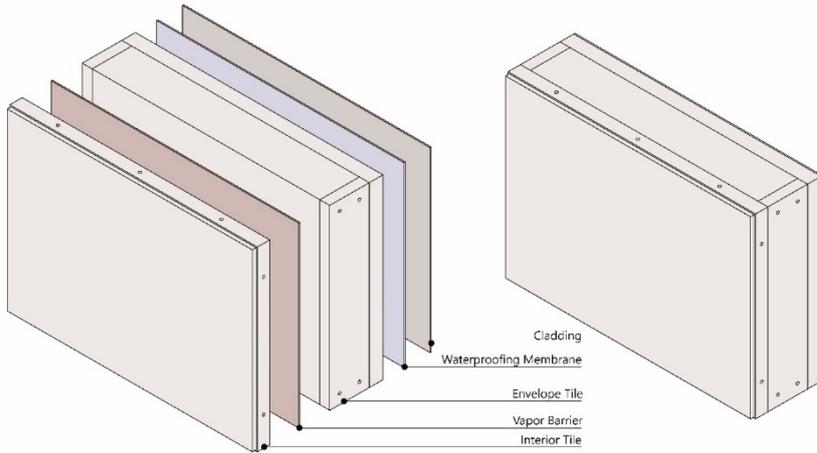


Figure 74: Integration of Membranes (Author)

CEILING

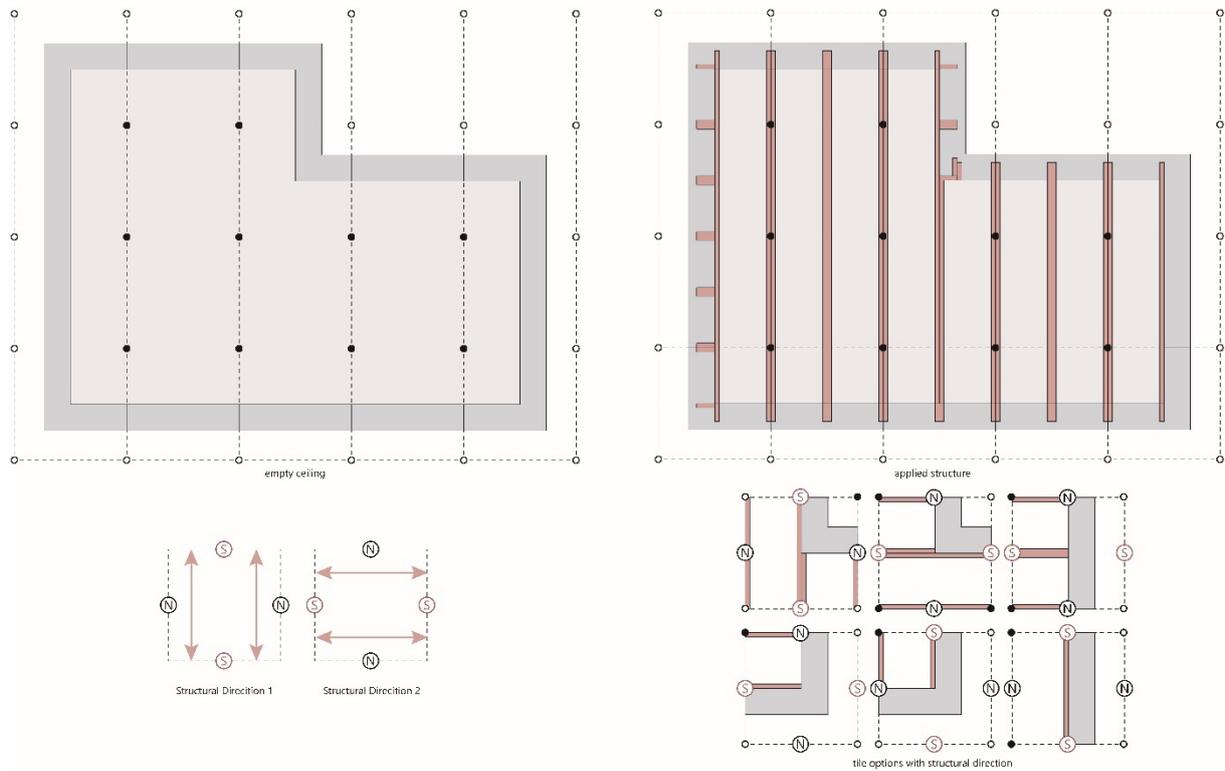


Figure 75: Defining a structural direction to produce ceilings (Author)

Regarding the structure, the ceiling is a considerable challenge. While the system can generate unidirectional structures without any issues, most discrete structural systems are directional. Therefore, it is necessary to introduce the parameter of the structural connection on the level of computational tiles. When a structural direction is assigned to all necessary ceiling tiles, it becomes easy to produce a structurally valid ceiling. The structural direction can be set to the x or the y- axis of the model, and the valid computational tiles can be placed according to that direction.

Bespoke Tiles

The design tool should offer the opportunity for application in projects of different sizes and spatial conditions. The most straightforward application that unfolds the system's full potential is a free-standing house without any spatial restrictions. On these projects, full modularity of the system can be provided, as tolerances can be balanced within the system. The system needs to adapt by using bespoke tiles when there are restrictions, such as a building gap for a terraced house. As seen in the figure below, a large part of the system can still be used, but around one-third of the tiles need to be modified. This can be done either on-site by disassembling the prefabricated tiles or in an industrial setting.

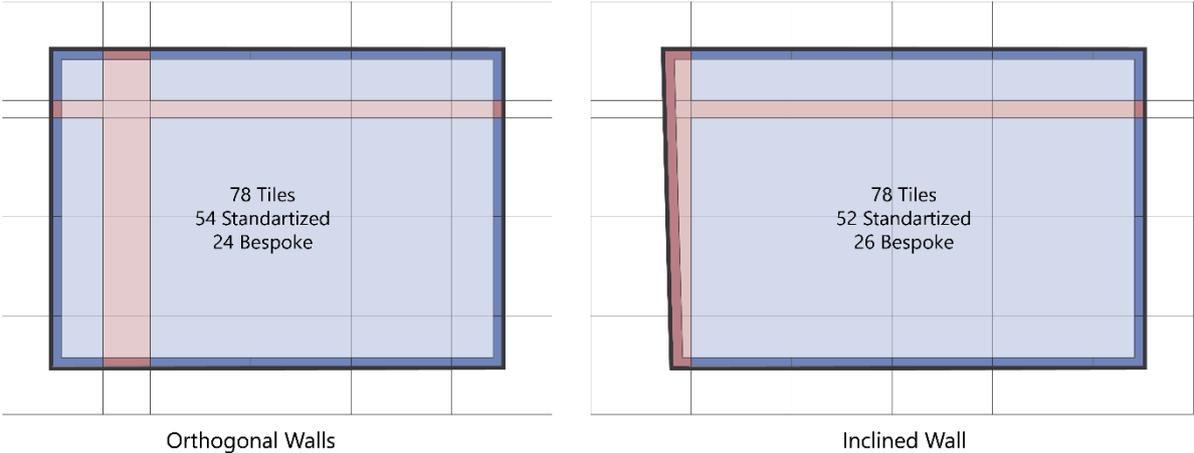


Figure 76: Application of the system on spatially restricted projects (Author)

4.7 Workflow Reflection

4.7.0 Performance

The workflow as presented in the chapters above is one possibility of applying the Design Methodology. Due to the technical abilities I acquired during this thesis's production, the applications are implemented in Microsoft Access, Rhinoceros, Grasshopper and VS Code with Python. Especially the geometry-based implementation of the interactive tile placement is not yet performing fluently. In Grasshopper, this effect was reduced through *data dam* components, but prospectively the workflow needs to be implemented in a different design environment. The data exchange between Python, Microsoft Access and Grasshopper is fluent. However, an integrated workflow developed only in Python and connected to an online SQL database would be a step for improvement.

4.7.1 Redundancy

Redundancy in computer sciences refers to the implementation of data backups during a workflow. Redundancy improves the reliability of a system and provides a better understanding of where the system fails. In the design tool, redundancy is integrated through the need to execute the tool steps individually. Therefore, it is possible for the user to detect unsatisfying results at the end of each step. However, in prospect it is necessary to implement the automatic detection of unsuccessful generated solutions, since the program is otherwise still dependent on a professional, who needs to validate the results of each step of the workflow.

4.7.2 Prospect

In prospect, the methods executed in this section need to be improved in reliability, performance and participation. Especially the Wave Function Collapse algorithm offers unused potential. By extending the design influence of this algorithm, it is possible to modify the spatial configuration of the design simultaneously with the tile placement. The algorithm could improve spatial decisions made in the topological design and guide the end-user through a participatory process based on performance- and design-based parameters.

5 Evaluation

5.0 Verification

To verify the design tool, a toy problem is selected to demonstrate all steps of the process, the usability and possibilities for participation and modification.

5.0.0 Building Task

The terraced house is the most common housing topology in the Netherlands, therefore this typology is selected to demonstrate the system. To rate the system's potential, the demonstration is designed to be comparable to the *beterBASISHuis* by tbi woonlab. As a building task, it is assumed that a small family plans to build the house, requiring a kitchen, a living room, two bathrooms and two sleeping rooms

beterBASISHuis

The *beterBASISHuis* is a product line for prefabricated rowhouses by TBI WOONlab. TBI is one of the largest Dutch construction companies, involved in the construction and technical equipment of various building projects and typologies. The *beterBASISHuis* focuses on affordability. An average terraced house is available for 120.000€, including building costs. Housing typologies, features and sizes can be individualized. The system allows the user to select a specific style, size, and height for the building design. Then, the user can select additional features, for example, a second toilet or dormer window. The house is constructed with prefab concrete panels (*De beterBASISHuis Woonplanner, 2022*).



Figure 77: *beterBASISHuis* (TBI Woonlab, 2022, 10 11)

5.0.1 Context



Figure 78: Site Context of the Toy Problem (Author)

An empty site in the centre of Delft is chosen to locate the toy problem. The site is embedded in the urban context of Delft and offers the possibility of extension, as one site on the property is left empty. The building to create is around 6 meters wide, 10 meters long and two stories high.

5.0.2 Topological Design

In the first step, a topological design of the desired building is to be configured. In collaboration with the architect, sizes for the several rooms and their position in the building are defined, as seen in the figure below. It is decided to position the sleeping rooms and one bathroom in the upper level, while remaining the other functions on the first floor. In a second step, internal connections and opening spaces of the building are assigned for each room.

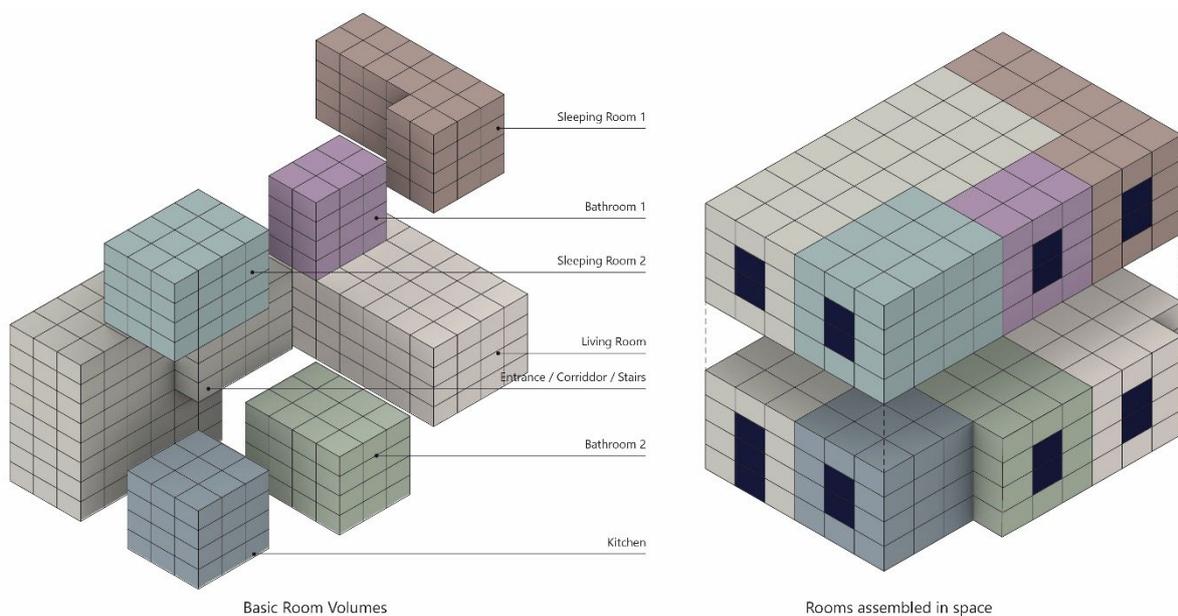


Figure 79: Topological design of a rowhouse

5.0.3 Tile Creation

After creating the topological design, the number of necessary tiles can be derived. There are at least two families required, one for the building envelope and one for the interior infill.

Envelope Family

The envelope family contains tiles the necessary tile for the generation of the foundation, walls and roof. Additionally, three options are created for window and door design. For the facade, there is a brickwork system and a timber system created, as seen in the following pages.

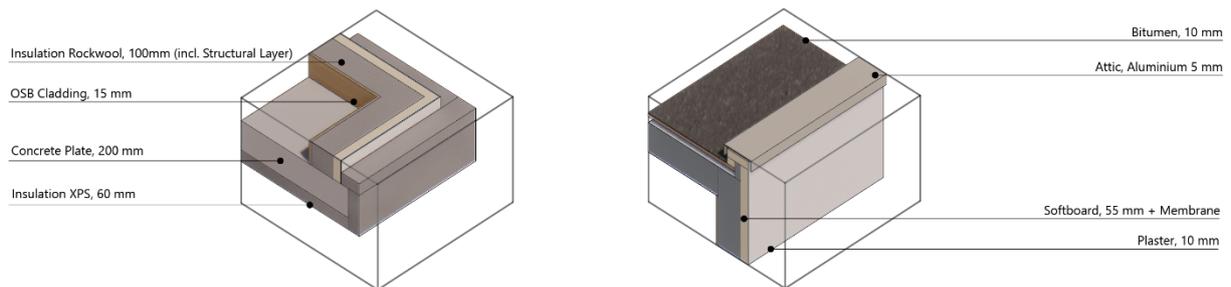


Figure 80: Computational Tiles from the Envelope Tileset (Author)

Interior Family

The interior family contains all necessary tiles to create walls, flooring, ceiling, interior doors, openings and stairs. As described in the limitations, a structural system is not part of this thesis. However, placeholders for structural components are considered. The interior family is based on the research described in the Development Section.

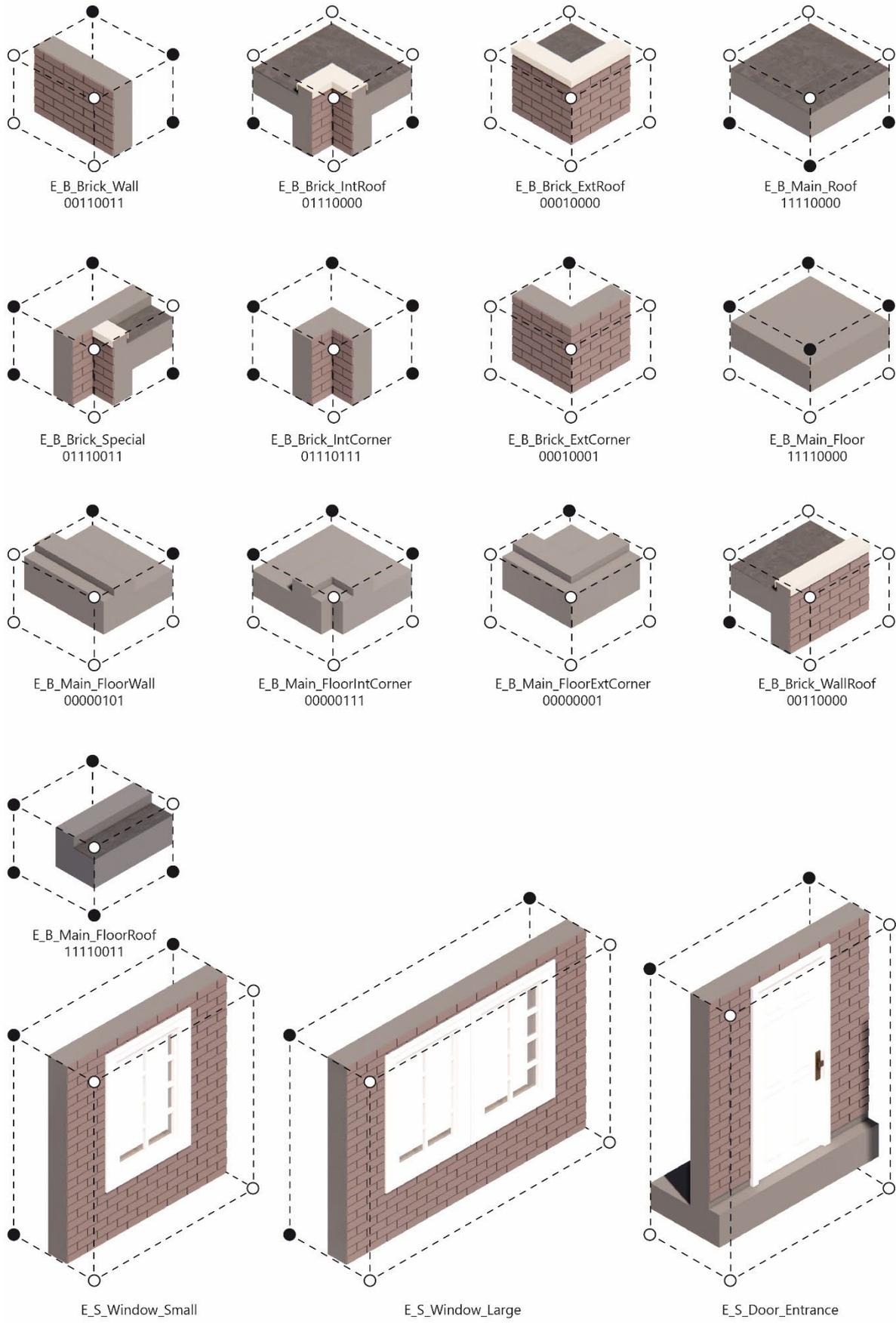


Figure 81: Envelope Tileset Brickwork

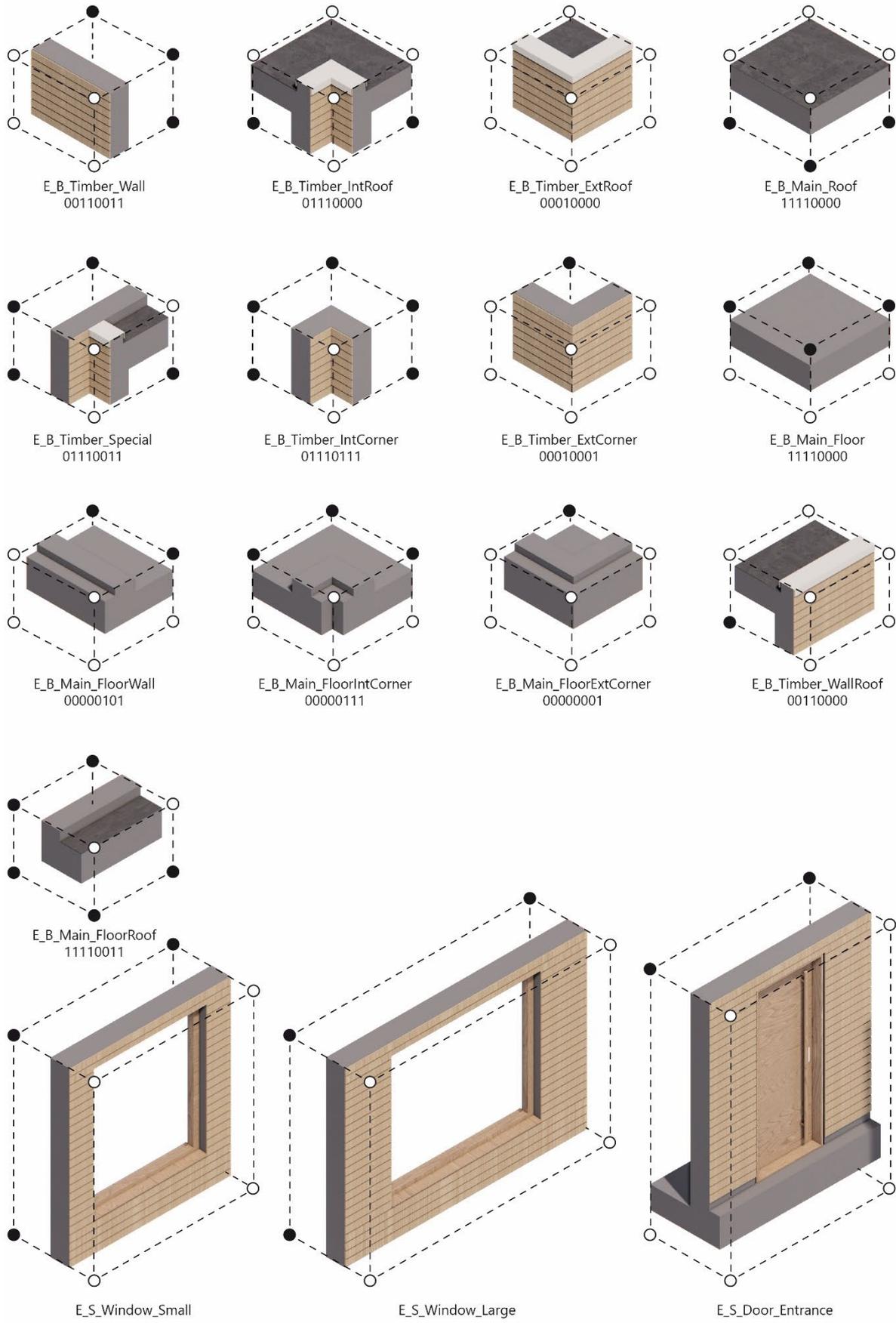


Figure 82: Envelope Tileset Timber (Author)

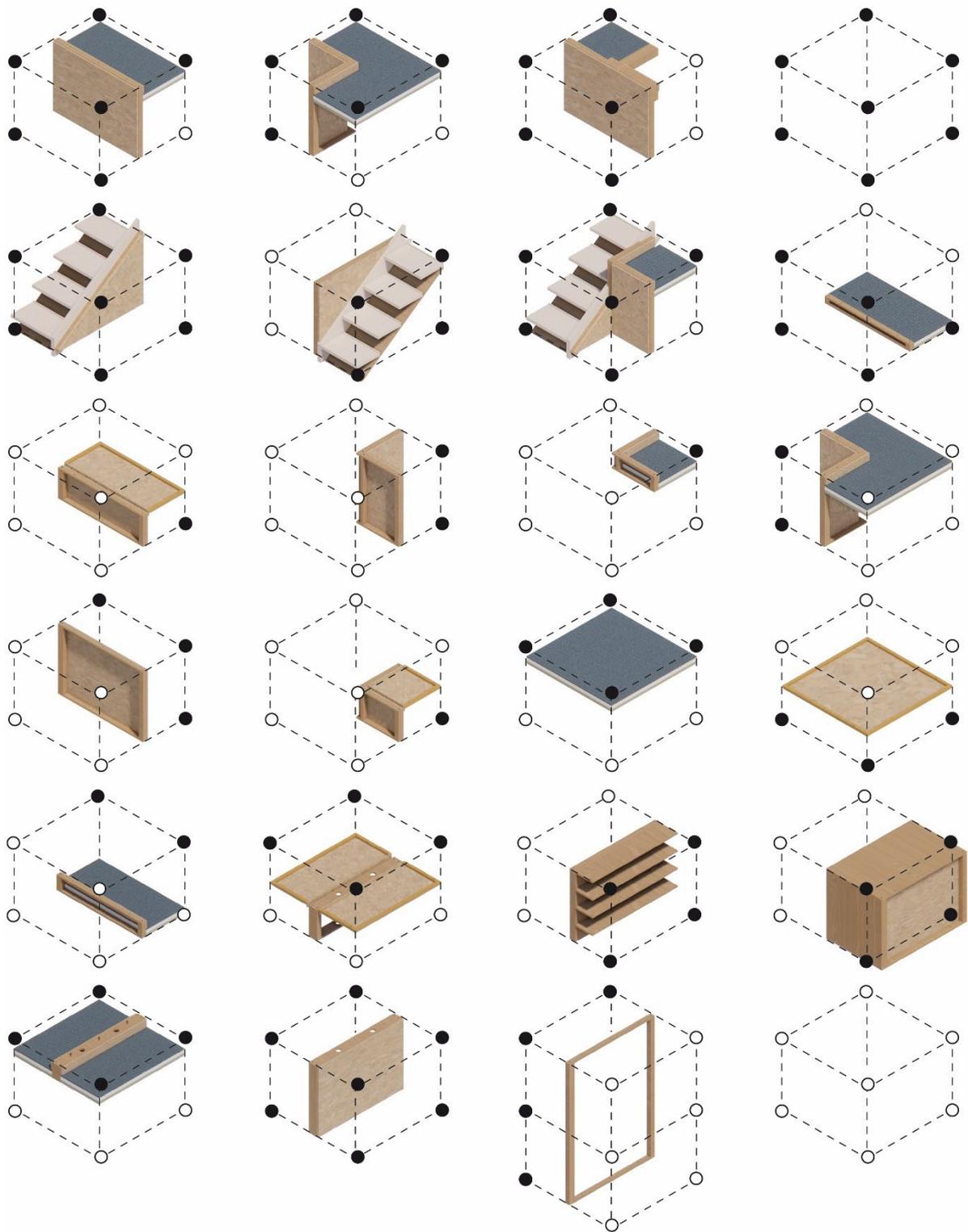


Figure 83: Infill Tileset (Author)

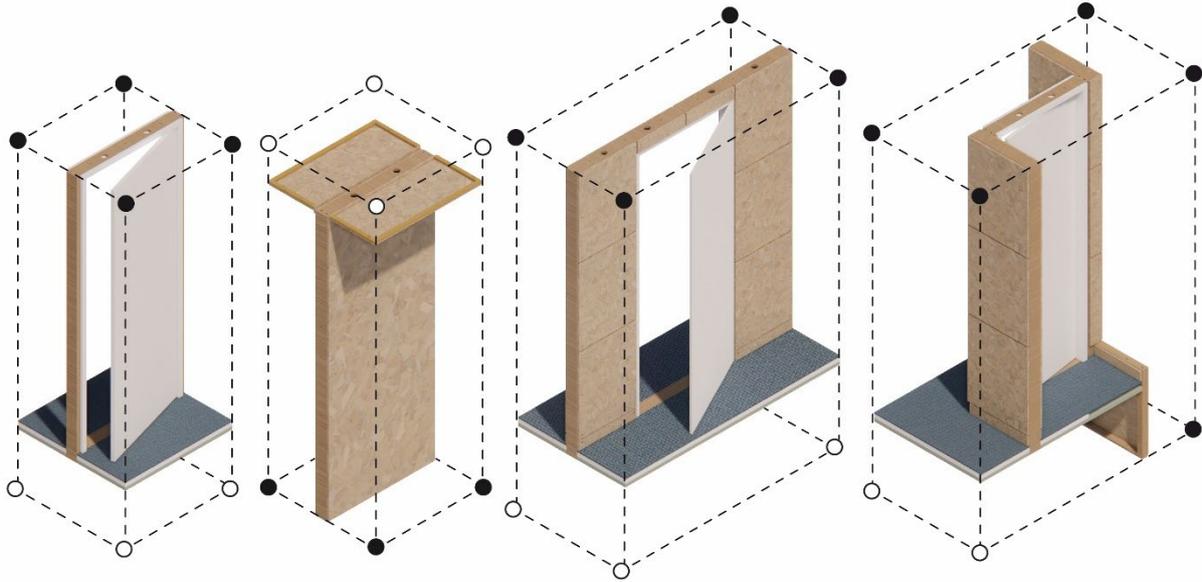


Figure 84: Additional Tiles Infill System (Author)

In the scope of the Interior Family development, the options of integrating several building products in the workflow are tested through the implementation of three different interior door types. Three door types with different sizes and frame types were selected for the implementation into the system. Especially, the compatibility of the system towards given products from the building industry is elaborated. The selected frames were a block frame, an enclosing steel frame and an enclosing wooden frame. These frames have different requirements towards connection and assembly. In the following figures, detail drawings of the implementation of the frames are presented. The simple wooden frame as basic module is very flexible regarding the implementation of new components, so all three door types were able to be implemented successfully.

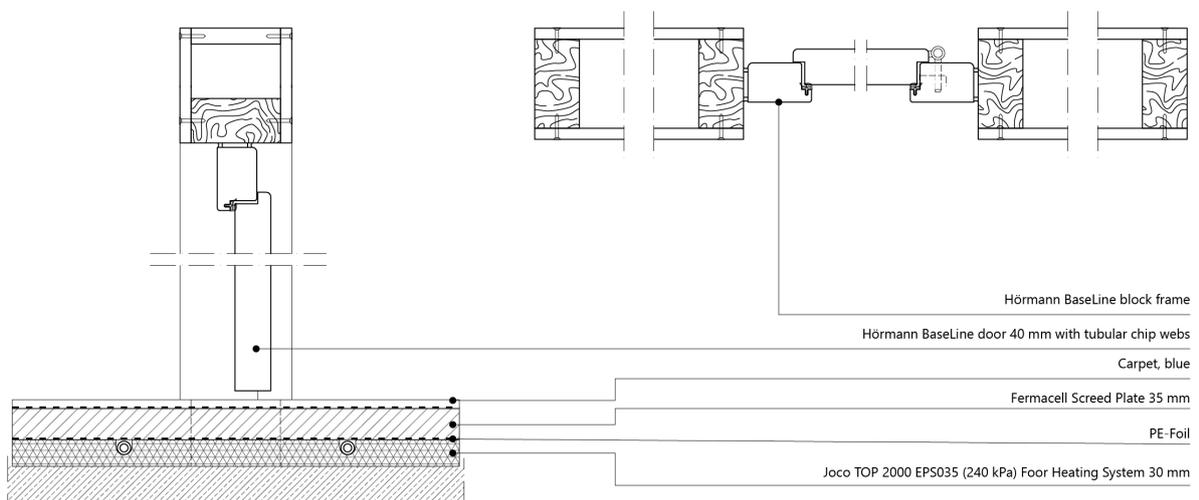


Figure 85: Detail Integration Hörmann BaseLine with block frame (Author)

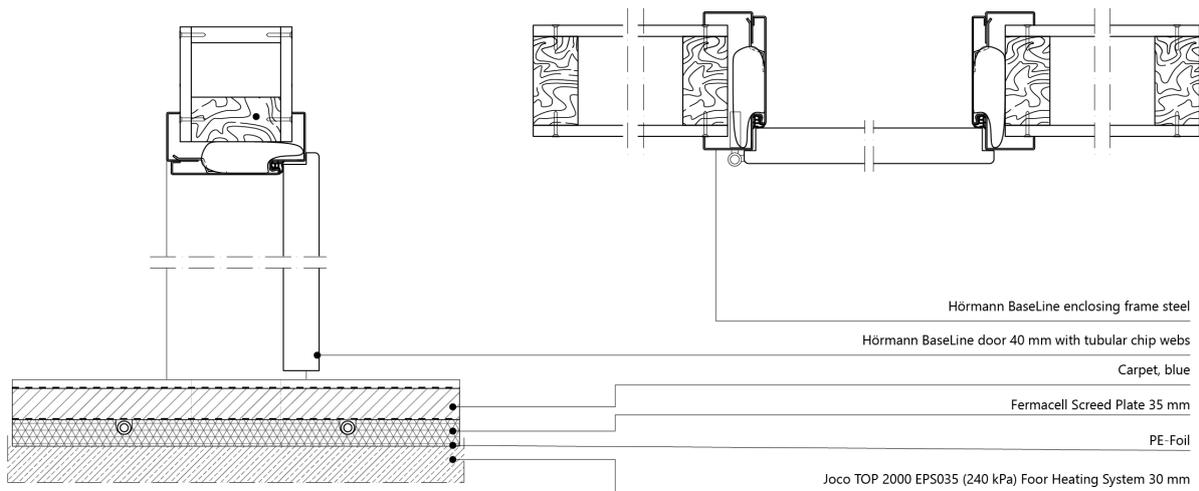


Figure 86: Detail Integration Hörmann BaseLine with enclosing steel frame (Author)

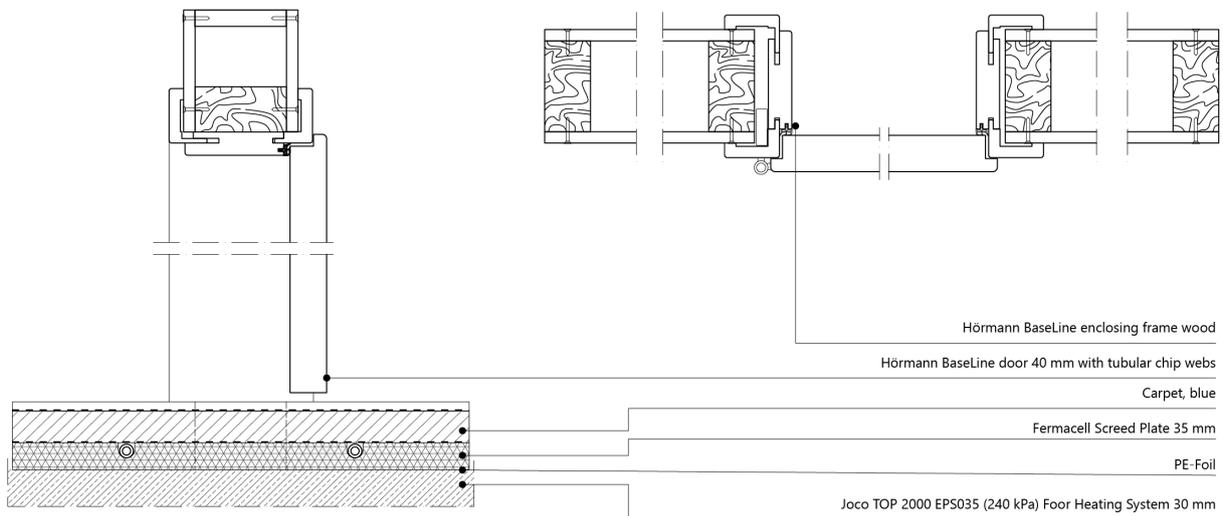


Figure 87: Detail Integration Hörmann BaseLine with enclosing wood frame (Author)

5.0.4 Interactive Tile Placement

The tile options for the cube array can be calculated based on the created tilesets and the topological design. Compatible tiles are selected from the database and imported into Grasshopper. The user interface for replacing tiles is initialized based on these tile options. Below, some examples of potential tile modifications are shown. Firstly, the facade is designed, and several options are examined. When the facade design is committed, each room can be designed individually. Finally, the whole configuration, containing the envelope and seven rooms can be committed and exported as a final design.

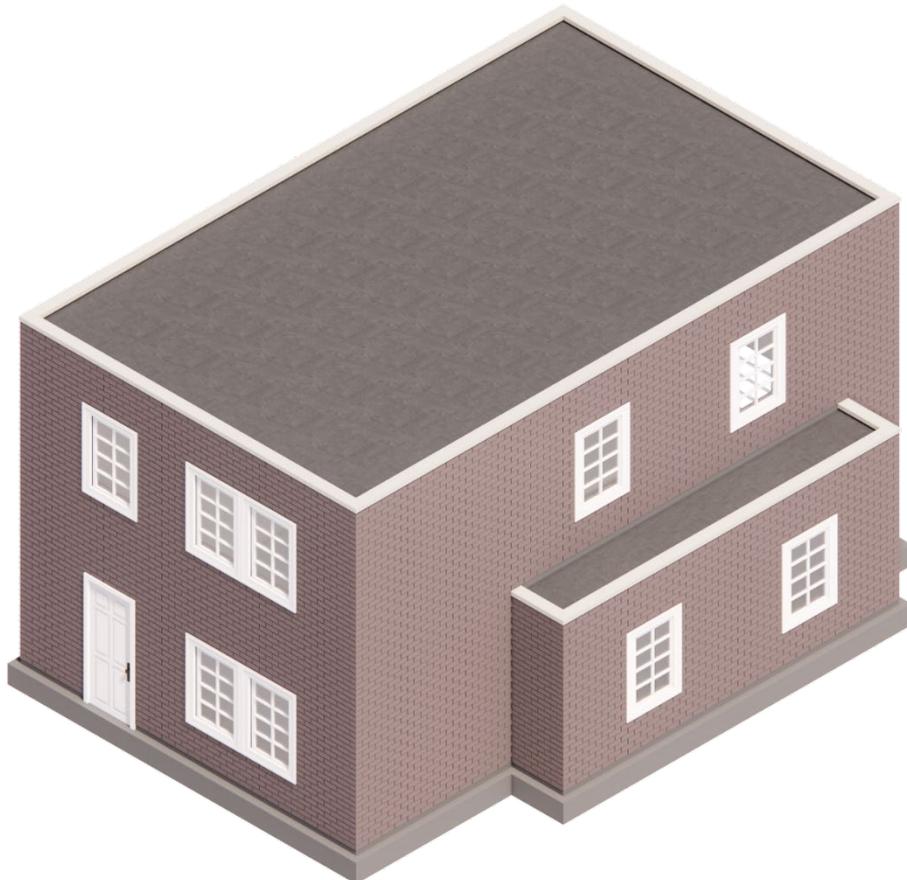


Figure 88: Envelope Variations (Author)

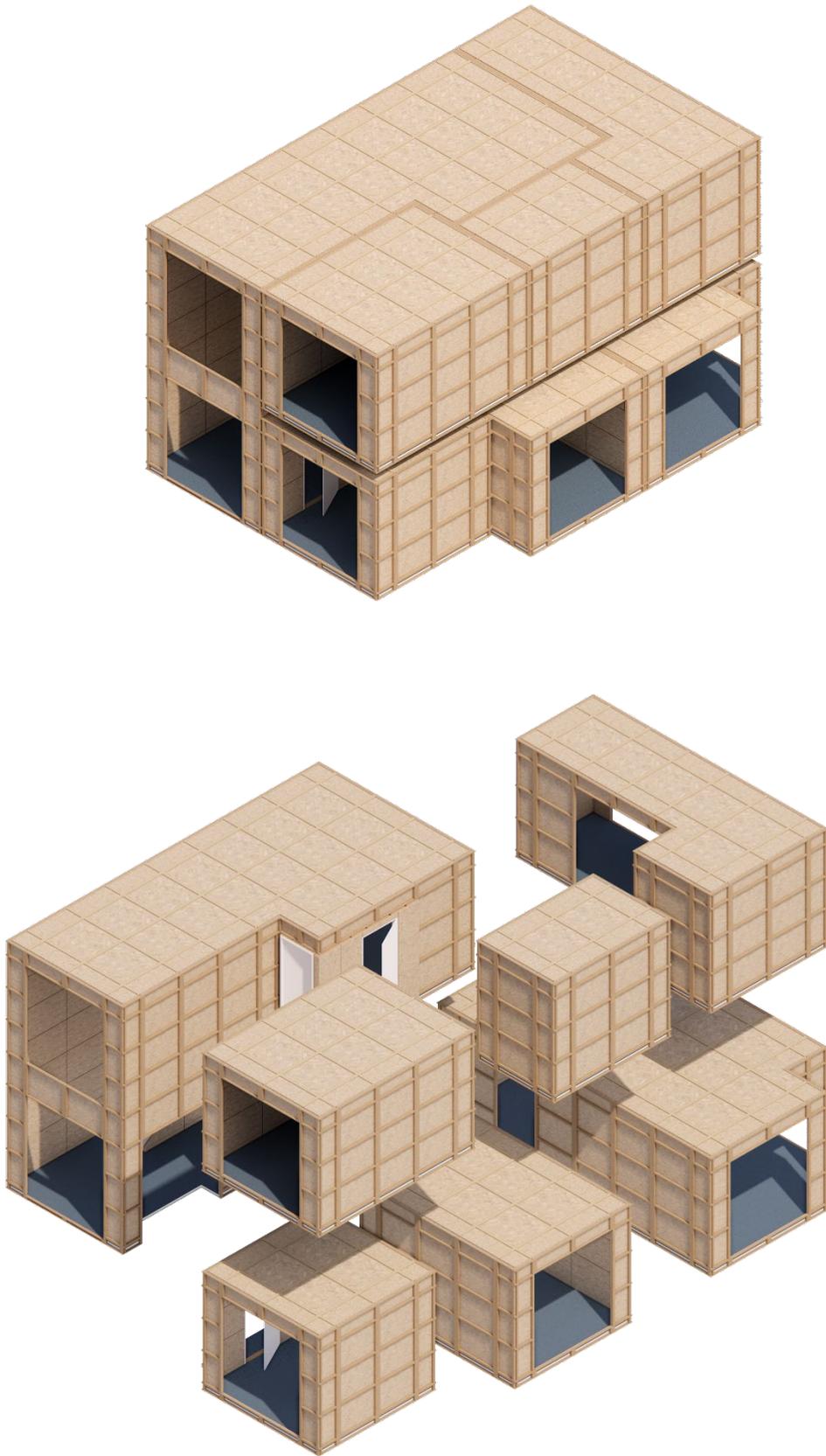


Figure 89: Applied Infill System (Author)

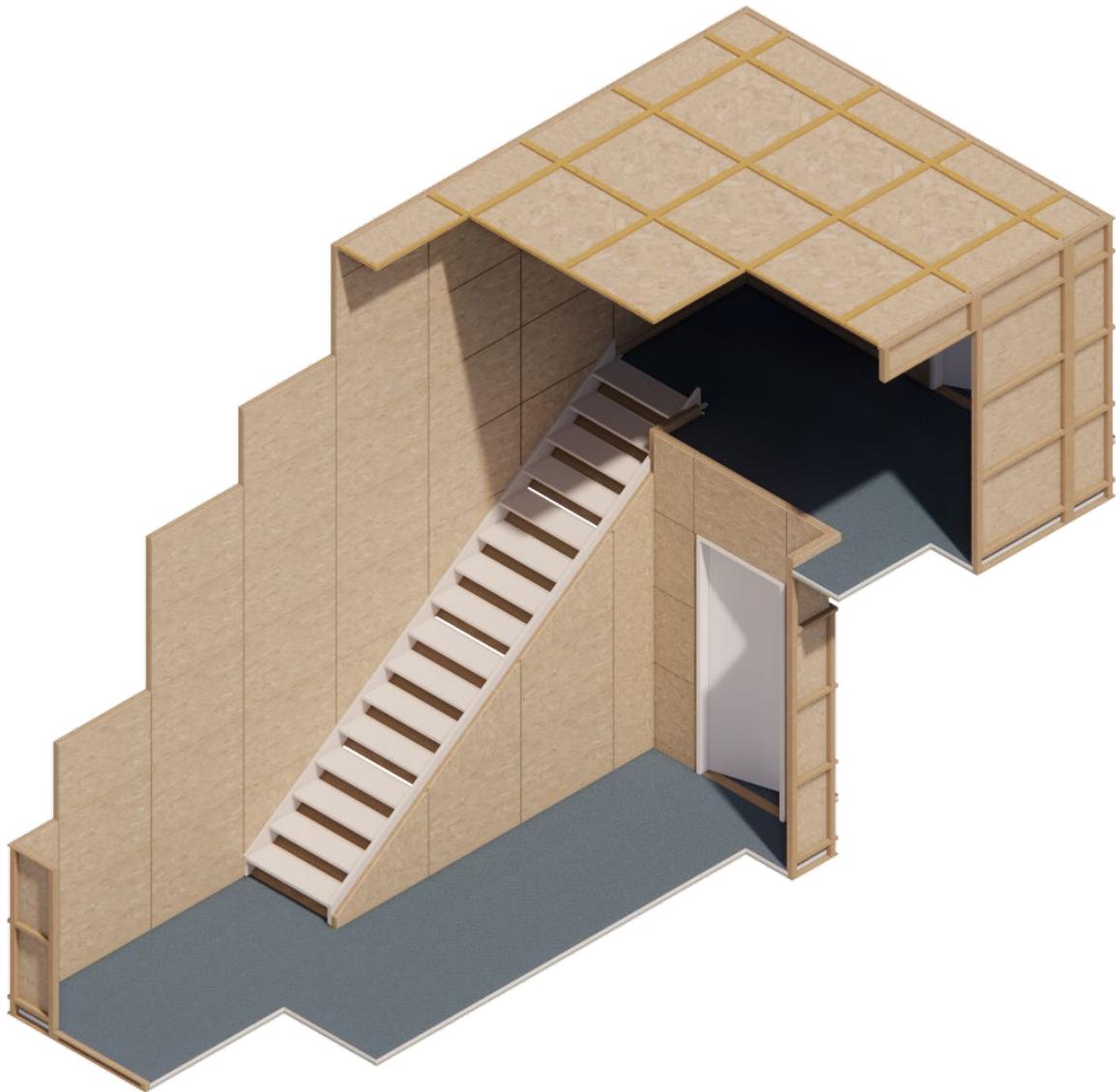


Figure 90: Detail Infill System (Author)

5.0.5 Postprocessing

When the final design is committed, the building design can be prepared for fabrication. For that, firstly the IDs and quantities of all used Building Tiles are extracted. The obtained information is the basis of producing the necessary tiles and based on the referencing of the design to the database, information about prices, materiality and assembly can be extracted.

5.0.6 Results

The application of the workflow to the toy problem resulted in the successful creation of a topological design, tile creation and tile placement towards a successful building design. The whole building design, neglecting structural components and sealing, was able to be produced with discrete components compatible with a topological design. A comparison between the design tool to the beterBASIShuis and an individualized design is shown below. The proposed solutions bridges the gap between a

prefabricated in-house design as the beterBASIShuis and an individualized construction. The system can combine the individualization of living spaces with the advantages of discretization and prefabrication. The proposal is classified in the context of the whole building industry in the following section.

System Name	Discrete BIM Design Tool	beterBASIShuis	Individualized Design by an Architect
Spatial Individualization	⚡ individual configuration and sizing of all rooms (based on grid)	✖ lenght, with and amount of floors adjustable, room configuration fixed	✔ individual configuration and sizing of all rooms (based on grid)
Style / Finishing Option	✔ style options according to tile implementations - theoretically unlimited	⚡ only six facade style options	✔ all styles / finishes possible
Industrialization Potential	✔ All Components can be prefabricated	✔ prefabrication in-house	✖ no industrialization potential
Workflow/ Product Monitoring	✔ Data control and construction management through integrated discrete BIM	✔ standartized construction workflow in-house	✖ need to create BIM model or manual monitoring

Table 1: Comparison of design methods (Author)

5.1 Validation

In the following, the features of the thesis product is set in the context of the construction industry. Therefore, the impact on the stakeholders, the application strategies, and the circular potential is elaborated.

5.1.0 Compatibility Potential

To test the compatibility of the developed design tool with building systems from the industry, the Skylark 250 building system from WikiHouse implemented into the workflow. The system was selected because the digital models and construction data are openly available, which is the main requirement for implementing building products into the developed workflow. Skylark 250 is the most common building system from WikiHouse, with an insulation thickness of 250mm. It is a demountable system built from several discrete components, as shown in the figure below. The components are produced from 18mm plywood sheets. A CNC machine is necessary to cut out the required shapes. The system is documented in an openly available guide, helping with the building's design, manufacturing, and assembly (Open Systems Lab, 2022).

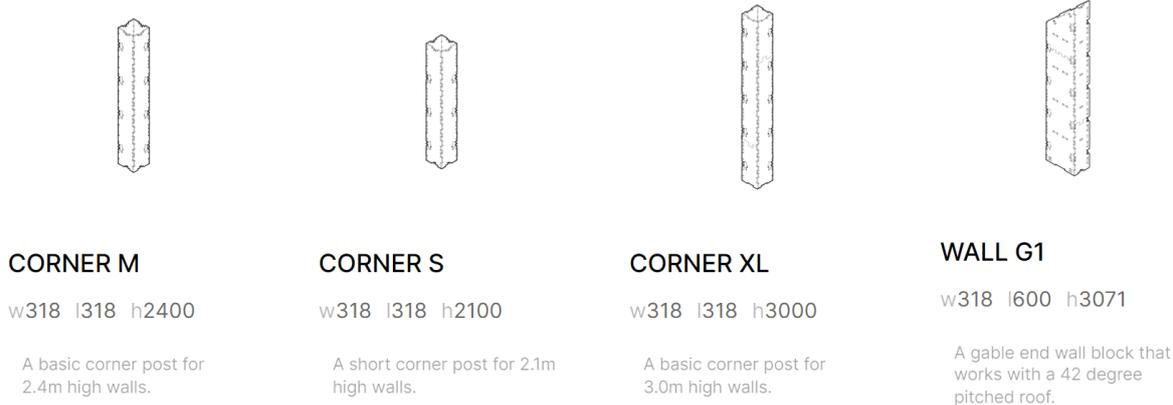


Figure 91: Skylark 250 Blocks (Open Systems Lab, 2022)

The first step of implementing a building system is the definition of voxel size. The Skylark 250 system is based on a 0.6m x 0.6m grid, with height increments of 0.3m. A voxel size as small as 0.6m x 0.6m x 0.3m requires relatively high computational resources. Therefore, the voxel size for this implementation is set to 0.6m x 0.6m x 0.6m.

In a second step, suitable building components for the implementation in the workflow are selected. Therefore, several voxel sets are designed, and suitable blocks are placed manually in the best identified positions.

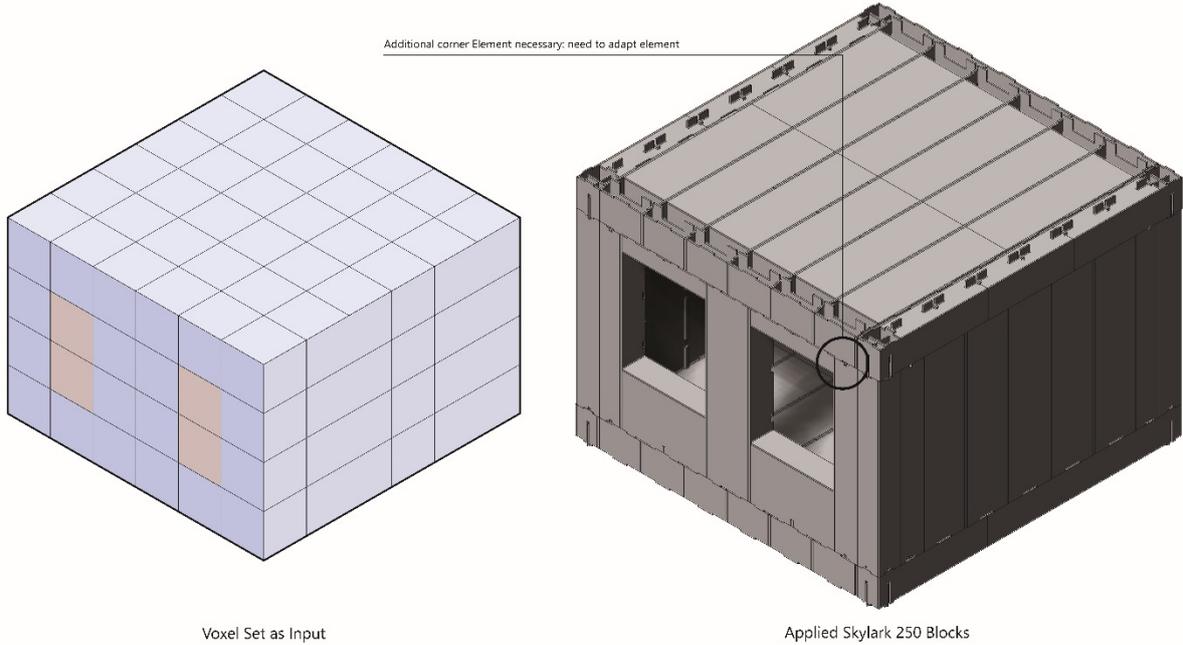
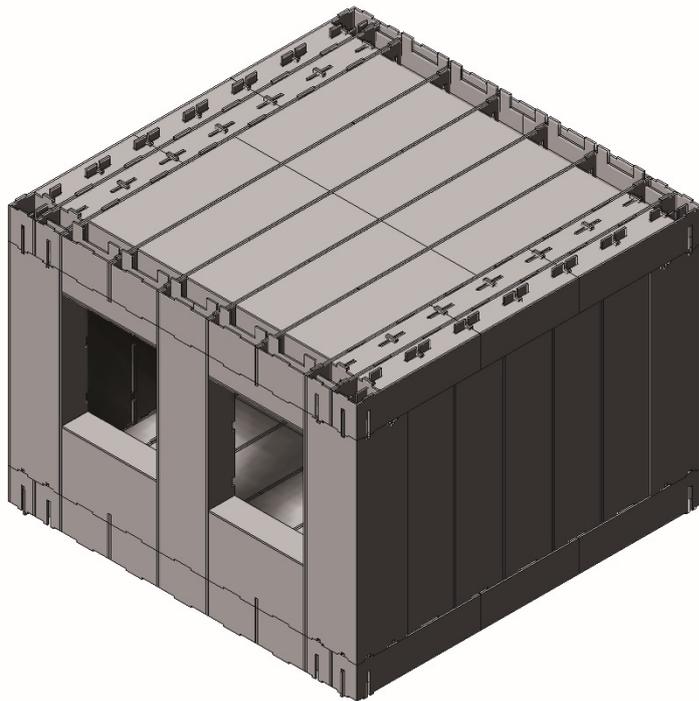
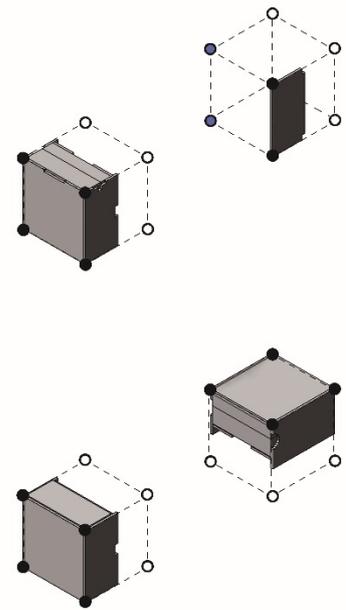


Figure 92: Application of the Skylark 250 system onto a voxel set (Author)

Resulting this experimental test of applying a topological design onto the Skylark system, missing components can be identified. As seen in the graphic above, the corner block cannot be implemented successfully. Therefore, the development of a new corner block is required.



Adjusted Skylark System



Example of produced Computational Tiles

Figure 93: Adjusted Blocks and Computational Tiles (Author)

In the figure above, the adjusted application with newly developed corner modules is shown. It can be concluded that a building system like the Wikihouse Skylark 250 can be integrated in the design tool workflow. With creating new blocks that simplify the integration in the workflow, the Wikihouse system can profit from the simplified design process. Compared with the current design workflow in Sketchup or Rhinoceros, the design tool developed in this thesis can contribute to a more accessible design for unskilled users.

5.1.1 Impact on the Stakeholders

Projecting the application of the workflow to the building industry, the influence on the addressed stakeholders as defined in the Design Methodology needs to be considered. In the table below, the advantages and disadvantages for the individual parties are displayed. The architect, as the main operator of the design tool, needs to take restrictions in the design freedom into account to profit from a simplified design process. This dilemma is further discussed in the general reflection.

	Architect	House Owner	Engineers / Consultants	Building Product Manufacturer
Participation	<ul style="list-style-type: none"> ✓ Topological Design ✓ Tile Placement 	<ul style="list-style-type: none"> ✓ Topological Design ✓ Tile Placement 	<ul style="list-style-type: none"> ✗ only read access 	<ul style="list-style-type: none"> ✓ Building Product Design
Added Value	<ul style="list-style-type: none"> ✓ Simplified / faster Design Process ✓ BIM Monitoring 	<ul style="list-style-type: none"> ✓ Information / Influence on cost, design and performance 	<ul style="list-style-type: none"> ✓ IFC data for all Tiles + Placement 	<ul style="list-style-type: none"> ✓ Simplified product integration
Disadvantages	<ul style="list-style-type: none"> ✗ Design restrictions to building grid 		<ul style="list-style-type: none"> ✗ Individualized / continuous structures difficult to integrate 	

Figure 94: The Design Tool in the context of stakeholders (Author)

5.1.2 Circularity Potential

Regarding the circularity potential of the system, three aspects need to be differentiated: the data aspect, the system aspect and the reconfiguration aspect. Firstly, through the information stored in the database and the integrated tracking of all used building products down to screws, the circularity potential can be estimated at all times. Wear and tear, same as damage, can be calculated based on spot checks, and the resulting rate can be projected to installed elements in similar conditions and circumstances. Regarding the system, the design following the guidelines for design for deconstruction, as described by Guy and Ciarimboli (2005) ensures a systematic circularity potential. However, in prospect, the system would need to be developed together with carpenters and experts on circular design to maximize the actual percentage of the system potential for reusability.

Regarding reconfiguration, the design tool setup allows house owners to modify their buildings after some time to personal preferences. Instead of removing components and throwing them away or downcycling, there is the potential to create a marketplace for exchanging building tiles for house owners. That way, building tiles can be reused in multiple projects. This approach would integrate into an upcoming trend in the construction industry for private households in the form of platforms for trading circular building materials as *restado* (*restado*, n.d.).

5.1.3 Application Potential for the Industry

To estimate the potentials and weaknesses of the proposed system regarding the application for the industry, two interviews with industry professionals were conducted.

The first interview partner was Nadia Remmerswaal. She was from 2020-2022 project leader at The New Makers, and leads her own architectural practice. As Remmerswaal states, the idea of the open system is well known at the New Makers, as it was one of the initial visions of the company. The ideas are implemented partially in current projects of The New Makers through the data exchange and modular combination of a system by The New Makers with a construction system of other companies.

Remmerswaal sees the potential of the proposed system regarding disassembly and material tracking. The voxel grid of the system would allow a simplified disassembly and reassembly of components, as the sizing and connections are very restricted. The integration of builders and constructors as designers of building components is seen critically by Remmerswaal. In the current industry, builders do not want to share their systems with the competition, as it would weaken their market position. Also, the business model of the builders would need to adapt. It would become necessary to introduce extensive storage facilities for components, as same as extended production facilities. However, the current economic pressure in the industry could lead to a change in the approach of some builders and manufacturers to evolve their perspective towards open building systems.

A second interview was conducted with Tim Castelijin, Head of Software and Data at The New Makers. Castelijin could evaluate the technical perspective of the proposed design tool. As an advantage of the voxel-based system, Castelijin identified the possibility of creating several Building Tiles from only a few Computational Tiles, simplifying detail generation. He generally appreciates the concept of an open system for multiple manufacturers. However, the need to follow predefined protocols could be less efficient than the tailored software solutions developed by The New Makers. The balance between standardization and flexibility for the individual needs of the product developers is one of the main challenges of a system like proposed in this thesis.

Concluding the outcome of the industry interviews, it can be seen that the concept of an open and modular construction system is known and partially applied. However, a large part of the industry is not yet open to participate in such a system, mainly because of intellectual property reasons, convenience and a lack of individualizable open software solutions. However, the industry is currently facing challenges through the environmental, political and economic situation and might slowly open up for a disruptive approach like a system as proposed in this elaboration.

6 Conclusions

6.0 Research Questions

In concluding this research, the research questions formulated in the introduction are answered and contextualized.

How to create a computational design tool that encodes building products into a participatory and discrete architectural workflow?

This thesis proposed a framework to break down the complexity of a discrete BIM workflow into functional units with defined inputs and outputs in which different stakeholders could operate in prospect. The workflow was divided into five stages: "Topological Design", "Tile Creation", "Calculating Tile Options", "Tile Placement", and "Postprocessing". Each of these stages can be solved with different design methods, of which some possible solutions are presented in this thesis.

A first unit is a design tool allowing the creation and codification of topological designs. This is done by creating design rules and creating a solution space in which successful designs can be produced. The second unit is a Database, which can translate building products into functional components that can be applied to a topological design. The third step is the determination of possible tile placements in relation to the topological design through constraint solving. The fourth step is the application of the components created in the database on the topological design. This is done with an interactive Grasshopper environment using an extended version of the Boolean Marching Cube algorithm. In the last step, data transfer of the design result to the building industry is enabled.

In the finalized workflow, an infill system for rowhouse configurations is implemented and applied successfully on a test case to validate the design tool.

This thesis demonstrated the successful creation and application of a design tool that allows the creation and configuration of spaces and components. While the tool can produce multiple solutions and run through several iterations, the stability of the tile placement algorithms is not ensured yet, as the implementation is done only for demonstration purposes.

How to implement several kinds of building components with various dimensions into a grid-based system?

The implementation of building products of all kinds of sizes is made possible by the distinction between a virtual computational module, "Computational Tile", and a building module which is sized as a manifold of the computational module. This workaround solves the application of all possible building products. However, it eventually requires partitioning one building product into virtual units at the beginning of the design process and, at the end of the process, reconstructing the building product from the virtual modules.

Systems available on the market, like the Skylark 250 system by Wikihouse, can be implemented into the workflow. However, the voxel size needed to adapt to this specific system, leading to the lack of compatibility with systems based on other voxel sizes. The workflow can simplify the access of several building systems but cannot overcome the problem of manufacturers using different grids for their product lines.

What are the participation possibilities in such a system for the different stakeholders in a building project?

The stakeholder structure was simplified in the scope of this thesis. However, conclusions regarding the participatory potential can be drawn for the most important stakeholders involved in the process, the

architect, the end-user, and the building product manufacturer. The architect needs to act as a mediator during the design process. She needs to guide the user towards a topological design and react on potential system errors. Furthermore, the integration of membranes and structure needs to be in the architect's scope. The end-user is involved in large parts of the design process and can make spatial decisions as same as aesthetic differentiations. Finally, the participation of the building product manufacturer in the design process is increased significantly, as he has the whole responsibility of the kind, amount, and style of components. It will be necessary to support the building manufacturer with a designer to evaluate the needs and requirements of architects and end-users for the system.

6.1 Method-specific Conclusions

Polygonization

The initial implementation of the Boolean Marching Cubes Algorithm was insufficient to translate a topological design, and to produce valid designs incorporating relations between and in building components. To translate a topological design, the binary codification of the Boolean Marching Cube algorithm was extended by two more voxel options, resulting in a heximal codification. The interrelation of Building Components was implemented through the addition of encoding the tile faces. With applying a stencil, neighbouring faces were able to be detected.

Discretization

The discretization of building components expressed in the "Tile Creation" offered the possibility of implementing any building product in a computational workflow. While the Database implementation is a success, the discretization requires the manual input of multiple sub steps. This process should be automatized in prospect.

Voxelization

Voxelization was the basis for creating a discrete workflow, as the restriction to a three-dimensional array of voxels simplified the design process. However, the voxel grid also showed disadvantages since it also restricted the design. Even though it is possible to create tiles without showing the voxel grid, it is still necessary for connecting to other components and restricts, therefore, the design freedom.

6.2 Discussion

One of the initial aims of this thesis is the contribution the one million homes initiative, meaning to provide qualitative housing in less time. To evaluate the contribution potential towards that, it needs to be assessed in which market segments the design tool could be applied.

Self-builders (detached or terraced housing)

Designing and constructing an individualized building is not very common in the Netherlands. However, the workflow proposed in this thesis is optimized for this use case. It is possible that more houses can be created with this method cause of a more efficient design process and prefabricated components.

Housing development (detached or terraced housing)

The terraced house is the most common building typology in the Netherlands. Because of the lack of building plots, in most cases are series of houses of the same building type constructed next to each other. This leads to monotonous architecture and a lack of individualization. With the design tool proposed in the thesis, project developers could offer their clients the opportunity to individualize their homes. The project developer can define rules and restrictions such as plot size, dimensions and possible tiles. In that solution space, clients could individualize houses to their needs.

Apartment buildings

Apartment buildings are often complex constructions with more than three stories. Since the proposed design tool's main aim is the use of discrete components, the possibilities of constructing large-scale buildings are limited. To incorporate apartment buildings, the workflow would need to be developed much further, eventually resulting in losing the initial simplicity. Regarding infill systems, there is indeed the potential to apply the workflow in apartment buildings to let the inhabitants configure their living spaces.

	 Self-Builder	 Development Projects	 Apartment Buildings
Initial Platform Set-Up	 Architect / Client 	 Architect / Investor 	 Architect / Investor 
Topological Design	 unrestricted 	 site-restrictions 	 only interior design 
Tileset Selection	end-user (+architect) 	end-user / investor (+architect)	investor (+architect) 

Figure 95: Application of the design tool to different markets (Author)

7 Reflection

The main objective of this research was to create a design tool for mass-customizing buildings with discrete building components. An increase in the use of discrete building elements is extremely relevant in the building industry, as it increases the circular potential of buildings through the reuse and reconfiguration of building components. The research produced a methodology for creating discrete building components, including their applicability to unlimited topological designs. Furthermore, the process is collaborative, and the degree of participation of professional, client and machine can be adjusted and individualized in every project.

7.0 Problem Solving Process

The first step in solving the design problem was to define a clear methodological framework, in which the scope and limitations of the systems were clarified. The problem was then split into more minor problems to have a clear solution space for each individual step. Experiments of several methods and processes elaborated a suitable method for each step of the overall problem. The final methods were examined in depth and executed to create sub-packages of an operational workflow. Finally, the potential of the design tool was evaluated through the application of the system to a toy problem, where the chances and weaknesses of the system were concluded.

7.1 Effectiveness of the Process

Computational Design in Architecture is a thriving research environment, as the developments are at an early stage, so experimentation in a broad context is necessary to achieve progress. Regarding this premise, I would say that the chosen process was indeed efficient, as it allowed me to tackle the objectives with several experiments. Furthermore, the experimental setup of the workflow allowed the intersection with other disciplines, such as Mathematics, Product Design, and Game Design to solve specific parts of the problem.

7.2 Mentor Feedback

My mentors were deeply involved in the process of ideation, design and development, which was a considerable contribution to the success of this thesis. Dr.Ir. Pirouz Nourian was during the largest part of my working time my first mentor, and helped me with the conceptual, methodical and mathematical part of the thesis. Ir. Shervin Azadi supported me with his broad knowledge on gamification and algorithmic design on the implementation of the design workflow. Their main feedback on structuring the thesis in working packages and starting to develop a working version of the design tool that can be extended in the process was a huge help in structuring my work and keeping an overview of the process. My first mentor Ir. Hans Hoogenboom supported me in combining the different ideas evolved in this thesis to an articulate storyline of the system. His main feedback focussed on the usability of the design tool for the practice and guided me to show the features in different contextual situations.

My second mentor Prof. Dr.-Ing. Tillmann Klein gave me important feedback towards the usability of the system in the building industry, and helped me to accept that there are no perfect solutions in the final industry application,

7.3 Relation of the thesis to the master track and the master program

Creating one million homes in the Netherlands until 2030 is a large challenge for the building industry and needs integration of a wide range of subjects from the Master of Architecture, Urbanism, and Building Sciences. In this interdisciplinary research, I am focussing on the possibilities in Building Technology towards efficient, affordable and individualized housing solutions. For me, Building

Technology is bridging the gap between architectural designs, principles and systems with the demands and possibilities of engineering, technology, and industry. With this thesis, I am tackling exactly this bridge, with bringing together requirements, concerns and limitations from multiple perspectives towards an integrated solution materialized in the design tool created as objective of this thesis.

7.4 Societal Relevance

During the work on my thesis, I learned that the construction industry is not by coincidence in the current state of slow innovation. Negotiating between different disciplines, departments, interests, programs and humans is a complex task. I finally had to accept, that a workflow incorporating all interests is impossible to tackle in one master's thesis. However, this thesis outlines a roadmap of possibilities for a collaborative workflow, that can finally result in a more participatory design process and a more effective housing production for using discrete components and their potential of reconfigurability and reusability.

7.5 Scientific Relevance

Building Information Modelling and especially the ifc-format provide already today a solution of collaboration between different stakeholders and interests. However, the introduction of the discrete component to the idea of BIM is novel and allows the customization of BIM to the needs of prefabrication. Based on the elaborated design methodology, further research can be done regarding the implementation of generative design methods and analysis lattices to the discrete design. The voxel-based system directly transfers building- and non-building related data structures on the design and can be located with spatial precision.

7.6 Ethical Issues

The largest ethical issue in discretization of architecture is the risk, that buildings are losing design quality through the need of reacting to grids and given components. This is expressed in the concern of an unitary grid, that will take the flexibility and uniqueness from spaces and creates an architecture of anonymity. During the thesis, I learned that an unitary grid for all buildings cannot be the solution for the problems in the building industry. An unitary grid would not be able to react to various building sites or in-situ projects without creating unpleasant transition components. Therefore, it is specifically included in the process, that the voxel size is not fixed, and can be adapted to be suitable for multiple context situations. Even if there are multiple building systems with different sizing existing next to each other, there can be still use of the methodologies elaborated in this thesis, by for example using the same components, parts and materials for multiple Building Tiles based on different voxel sizes. However, this is a question that cannot be finally answered with the product of this thesis. Another, more theoretical issue is the question if this thesis is contributing for the computer to replace the architect as the designer perspectively. However, one of the learnings I developed during the work on my thesis is that aesthetics and design are essential for qualitative living. The architect must be an essential part of the design process, but maybe the architect's role can change. She can act more as a mediator and designer who can focus on designing spaces and configurations and can leave the repetitive work of detail development and calculations to algorithms. I am looking positively to the architect's role in the future. Architects can be visionaries working together with computers instead of against them.

7.7 Self-development

During the work on my thesis, I got insights into the connection of BIM, design and industry. The concept of BIM, for me only known in the form of programs like Revit, is thanks to smart formats like the ifc already much more advanced. However, I hope I also could point out some weaknesses of the concept, especially in relation to design and industry.

Since most of the design workflow was built in the programming language python, I could improve my programming skills a lot. I saw the potential that is offered by the open-source concept of python libraries and was impressed by the power of the developer’s community, that were able to solve almost all technical questions I encountered.

7.8 Future Developments

7.8.0 Automatic Tile Placement

The automatic tile placement is an alternative to the gamified approach described in the design section. While the interactive approach is driven by the personal preferences of the user, the automatic approach aims for the optimization of specific design factors. This can be the price of the building, but also the performance or the adaption to the environmental context. The automatic placement is divided in two steps, as seen in the figure below. Firstly, the data, on which base an optimization should be performed, is selected initialized. In a second step, the Wave Function Collapse algorithm generates a valid solution.

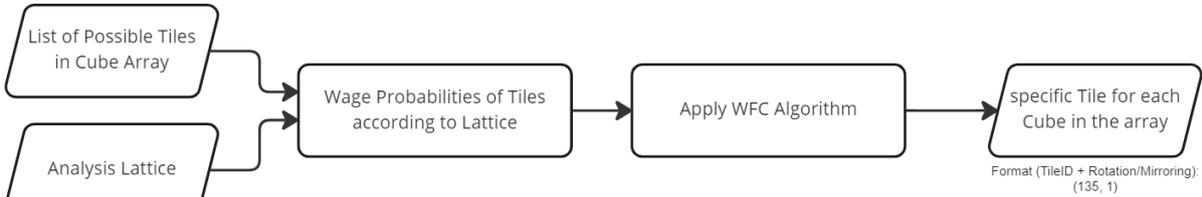


Figure 96: Automatic Tile Placement (Author)

Integrating the Algorithm in the Interactive Tile Placement would offer new possibilities to improve design quality. The algorithm could correct or extend the spatial configuration created in the topological design and present new spatial configuration options.

7.9.1 Human-Machine Interaction

Another development improvement would be the introduction of a more advanced human-machine interaction. This can be done by performing the topological design and interactive placement in a virtual reality environment. Components, tiles and spaces can be displayed in the scale 1:1, and end-users would get a better relation to the final executed design.

7.9.2 Digital Platform

The methodology elaborated in the thesis presents many possibilities for implementing solutions for the working packages. The major development that needs to be taken is the implementation of the workflow into a context that can actually be used by the construction industry. The best way would be to create a cloud-based platform in which the individual work packages can be implemented. An online platform would allow the architect, tile developer and end-user to work simultaneously on designs. They could exchange information about their needs and preferences and communicate problems in real time. In the figure below, a visual concept is presented for a potential design of the platform.

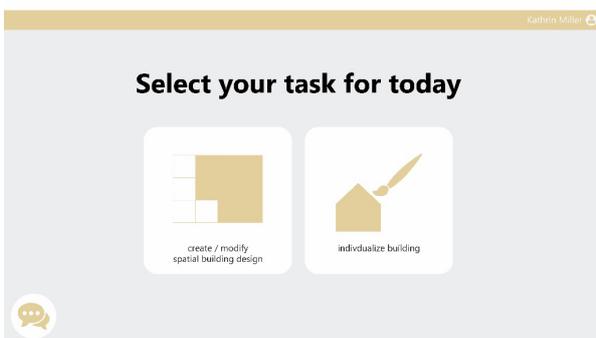
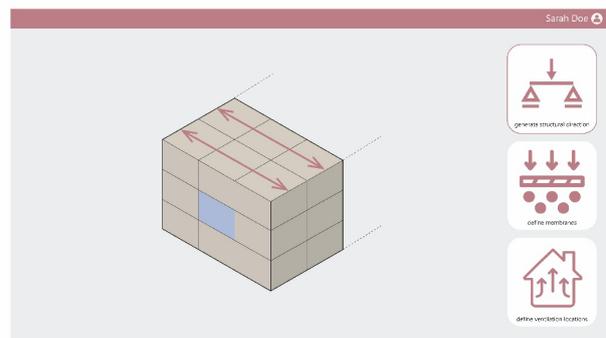
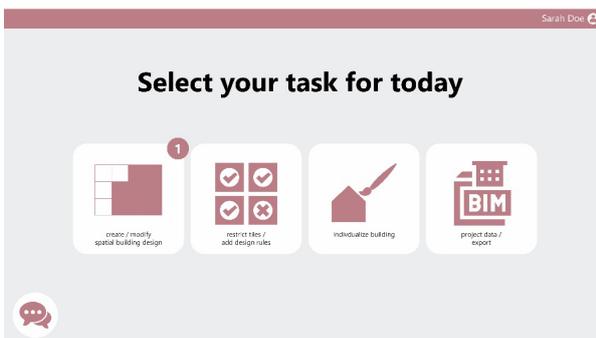
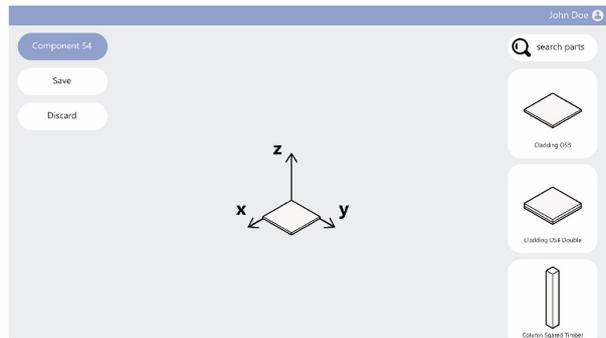
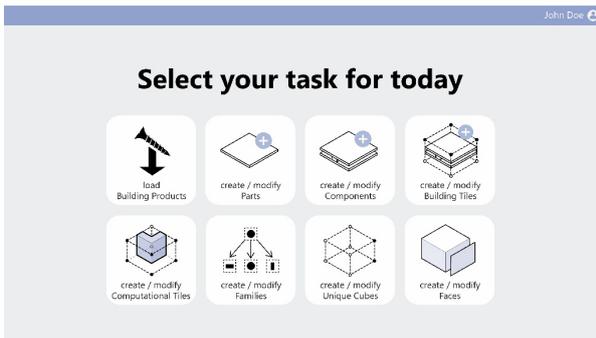
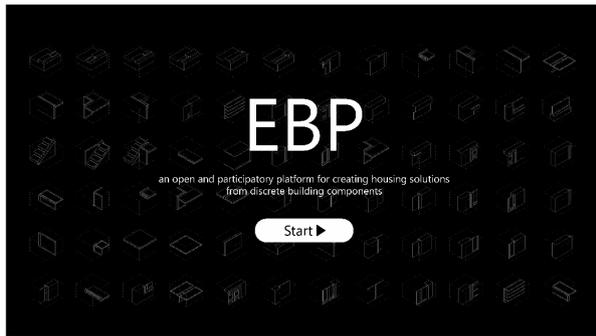


Figure 97: UX design Potential Platform (Author)

8.1 External Code Packages used for this Thesis

Numpy

Numpy is a python library that provides the opportunity to create, access and modify multi-dimensional arrays.

<https://numpy.org/doc/stable/index.html>

pyodbc

pyodbc is a library that allows the access of odbc databases. It is used in this project to read the Microsoft Access Tile Database.

<https://pypi.org/project/pyodbc/>

IfcOpenShell

IfcOpenShell provides tools to access, modify or create ifc files from scratch. It is used for the export of ifc files.

<https://ifcopenshell.org/>

Other

The framework for the two-dimensional implementation of the Wave Function Collapse algorithm is created by Ir. Shervin Azadi. The contributed code is contained separately in the file "WTC_Pillow" in the thesis repository on github.

9 Bibliography

9.0 Literature

Abramowitz, M. (2012). *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. (Vol. 1–1 online resource). Dover Publications. <https://www.hoopladigital.com/title/11605855>

Azadi, S., & Nourian, P. (2021). GoDesign: A modular generative design framework for mass-customization and optimization in architectural design. 39th eCAADe Conference 2021, Novi Sad.

Azadi, S., & Nourian, P. (2020). *'topoGenesis: Initial Pre-release'*. Retrieved 10. October 2022 from <https://topogenesis.readthedocs.io/>.

Berlo, L. van, Krijnen, T. F., Tauscher, H., Liebich, T., Kranenburg, A. van, & Paasiala, P. (2021). Future of the Industry Foundation Classes: Towards IFC 5. *Proceedings of the 38th International Conference of CIB W78*. <https://repository.tudelft.nl/islandora/object/uuid%3Abd3b0a6b-e956-4474-bddd-94b77aa7c6e0>

bimobject. (2022). *BIM objects- Dresser Door | BIMobject*. BIMobject®. Retrieved 14. November 2022 from <https://www.bimobject.com/en/murphydoor/product/dresser-door>

Block by Block. (2022). *Block by Block*. Block by Block. Retrieved 02. December 2022 from <https://www.blockbyblock.org>

Bock, T., & Linner, T. (2010). *Individualization of Design Variation in Construction*. 27th International Symposium on Automation and Robotics in Construction. <https://doi.org/10.22260/ISARC2010/0064>

Brand, S. (1994). *How buildings learn: What happens after they're built*. Viking.

Bugaj, M. (2020). Wave Function Collapse—Tutorial of a Basic Example Implementation in Python. *The Startup*. <https://medium.com/swlh/wave-function-collapse-tutorial-with-a-basic-exmple-implementation-in-python-152d83d5cdb1>

buildingSMART-International. (n.d.). BuildingSMART Data Dictionary. *BuildingSMART International*. Retrieved 22. November 2022 from <https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/>

buildingSMART-International. *Industry Foundation Classes (IFC)*. buildingSMART International. Retrieved 20.09.2022 from <https://technical.buildingsmart.org/standards/ifc>

Chasioti, E. (2020). *Gameplay with encoded architectural tilesets: A computational framework for building massing design using the Wave Function Collapse algorithm*.

Chien, S.-F., & Shih, S.-G. (2001). Design through Information Filtering. CAAD Futures 2001, Eindhoven.

Cuperus, Y. (2001). *An Introduction to Open Building*.

Daga, L. (2021, February 19). *A Practical Approach to Level of Detail (LOD)—United-BIM*. Retrieved 03. December 2022 from <https://www.united-bim.com/practical-approach-to-level-of-detail/>

Ekholm, A. (2005). ISO 12006-2 and ifc – prerequisites for coordination of standards for classification and interoperability. *ITcon*, 10, 276–279.

- Gablok (2022). Retrieved 23. January 2022 from <https://gablok.be/en/>
- Gan, S., Zhang, H. (2021). Research on Maintainability and Renewability of SI Housing. IOP Conference Series: Earth and Environmental Science
- Garber, R. (2014). *BIM Design : Realising the Creative Potential of Building Information Modelling* (Vol. 1). John Wiley & Sons, Incorporated.
- Giacosa, F. (2014). On Unitary Evolution and Collapse in Quantum Mechanics. DOI: 10.12743/quanta.v3i1.26
- Global Property Guide (2022). *The Netherlands: pandemic not enough to cool down its red-hot housing market*. Retrieved 10. January 2022 from <https://www.globalpropertyguide.com/Europe/Netherlands/Price-History>
- Gumin, M. (2022). *WaveFunctionCollapse*. Retrieved 10. January 2022 from <https://github.com/mxgmn/WaveFunctionCollapse>
- Guy, B. a. C., Nicholas (2005). Design for Disassembly in the built environment: a guide to closed-loop design and building. In C. o. Seattle (Ed.).
- Hardin, B., & McCool, D. (2015). *Bim and construction management : Proven tools, methods, and workflows*. John Wiley & Sons, Incorporated.
- International Organisation for Standardization (2015). ISO 12006-2:2015 Building construction — Organization of information about construction works — Part 2: Framework for classification. Retrieved 4. January 2023 from <https://www.iso.org/standard/61753.html>
- Kendall, S. (2003). *Product Bundling or Kitting: Balancing Efficiency and Variety in the Market*.
- Kendall, S. (2015). *Notes Toward a History of the Matura Infill System Development*.
- Klein, T. (2013). *Integral Facade Construction*. Delft University of Technology, Faculty of Architecture.
- Kleineberg, M. (2022). Wave Function Collapse. Retrieved 10. January 2022 from *GitHub*. <https://github.com/marian42/wavefunctioncollapse>
- Komura, T. (2008). *Scalar Algorithms: Contouring*.
- NBvT. (2014). *Wat is de KVT-online?* NBvT. <https://nbvt.nl/services/kvt/wat-de-kvt-online>
- Norén, M., & Gauthier, T. (n.d.). *IfcOpenShell—The open source IFC toolkit and geometry engine*. Retrieved 2 December 2022, from <https://ifcopenshell.org/>
- Level of Detail* | GSA. (n.d.). Retrieved 16 September 2022, from <https://www.gsa.gov/real-estate/design-and-construction/3d4d-building-information-modeling/bim-software-guidelines/document-guides/level-of-detail>
- Liu, Z. (1998). *Algorithms for Constraint Satisfaction Problems (CSPs)*.
- Lou, J., Lu, W., & Xue, F. (2020, November 27). *A review of BIM data exchange method in BIM collaboration*.

- McAndrew, C. (n.d.). *The building blocks for a new kind of architecture—The Bartlett Review 2021—UCL - University College London*. Retrieved 6 December 2022, from <https://www.bartlettreview.com/features/the-building-blocks-for-a-new-kind-of-architecture>
- NBS Source. (2022). <https://source.thenbs.com/>
- Lorensen, W., & Cline, H. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics* 21, 163-169.
- McAndrew, C. (2021). The building blocks for a new kind of architecture. Retrieved 02. November 2022 from <https://www.thebartlettreview.com/features/the-building-blocks-for-a-new-kind-of-architecture>
- Merrell, P. (2022). Retrieved 10. January 2022 from *Model Synthesis*. Retrieved 10. January 2022 from <https://paulmerrell.org/model-synthesis/>
- Nourian, P., & Azadi, S. (2022). *Boolean Marching Cube*. Retrieved 02. February 2022 from from topoGenesis: https://topogenesis.readthedocs.io/notebooks/boolean_marching_cubes/
- Oorschot, L. (2021). Modulair, circulair, opschaalbaar en betaalbaar -So You Think You Can BUILD challenge. *1M Homes Initiative*. Retrieved 08. August 2022 from <https://1mh.openaccess.ac/index.php/1mh/preprint/view/44/75>
- Oorschot, L., & Asselbergs, T. (2021). New Housing Concepts: Modular, Circular, Biobased, Reproducible, and Affordable. *Sustainability*, 13.
- Open Systems Lab. (2022, 01 10). [www.wikihouse.cc](https://files.cargocollective.com/c229869/WHouse_Guide_1.1.pdf). Retrieved from Wikihouse: https://files.cargocollective.com/c229869/WHouse_Guide_1.1.pdf
- Park, S. (2022). *Houscaper*. Delft University of Technology.
- Retsin, G. (2019). Discrete Architecture in the age of automation. *Architectural Design*, 7-13.
- restado. (n.d.). *Restado*. restado.de. Retrieved 7 December 2022, from <https://restado.de/>
- Rigips. (2021). *Verarbeitungsrichtlinien Trockenbau*. Düsseldorf: Saint-Gobain Rigips GmbH. Retrieved 23. September 2020 from [medien.rigips.de](https://www.rigips.de)
- Sass, Lawrence. (2005). Wood Frame Grammar. Computer Aided Architectural Design Futures 2005 [Proceedings of the 11th International Conference on Computer Aided Architectural Design Futures / ISBN 1-4020-3460-1] Vienna (Austria) June 2005, pp. 383-392. 10.1007/1-4020-3698-1_36.
- Savov, A., & Tessmann O., & Winkler R. (2020). Encoding Architectural Designs as Iso-surface Tilesets for Participatory Sculpting of Massing Models. 10.1007/978-3-030-29829-6_16.
- Soman, A. (2021). *Gen Arch*. TU Delft University of Technology.
- SPAX International GmbH & Co. KG (2022). *Spax Screws*. Retrieved 22. November 2022 from <https://www.spax.com/en/products/universal/flat-countersunk-head/universal-screw-5-x-70-mm-200-pieces-full-thread-flat-countersunk-head-t-star-plus-t20-4cut-wirox-1191010500705/pid-814/>
- Smith, G. (2015). *An Analog History of Procedural Content Generation*. Boston: Northeastern University, Playable Innovative Technologies Lab.

Stålberg, O. (2022). *Townscaper*. Retrieved 7 December 2022 from <https://oskarstalberg.com/Townscaper/>

TBI Woonlab. (2022, 10 11). *De beterBASIShuis Woonplanner*. Retrieved from <https://www.tbiwoonlab.nl/labels/beterbasishuis/woonplanner-beterbasishuis/>

TBI Woonlab. (2022, 01 12). *Houtbaar Loft Concept*. Retrieved from *Houtbaar*: <https://www.houtbaar.nl/loft/>

The New Makers. (2022, 01 09). *The New Makers*. Retrieved 11. January 2022 from <https://thenewmakers.com/product/uuthuuske/>

Trading Economics. (2022, 01 11). *Netherlands Home Ownership Rate*. Retrieved 11. January 2022 from <https://tradingeconomics.com/netherlands/home-ownership-rate#:~:text=Home%20Ownership%20Rate%20in%20Netherlands%20averaged%2067.55%20percent%20from%202005,of%2063.90%20percent%20in%202005>

Tsang, E. (1993). *Foundations of Constraint Satisfaction*. Academic Press.

van Overveld, M., van der Graaf, P. J., Eggink-Eilander, S., & Berghuis, M. I. (2011). *Praktijkboek Bouwbesluit 2012*. Sdu Uitgevers.

9.1 Figures

Figure 1: GoDesign Framework (Azadi & Nourian , 2021, p. 288)	8
Figure 2: Thematic organisation of the research (Author)	10
Figure 3: The Continuous and the Discrete (Author)	13
Figure 4: Assembled Discrete Building Blocks, Photo by James Harris (McAndrew, 2022)	13
Figure 5: Shearing Layers of Change (Stewart Brand, 1994)	14
Figure 6: Matura Infill System (Gan, S., Zhang, H. , 2021)	14
Figure 7: The Uuthuuske System (The New Makers, 2022)	15
Figure 8: Skylark System (Open Systems Lab, 2022)	16
Figure 9: Houtbaar Loft. TBI Woonlab (2022, 01)	16
Figure 10: Gablok System (Gablok, 2022)	17
Figure 11: Implicit Modeling with Rhino (Author)	18
Figure 12: Parametric Configurator of the beterBASIShuis (TBI Woonlab, 2022 10 11)	18
Figure 13: IKEA Kitchen Planner (IKEA, 2022)	19
Figure 14: Rebuilding Tradition In The Kathmandu Valley- Minecraft Model and Reality (Block by Block, 2022)	19
Figure 15: Two Designs created in seconds by the author in Townscaper (Stålberg, 2022)	20
Figure 16: Assisted Sculpting (Savov et. al, 2020)	20
Figure 17: Comparison of strategies for encoding designs (Author)	21
Figure 18: Map-coloring (author)	21
Figure 19: Workflow of the Wave Function Collapse Algorithm (Author)	22
Figure 20: A shape grammar and grammar derivation (Sass, 2005)	23
Figure 21: Visualization of Design Objects (Author)	24
Figure 22: Coordinate System (Author)	26
Figure 23: Derivation of the counting sequence of the cube faces (Author)	26
Figure 24: Counting Sequence (Author)	27

Figure 25: Connection types of sub-voxels (Author)	27
Figure 26: Dimensioning System (Author)	28
Figure 27: Stakeholders involved in different stages of the planning process (Author)	29
Figure 28: Flowchart of the Tile Database Structure	30
Figure 29: Applying the Marching Squares Algorithm (Author)	32
Figure 30: Marching Squares Tileset (Author)	33
Figure 31: Applying the Marching Squares Algorithm on different fields (Author)	33
Figure 32: Applying two tilesets with Marching Squares (Author)	34
Figure 33: 26 Unique Cube Configurations (Author)	34
Figure 34: 26 Unique Tiles generated with Grasshopper (Author)	35
Figure 35: Calculating all possible tiles from one unique tile (Author)	35
Figure 36: Flowchart of the Boolean Marching Cube Algorithm (Author)	35
Figure 37: Applying two tilesets each on two different set of voxels (Author)	36
Figure 38: Tile Set to test the WFC Algorithm (Author)	37
Figure 39: Applied tileset on a 20x20 matrix (Author)	38
Figure 40: extended tileset with furniture (Author)	38
Figure 41: Application of the extended tileset on a 20x20 matrix (Author)	38
Figure 42: Comparison of infill systems relevant for discrete design (Author)	39
Figure 43: Basic Infill Module (Author)	40
Figure 44: Extended Infill System (Author)	40
Figure 45: Services and Assembly (Author)	41
Figure 46: Application Methods for the Design Tool in context (Author)	43
Figure 47: Flowchart of the Design Tool (Author)	44
Figure 48: Materialized Design in exchange with a voxelized topological design (Author)	44
Figure 49: Kind of Voxels (Author)	45
Figure 50: Topological Design and Application of Layers (Author)	46
Figure 51: Pseudo-Code Voxel Reader (Author)	47
Figure 52: Encoding the Topological Design (Author)	48
Figure 53: ERD Diagram of the Tile Database (Author)	49
Figure 54: Two workflows to create Building Tiles (Author)	50
Figure 55: Unique Cubes and Families (Author)	51
Figure 56: Table: Demonstration of a Material Import (Author)	
Figure 57: Countersunk Screw (Spax)	52
Figure 58: Demonstration of Part Implementation (Author)	52
Figure 59: Assigning Components (Author)	53
Figure 60: Representation of a component through Parts, Instances and as a whole (Author)	53
Figure 61: Assigning Building Tiles (Author)	54
Figure 62: Assembly of a building tile (Author)	54
Figure 63: Demonstration of the creation of Computational Tiles (Author)	55
Figure 64: Encoding Faces (Author)	56
Figure 65: Converting an array of Cubes to Cube Subsets (Author)	57
Figure 66: Pseudo Code for creating cube subsets (Author)	57
Figure 67: Pseudo Code for the Algorithm to assign Tiles to Cube Subsets	58
Figure 68: Several Building Assemblies as Algorithm Output (Author)	59
Figure 69: User Interaction Diagram (Author)	60
Figure 70: Interactive Tile Placement in Rhinoceros	60
Figure 71: Van Neumann Stencil applied for the interactive tile placement (Author)	61
Figure 72: Detail Sections generated in Grasshopper (Author)	62
Figure 73: Structural Components (Author)	63

Figure 74: Integration of Membranes (Author)	64
Figure 75: Defining a structural direction to produce ceilings (Author)	64
Figure 76: Application of the system on spatially restricted projects (Author)	65
Figure 77: beterBASISHuis (TBI Woonlab, 2022, 10 11)	67
Figure 78: Site Context of the Toy Problem (Author)	68
Figure 79: Topological design of a rowhouse	68
Figure 80: Computational Tiles from the Envelope Tileset (Author)	69
Figure 82: Envelope Tileset Brickwork	70
Figure 83: Envelope Tileset Timber (Author)	71
Figure 84: Infill Tileset (Author)	72
Figure 85: Additional Tiles Infill System (Author)	73
Figure 86: Detail Integration Hörmann BaseLine with block frame (Author)	73
Figure 87: Detail Integration Hörmann BaseLine with enclosing steel frame (Author)	74
Figure 88: Detail Integration Hörmann BaseLine with enclosing wood frame (Author)	74
Figure 89: Envelope Variations (Author)	75
Figure 90: Applied Infill System (Author)	76
Figure 93: Detail Infill System (Author)	77
Figure 95: Skylark 250 Blocks (Open Systems Lab, 2022)	78
Figure 96: Application of the Skylark 250 system onto a voxel set (Author)	79
Figure 97: Adjusted Blocks and Computational Tiles (Author)	80
Figure 98: The Design Tool in the context of stakeholders (Author)	80
Figure 99: Application of the design tool to different markets (Author)	84
Figure 100: Automatic Tile Placement (Author)	87
Figure 101: UX design Potential Platform (Author)	88