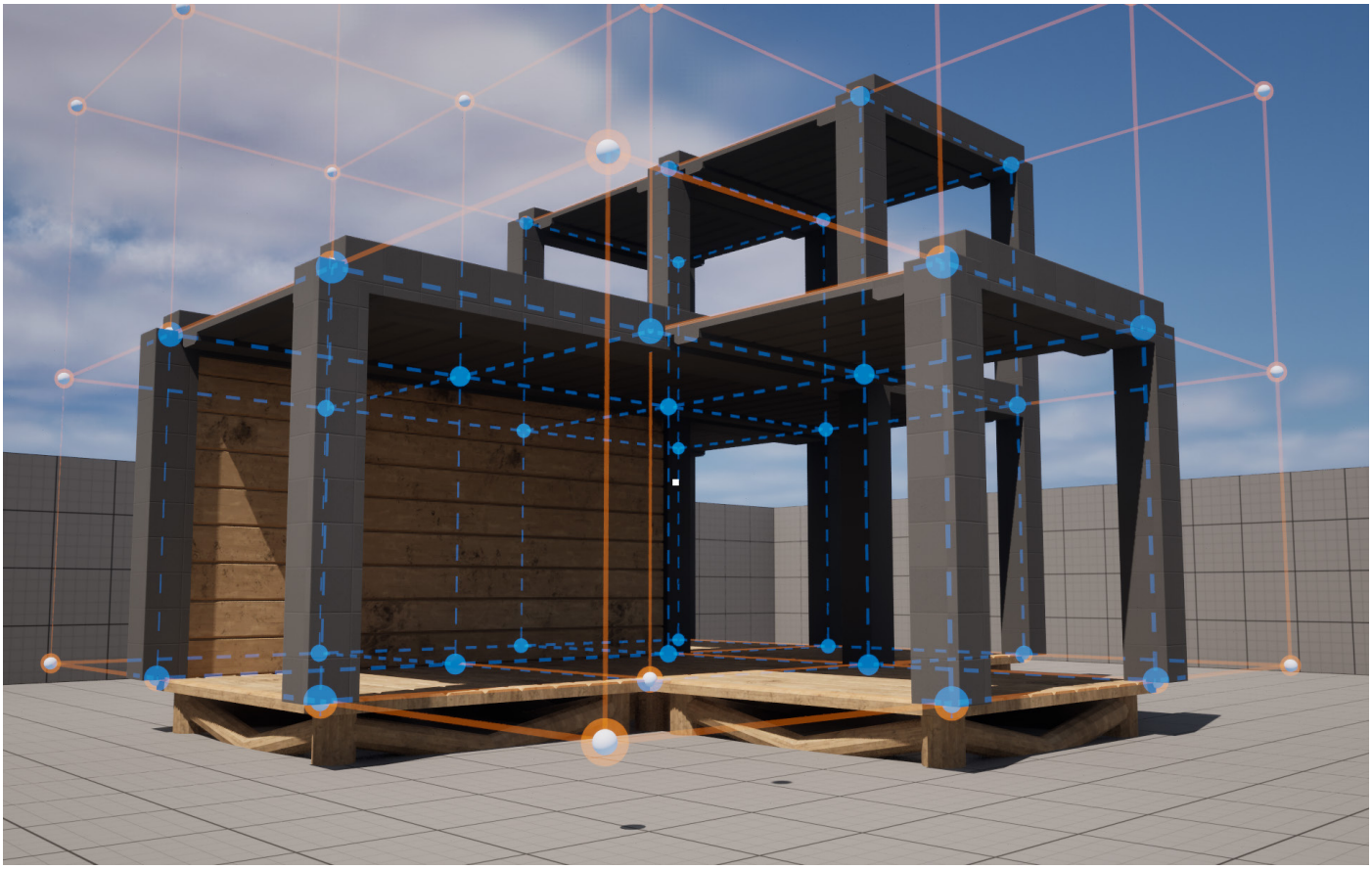
Construction configurator

Increasing the design space of integrated construction configurators



TU Delft
BK Bouwkunde

MSc Thesis by B. M. Smeekes
27-05-2024

Delft University of Technology
Faculty of Architecture and the Built Environment

Construction configurator

Increasing the design space of integrated construction configurators

Key words: construction, configurator, integrated, design space, database, structural, BIM

Author
Brent Smeekes | 4607643
Master thesis research
Delft | 2024

MSc in Architecture, Urbanism and Building Sciences
Building Technology track

Mentors
Dr. Stijn Brancart | Chair of Structural Design & Mechanics
Ir. Hans Hoogenboom | Chair of Design Informatics

Board of examiners delegate
Ir. Leontine de Wit | Architect

Delft University of Technology
Faculty of Architecture and the Built Environment

Abstract

A configurator is a platform which serves the goal of mass customization. It contains a set of common components from which a stream of derivative products can be efficiently developed and launched. Configurators in the construction industry have the potential to help integrate the design-to-production supply chain. Integration is required because there is a rising complexity in design and construction projects due to an increased number of parties involved in each venture. Configurators achieve integration by ensuring that a design is within the production capacity.

(Cao et al., 2021) identified three distinct typologies of existing construction configurators: planning, design, and production configurators. These typologies primarily indicate the stage of the construction phase in which the configurator is used. An integrated configurator includes all typologies/phases. The application of configurators in construction is limited and immature, this is especially the case for integrated construction configurators.

The application of configurators in construction is limited because current configurators lack scalability, many configurators may only be fit for one generation of products from one company. This issue originates from insufficient cross-organizational collaboration and integration with supply chains. Additionally, academia indicates a need for research on increasing the design space of modular buildings. For this reason, the research question of this thesis is: How can the design space of integrated construction configurators be enlarged?

The approach for developing an integrated construction configurator which increases the design space is inspired by games with building systems such as Valheim. It consists of a grid, modularity on building component level, and configuration rules which dictate how components are placed on the grid. However, a major limitation of these games is the lack of proper analytical tools to validate configurations for physical construction. A configurator was developed which overcomes this limitation by implementation of a structural analysis.

The configuration process of this configurator consists of 3 parts: the grid, the components, and the analysis. The grid defines the design space in which components can be placed, components from a building product database ensure product manufacturability, and analyses ensure that the assembly of components is possible. In each iteration of the configuration process, the configuration is controlled by editing the grid and placing or removing components on the grid. Then, the configuration is evaluated by means of analysis, results from the analysis are the basis for edits to the configuration in the next iteration cycle.

The main benefits of this type of configurator compared to BIM are that by integrating the manufacturing and assembly constraints in the design process, early design decisions are evaluated on manufacturability. This can avoid design issues that require reorganisation efforts and put pressure on the supply chain. Consequently, the configurator reduces the time and cost for the design and production phases, allowing users of the configurator to develop buildings faster and cheaper.

Table of contents

Table of contents

1. Existing configurators

1.1 Context on configurators

1.1.1 Mass customization

In the late 1980s, the manufacturing industry shifted from mass production to mass customization in response to the growing demand for product variety (Kotha & Pine, 1994). Mass customization is a manufacturing paradigm which has the ability to enable design flexibility and therefore produce products which match the customer's preference more closely (Barman & Canizares, 2015). It does this while maintaining the main advantages of mass production, the economies of scale. This means that mass-customized products are produced at the same price, quality, and time as mass-produced products (Cao et al., 2021).

Mass customization requires some degree of standardization and is compatible with prefabrication strategies (Bianconi et al., 2019). 'Effective implementation of mass customization would enable design flexibility that aligns with both customers' preference and manufacturers' capabilities' (Cao et al., 2021).

1.1.2. A definition of configurators

A configurator is a platform which serves the goal of mass customization. Configurators contain 'a set of common components, modules, or parts (e.g. the kit-of-parts) from which a stream of derivative products can be efficiently developed and launched' (Meyer & Lehnerd, 1997), see figure 1.1.

The kit-of-parts of a configurator are configured based on processes, these are the rules and taxonomies that guide the placement of parts. The processes are relying on a knowledge base, these are the datasets necessary for economic evaluation of the configuration (Louth et al., 2024).

This economic evaluation can come in the form of a Bill-of-Materials (BOM), which in essence is translating the customer specifications into product documentation. Since the kit-of-parts are modelled digitally to represent the actual components in production, it provides access to the rules and constraints originating from production (Cao et al., 2021). These rules and constraints make sure that designs made with the kit-of-parts are within production capacity.

Three main characteristics are inherent in configurators (Cao et al., 2021):

- reusability due to kit-of-parts,
- intelligence driven by embedded expert knowledge,
- high automation realized by off-the-shelf technologies, such as Application Programming Interface (API).

1.2 Potential of configurators in the construction industry

1.2.1 Opportunities for configurators

Industrialized construction is gaining more share in the construction market (Cao et al., 2021). Moreover, 'the Architecture, Engineering, and Construction (AEC) industry encounters a rising complexity in the design and construction of projects due to the increased number of parties involved in each major venture' (Abanda et al., 2017). Consequently, past research suggests the importance of a close collaboration between designers and manufacturers, such as a Design for Manufacturing and Assembly (DFMA) design team (Yuan et al., 2018). More specifically, research suggests the need to develop prefabricated construction building information models containing the production

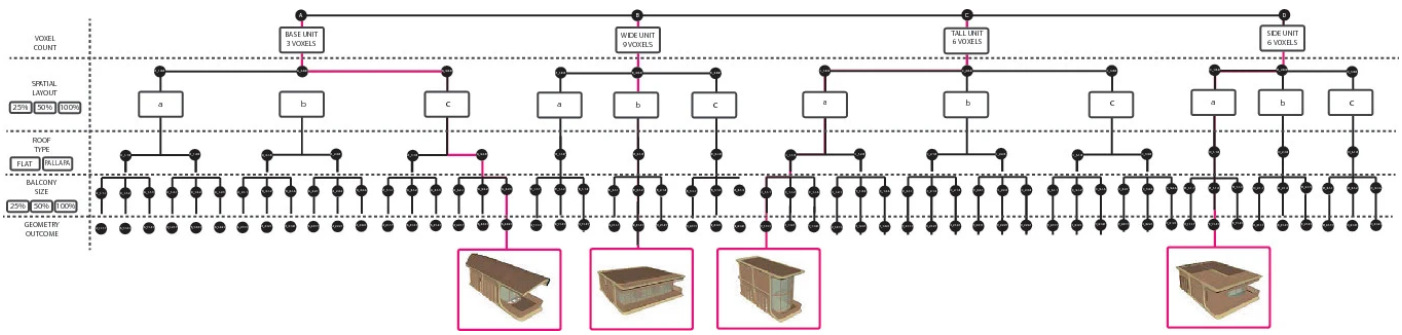


Figure 1.1: Industrialised construction building components tree (Louth et al., 2024)

details (Hamid et al., 2018).

Designers do not have sufficient knowledge on production to understand the rules and constraints that come with building components and assemblies. This results in design issues which put pressure on the supply chain and requires reorganization efforts and expert resources from the design company (Cao et al., 2021).

Configurators offer an opportunity to help integrate the design-to-production supply chain but require more efforts from manufacturers to develop kit-of-parts and configurators. Configurators ensure that a design is within the production capacity, this reduces estimated cost deviation caused by redesign. The reuse of the library of kit-of-parts in future projects will lead to continuous improvements of project quality and return of investment made initially inside a single project (Tetik et al., 2019). Other benefits of applying configurators in industrialized construction are shown in figure 1.2.

1.2.2 Comparison with BIM

Current BIM software can support collaboration among different stakeholders including planners, architects, engineers, and contractors. However, the complexity of BIM's technical workflow limits the possibility of engaging with clients, end-users, and other non-engineering professionals in the decision-making processes due to the so-called "black-box effect", which refers to a system without transparency (Potseluyko et al., 2022).

Moreover, few studies give attention to the possibilities of developing a product platform based on BIM contents (Piroozfar et al., 2019). Instead, 'most of the projects are built from scratch in the BIM environment' (Cao et al., 2021).

The benefit of a configurator based on BIM contents is that the contents can be used for analyses such as a Bill-of-Material (BOM) analysis. 'Such a shift can enable configuration lifecycle management built on BIM, and thus extend the application of BIM across all lifecycle phases of a product' (Cao et al., 2021). Decoupling the configurator from the dataset of BIM contents guarantees that the kit-of-parts is edited in one place. The kit-of-parts are modelled in BIM software and the configuration of the kit-of-parts happens in the configurator. This decoupling is important because it means that the kit-of-parts model and information is controlled by the manufacturer of that part.

	Benefits of Configurators
Product	Increases flexibility (Thuesen and Hvam, 2011; Jensen *et al.*, 2014; Bonev, Wörösch and Hvam, 2015; Smiding, E., Gerth, R., Jensen, 2016)
	Ensures product manufacturing and assembly of solutions is possible (Bonev, Wörösch and Hvam, 2015; Said, Chalasani and Logan, 2017; Jansson, Viklund and Olofsson, 2018; Yuan, Sun and Wang, 2018)
People	Minimizes the need for manual involvement (Frank et al., 2014; Jansson, Viklund and Olofsson, 2018; Lee and Ham, 2018)
	Smooths the learning curve (Jansson, Viklund and Olofsson, 2018)
Process	Reduces time and cost for design and production (Thuesen and Hvam, 2011; Frank *et al.*, 2014; Bonev, Wörösch and Hvam, 2015; Smiding, E., Gerth, R., Jensen, 2016; Said, Chalasani and Logan, 2017; Jansson, Viklund and Olofsson, 2018)
	Enhances coordination efficiency (Malmgren, Jensen and Olofsson, 2011; Xu *et al.*, 2018; Yuan, Sun and Wang, 2018)
	Develops construction documents efficiently (Jensen, Olofsson and Johnsson, 2012; Smiding, E., Gerth, R., Jensen, 2016; Jansson, Viklund and Olofsson, 2018)
	Preserves and reuses knowledge for the next product (Frank *et al.*, 2014; Jensen *et al.*, 2014)
	Increases reliability of schedule (Wu *et al.*, 2010; Larsson *et al.*, 2015)

Figure 1.2: Benefits of configurators applied in industrialized construction (Cao & Hall, 2019)

1.3 Categorisation of configurators

1.3.1 Levels of mass customization

One way of categorizing configurators is by their level of mass customization. The level of mass customization indicates the level of freedom that the configurator enables for the product design. It also relates to the degree of which the customers' needs are incorporated into the product development. This means that a product with a low level of mass customisation doesn't give the customer a lot of options to customise the product to their desire.

Existing configurators

Hvam and Forza similarly classify product development to four levels of mass customisation based on the Consumer Order Decoupling Point (CODP) (Forza & Salvador, 2006) (Hvam et al., 2008). The CODP is 'the point in the supply chain at which consumer orders are converted into production orders or schedules' (Cao et al., 2021). The four different CODP scenarios are:

- Engineer to order, define design and manufacturing rules.
- Modify to order, edit design parameters.
- Configure to order, select scalable modules and module interfaces.
- Select variant, select off-the-shelf products.

1.3.2 Typologies of existing construction configurators

(Cao et al., 2021) identified three distinct typologies of existing construction configurators, see figure 1.3. These typologies are based mainly on when the configurator is used in a project. But the typologies also include information on who the main users are, how the configuration is performed, what the targeted products are, and what the expected output is of the configurator. The typologies are:

- Planning configurator, the targeted product for this stage could be a conceptual building model created by architectural firms for early planning. The configurator helps real estate developers by allowing customers to interactively design their desired house by offering available selections. Moreover, intelligent algorithms can identify more feasible or advantageous plans using embedded configuration rules.

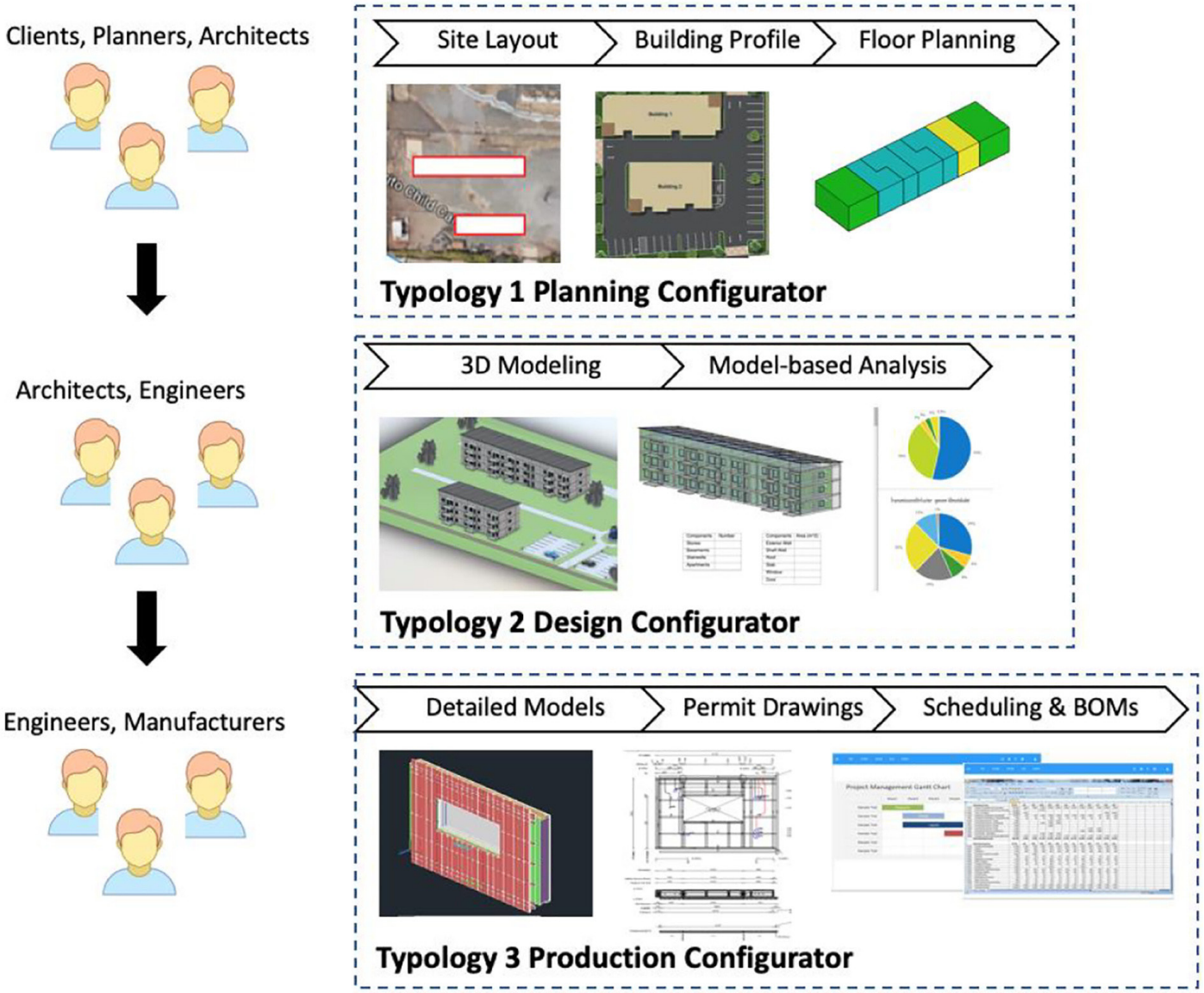


Figure 1.3: Configurator typologies (Cao et al., 2021)

- Design configurator, this is the most common typology for configurators in the market. The targeted product could be a detailed building model configured with predefined modules, such as timber panels. The generated output can be converted to BIM models to serve as a starting point for designers and engineers.
- Production configurator, this typology is mostly used by engineers and fabricators. The targeted products can be buildings planned, produced, and assembled by a vertically integrated firm. These types of configurators have the potential benefit of integrating manufacturing and assembly constraints in the design phase by adopting design for manufacturing and assembly principles. The outputs include BIM models, G-codes for NC (numeric control) machines, permit drawings, and bill of materials.

1.3.3 Integrated configurator

The last configurator typology is an integrated configurator, where the planning, design, and production is all included in the configurator. This type of configurator evaluates (early) design choices on manufacturability. Other benefits include the reuse of process and technical solutions, and that the formation of a stable supply chain can be facilitated (Cao et al., 2021). Integrated configurators enable stakeholders to maintain a common environment to control the project data.

1.4 Existing construction configurators

1.4.1 Construction configurator review

Figure 1.5 shows 15 configurators, they include both commercially deployed configurators and configurators developed for academic purposes. According to (Louth et al., 2024) 'configurators have seen a rise in web-enabled apps as exploration for property search and acquisition, residential test fitting, site planning and land utilisation as early-stage planning toolkits', the most notable commercially deployed web-based configurators being HiStruct, AGACAD, Creatomus, and MyProjectFrog.

Research from (Cao et al., 2021)

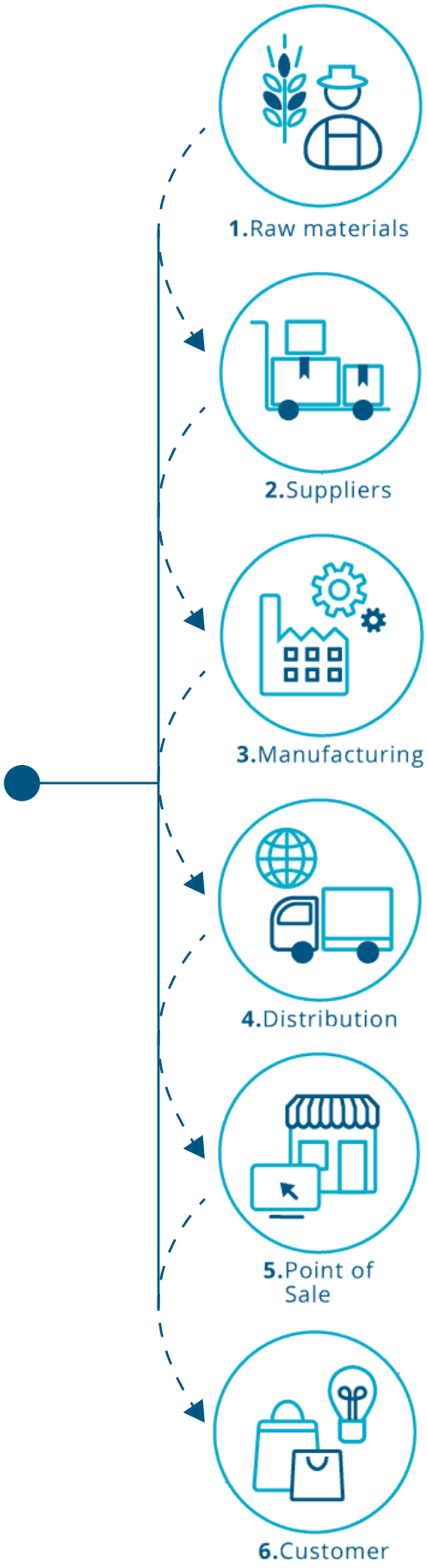


Figure 1.4: Supply chain integration

points out that the most popular typology is typology 2, design configurators. Typology 3, production configurators, seems to be the least common. For the CODP scenarios configure to order is the most common and engineer to order is the least common. Thus, configurators with complex production requirements and information available on production are especially rare.

There are a few configurators that span across multiple typologies and therefore can be called integrated configurators. However, none of these configurators manage to integrate all typologies. Most of the integrated configurators integrate across the planning and design process. These configurators include commercial configurators Testfit, PRISM, Hypar, and academic configurators Beyabu by (Louth et al., 2024), and the modular apartment configurator by (Cao et al., 2021). Only the commercially deployed configurators MyProjectFrog, and Uuthuuske configurator by The New Makers integrate across the design and production process.

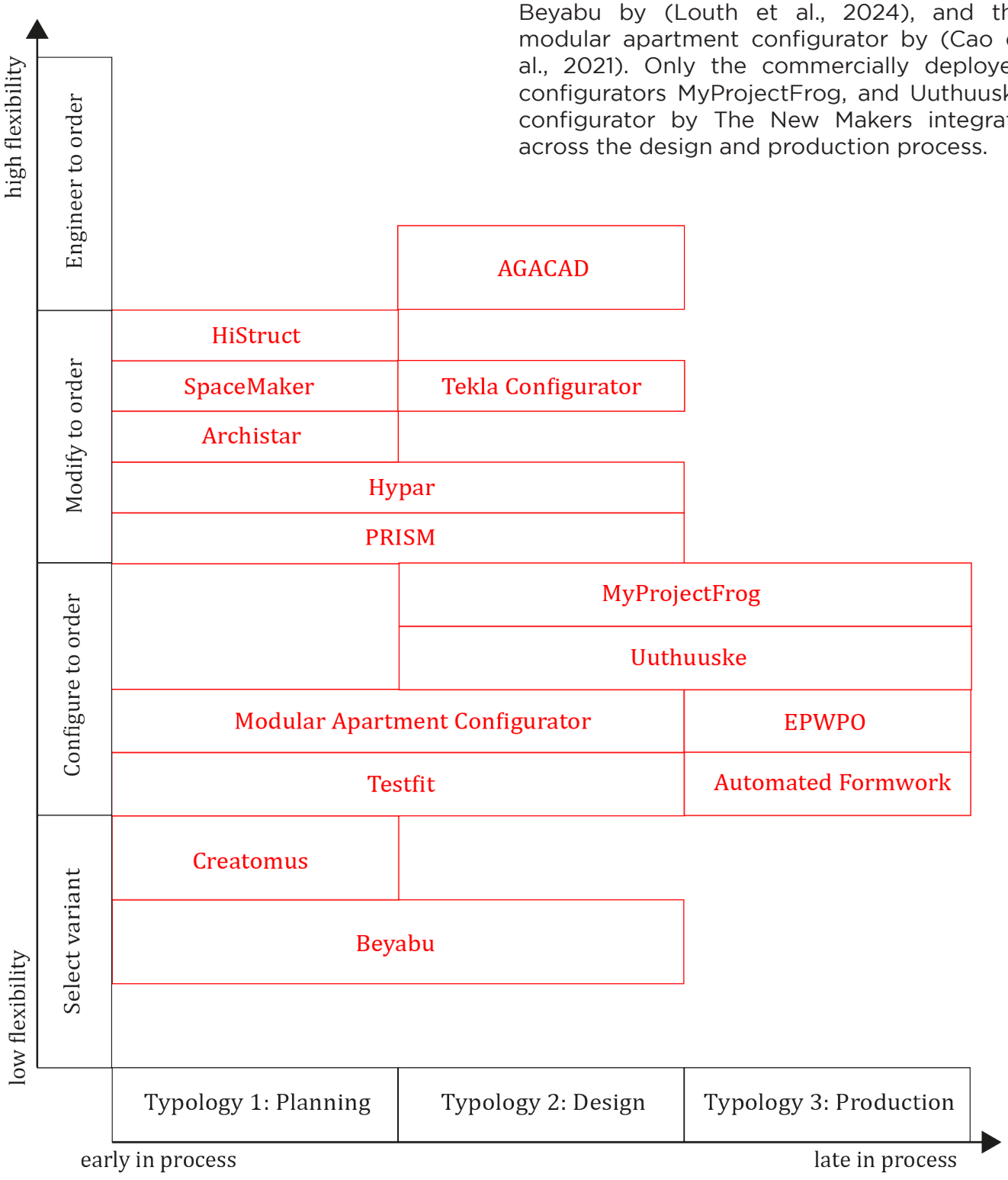


Figure 1.5: Categorisation of existing construction configurators in CODP and typologies

1.4.2 Beyabu configurator

The Beyabu configurator was developed at Zaha Hadid Architects (Louth et al., 2024). The configurator produces a planned community arrangement for a charter city development. Prospective home buyers and investors get to make decisions collaboratively on where their unit should be placed and the typology of their unit. After multiple configuration sessions, the configuration options of each user were submitted to generate a unified system grid to position within. Then the users make their first placements and decide on unit details, together a multi-family residential design is developed.

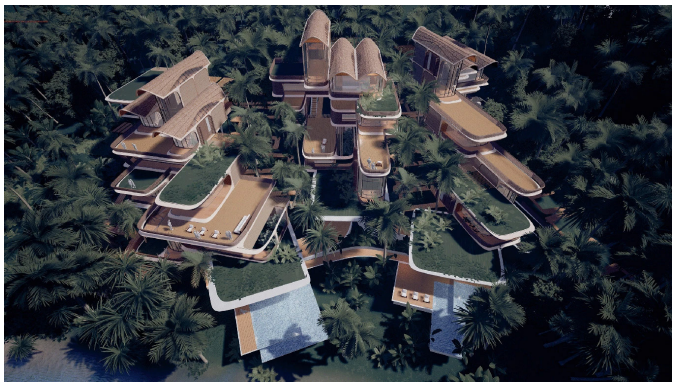


Figure 1.6: Beyabu configurator's visualisation of a configuration (Louth et al., 2024)

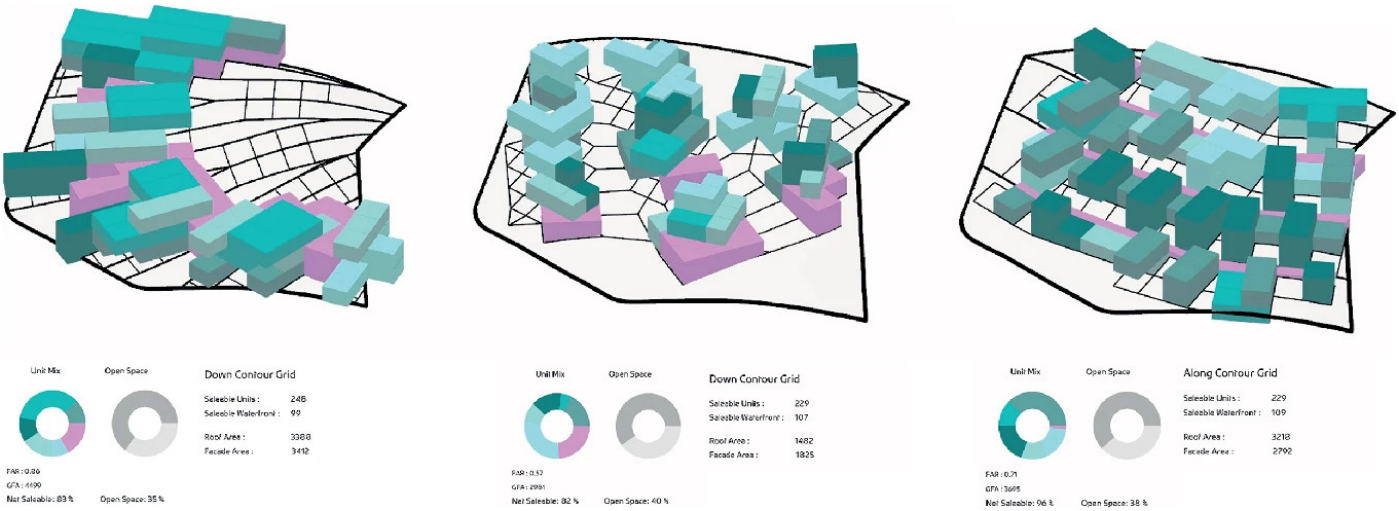


Figure 1.7: Beyabu configurator's variations in a system grid (Louth et al., 2024)

Figure 1.8: Beyabu configurator's gameboard grid. (Louth et al., 2024)

Existing configurators

1.4.3 PRISM configurator

PRISM is an open-source app which is developed to deal with London's housing crisis by accelerating the design process for 'Precision Manufactured Housing'. It works by choosing a site, then adjusting parameters that define what type of building the program generates. The user is given real-time analytical feedback and is alerted to potential planning issues, guiding the user to make appropriate decisions.

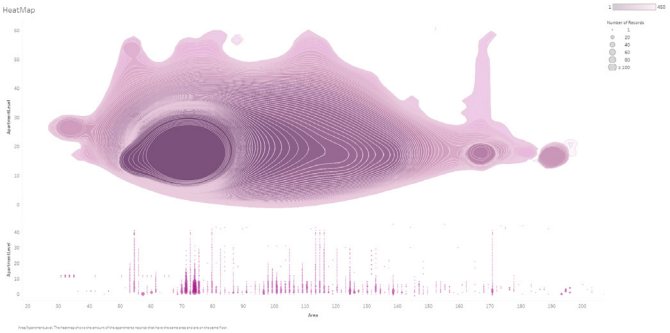


Figure 1.9: PRISM's analytics dashboard

Figure 1.10: PRISM's analytics dashboard

1.4.4 MyProjectFrog configurator

MyProjectFrog is a commercial configurator for timber panelised structures. It is built upon a library of panel products coupled with rules from local building regulations (Cao et al., 2021). This configurator is able to provide both design and fabrication deliverables.

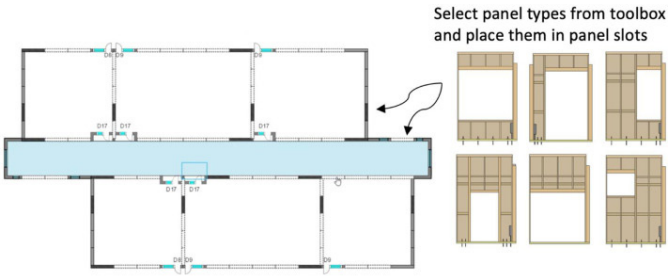


Figure 1.11: MyProjectFrog's configurator (Cao et al., 2021)

1.4.5 Uuthuuske configurator

Uuthuuske is a product developed by The New Makers which offers a solution for residential housing by making use of temporarily available locations. The product consists of modular building blocks that can be linked to create a range of floor plan layouts, these configurations can be explored in their configurator.

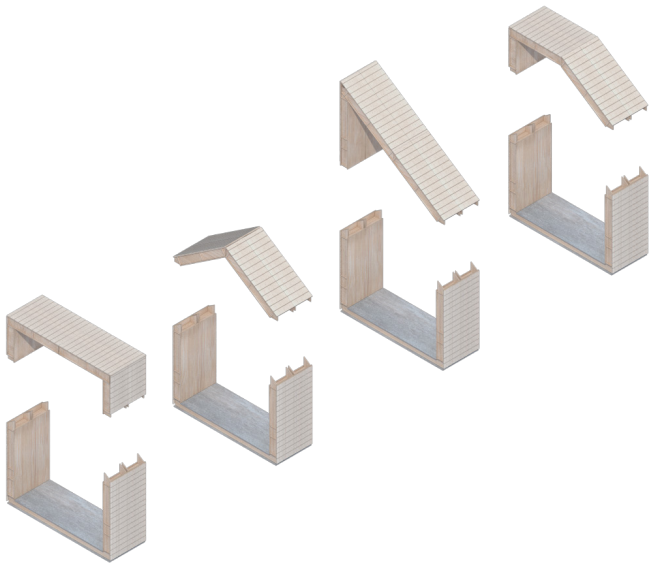


Figure 1.12: Uuthuuske's modules

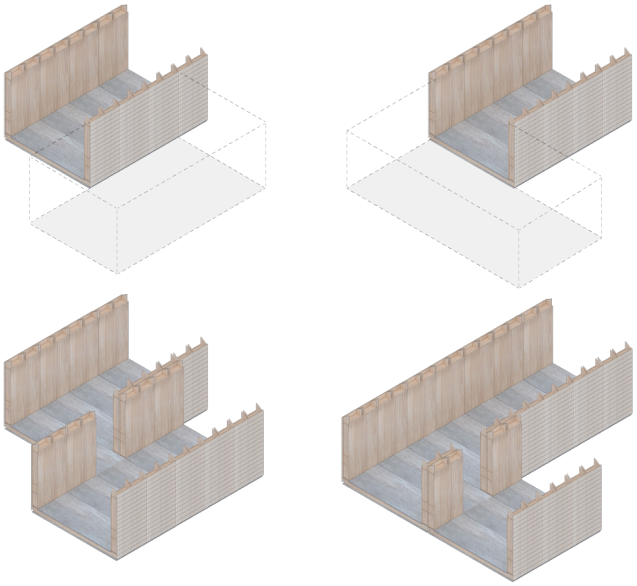


Figure 1.13: Uuthuuske's configuration options

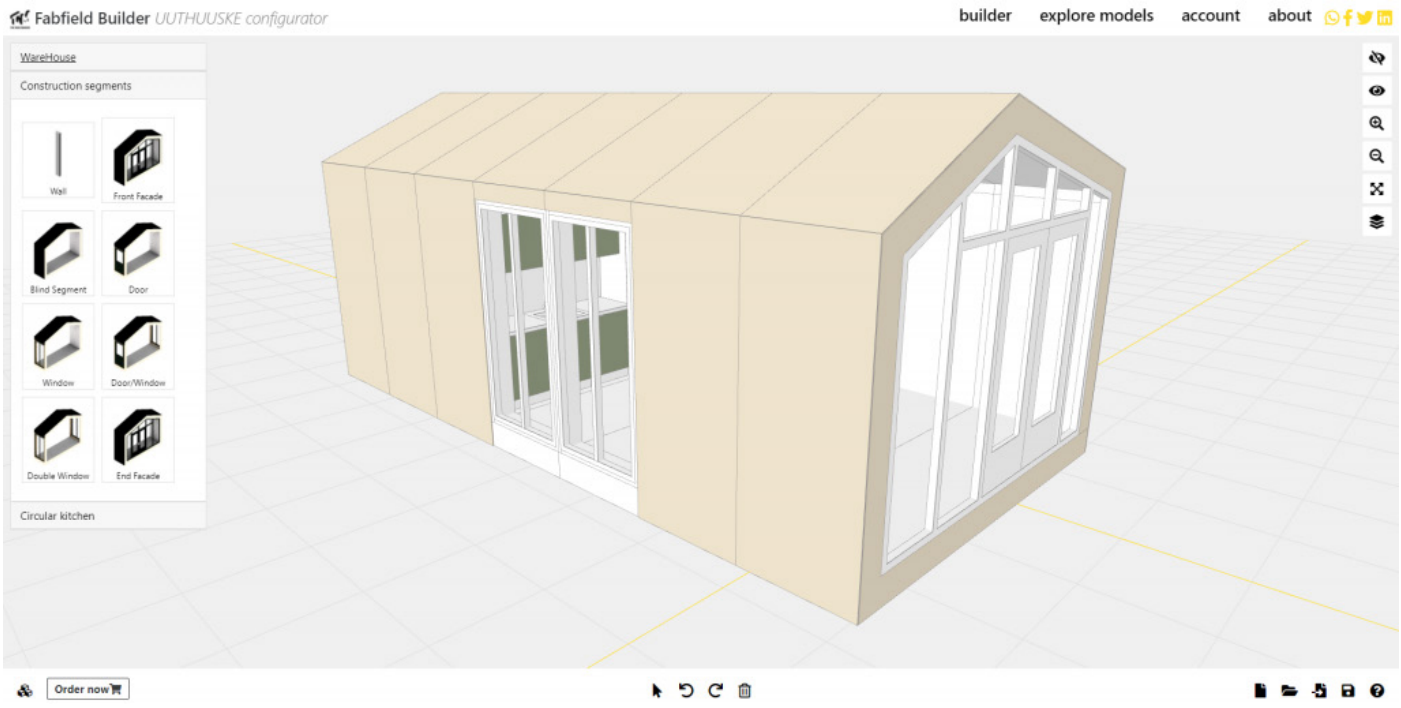


Figure 1.14: PRISM's analytics dashboard

1.5 Problem statement

Why isn't the application of configurators in construction common practice?

1.5.1 Integration

The application of configurators in construction is limited and immature (Cao & Hall, 2019). More research is required to see if the benefits of configurators apply to construction (Cao et al., 2021). Especially more research on integrated construction configurators is required, as the research by (Cao & Hall, 2019) indicates that many configurators may only be fit for one generation of products from one company due to the lack of scalability. There are a limited number of configurators with 'an integrated approach supporting design-to-production' (Cao et al., 2021), despite stakeholders in the construction industry seeking for such an integrated configurator (Cao & Hall, 2019).

Integration and successful application of configurators depends on cross-organizational collaboration (Myrodia et al., 2018) and supply chains (Cao et al., 2021). (Potseluyko et al., 2022) indicate that for many architectural and construction practices in the UK, collaboration with clients happens through email as primary communication medium. Configurators offer the opportunity for active real-time collaboration between multiple stakeholders, for example through participatory decision-making. Research by (Louth et al., 2024) offers a precedence as they developed a multi-user platform for the configuration and customization of a multifamily residential design, see figure 1.15.

Lack of integration of supply chains causes many construction companies to be unwilling to apply configurators in projects. The reasons are uncertainties and potential disturbance to their original project delivery processes and technical environments (Cao & Hall, 2019). Although, these barriers seem to be disappearing due to important advancements in digital technologies and configuration platforms (Goulding & Rahimian, 2019).

An advancement which may help with the integration of supply chains lies in the management of the kit-of-parts database. A flexible data structure for editing and exchanging kit-of-parts is required as a common environment (Cao et al., 2021). This would allow for the coordination of the parts during creation. This is especially important if the kit-of-parts are procedurally generated as opposed to being static geometry. Additionally, automatic database-to-configurator synchronisation of the kit-of-parts would be beneficial, because the 'manual maintenance and coordination of the configuration kit components are not a scalable protocol' (Louth et al., 2024)

1.5.2 Modularity on component level

'The configurator content could present a local marketplace of digital assets licensed to different regional or local suppliers and artisans who could commercialise their products in the platform' (Louth et al., 2024). This means a transition from configurators with modules

Figure 1.15: Unit position analysis of a multi-user platform (Louth et al., 2024)

which are room sized or larger, to modules which are building component or sub-assembly sized, see figure 1.16. Transitioning to smaller modules increases the design space of modular buildings. Which according to (Cao et al., 2021) is a topic which needs more research on, as there are currently few research studies regarding 'the flexible spatial arrangement of components to achieve design variations'.

'Previous studies have suggested that modularization strategies can be used to group components for offsite fabrication to ensure the ease of assembly and flexibility of building maintenance' (Peltokorpi et al., 2018). Another opportunity lies in the consideration of reusability when developing a kit-of-parts, but research on this topic is scarce (Cao et al., 2021).

Modularity on component level may also lead to a reduction in material use and lifecycle management complexity compared to concepts with room-sized modules. This is due to additional structural requirements to self-support room-sized modules during transportation and handling, and due to additional implications for modular design (Benjamin et al., 2022).

In contrast, modularity on component level offers the opportunity to fully automate the iterative process to find the optimal types and sizes of structural elements. (Benjamin et al., 2022) calls for such a structural analysis which is adapted to the singular characteristics for every structural system to test modular structures.

1.5.3 Problem statement summary

In summary, the application of configurators in construction is limited because current configurators lack scalability. This issue originates from insufficient cross-organizational collaboration and integration with supply chains. Academia indicates a need for research on the management of the kit-of-parts database, increasing the design space of modular buildings, the reusability of the kit-of-parts, and an approach to structural analysis for modular structures.

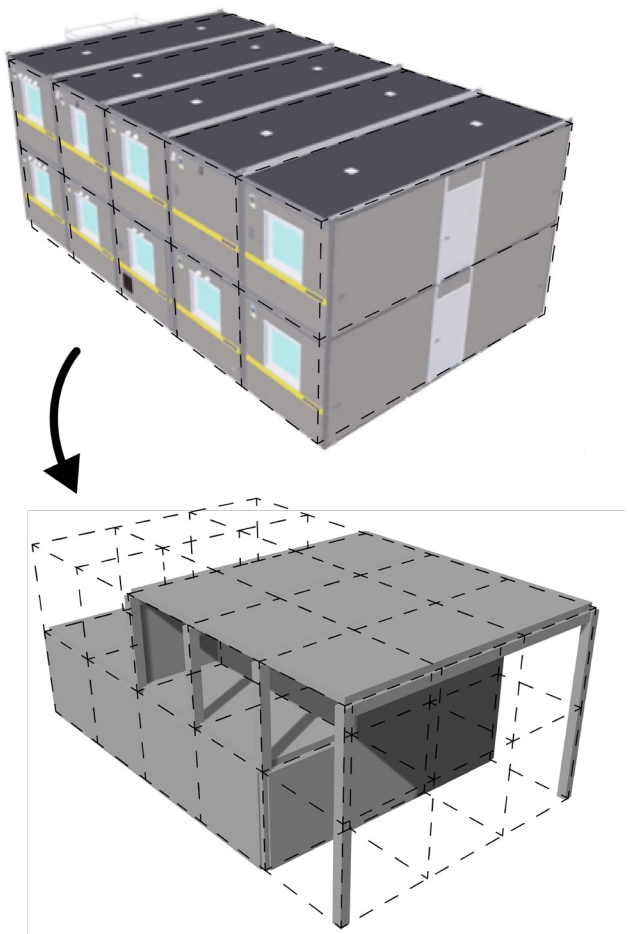


Figure 1.16: Transitioning from room-sized modules to building component sized modules

1.6 Objectives

Solving the lack of scalability in configurators requires an integrated construction configurator. Both the integration and the enlargement of the design space can be solved by using building components and assemblies as the kit-of-parts. For this reason, the objective of this research is to discover how an integrated construction configurator which enlarges the design space should function, and to develop (part of) the integrated construction configurator. This objective leads to the following main research question and sub-question:

Research question:
How can the design space of integrated construction configurators be enlarged?

Sub-question:
What are the benefits of this type of integrated construction configurator compared to prevalent BIM software?

1.7 Approach & methodology

Literary research on existing configurators
First, literary research is conducted on current construction configurators and research on construction configurators. This literary research is utilized to understand what a construction configurator is, how it compares to BIM, how they can be categorized, which ones are currently available, and what is missing in current construction configurators.

Software proposal and development for current project's configurator
Secondly, the literary research and inspiration from the game industry is used to propose a new construction configurator with features and improvements over current construction configurators.

Then, the theory and plan for the proposed construction configurator will get tested by translating it into code and software. For this step, Unreal Engine will be used to serve as the backbone for the software, it is the 3D environment in which the software development happens. Within Unreal Engine, code specific to the project will get written.

Proposing future developments
Lastly, the developed configurator will be contextualised by envisioning a more fully developed form. This includes examining how the configurator's building product database and configuration process should function. Moreover, the potential benefits of the configurator are discussed and the configurator is compared with prevalent BIM software.

2. Current project's configurator

2.1 Inspiration and theoretical framework for configurators

2.1.1 Inspiration from game industry

The inspiration for this project came from video games, specifically those with building systems like Minecraft, Fortnite, The Sims, and Valheim. These games make building structures easy and enjoyable, a stark contrast to the complexity found in CAD/BIM software used by architects. This disparity raises the question: why hasn't the construction industry adopted similar principles from these games to simplify the design process?

Research from Potseluyko et al., 2022, concluded that a game-like platform combined with BIM could provide simplified data delivery to a client. Another research project by Louth et al., 2024, utilises gamification principles to improve the user's problem-solving skills in a cooperative setting. Gamification is the process of turning a function-focused design into a human-focused design (Louth et al., 2024). This happens by 'employing concepts of behavioural science to engage human emotions through human psychology to motivate and incentivise users' (Louth et al., 2024). Figure 2.3 shows the strategy that Louth et al. used to implement gamification principles in their configurator through the Octalysis framework.

Figure 2.1: Sidney Opera House built in Minecraft by player

Figure 2.2: Player-made wooden structure in Valheim

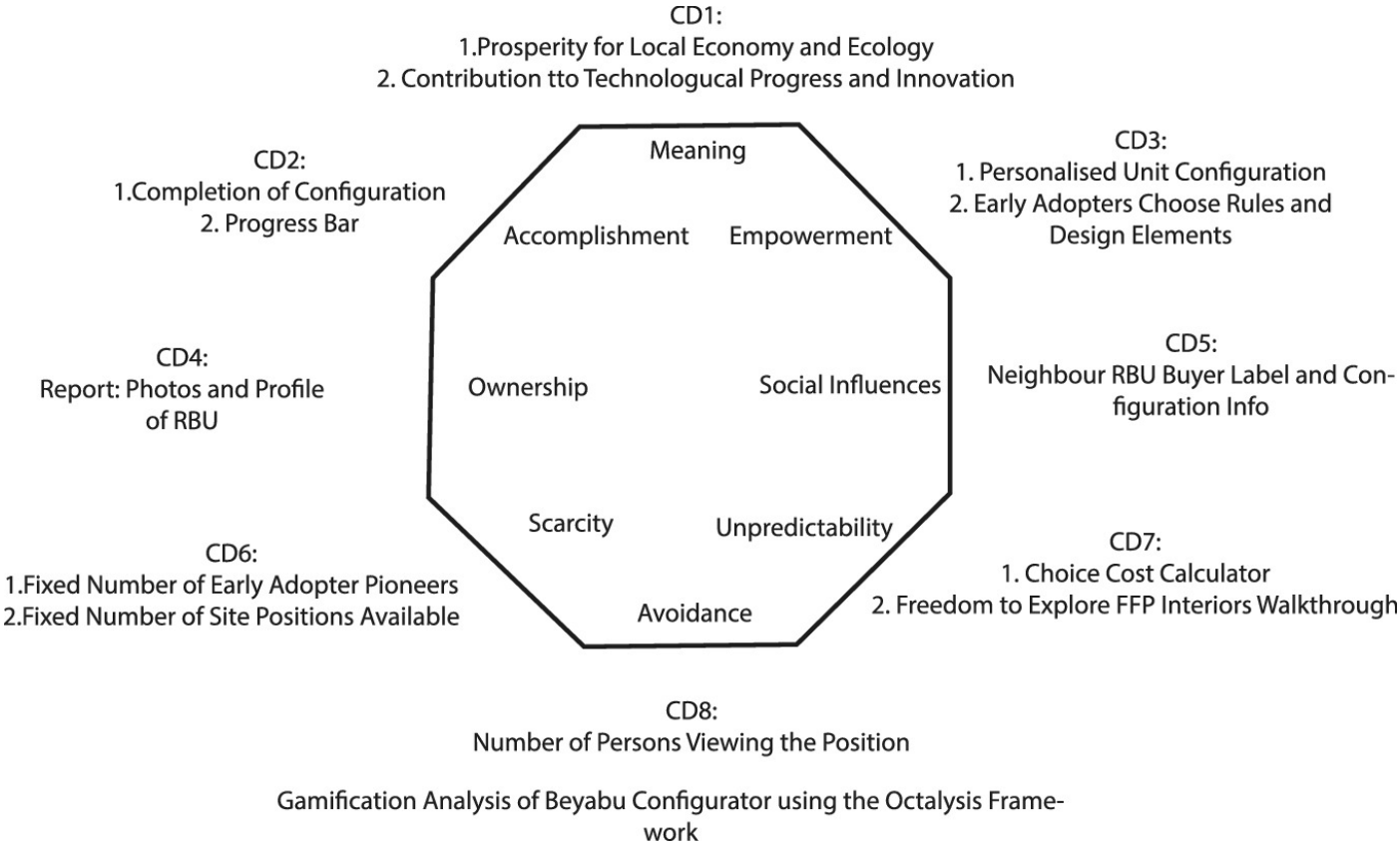


Figure 2.3: Configurator gamification analysis using octalysis framework (Louth et al., 2024)

2.1.2 Theoretical framework for configurators

Video games provide several key lessons for construction configurators. These games typically use a uniform grid system, which ensures building components snap and connect properly, reducing errors.

Each game features different building components or modules, varying in size, geometry, and materials. Despite these differences, all games utilize a database of diverse building components, which can include linear, planar, and volumetric elements, see figure 2.4.

The process of placing these components on the grid, known as the configuration process, varies among games. A crucial aspect of the configuration process is the User Interface (UI), which dictates how building components are selected from the database and placed. Video games have highly developed configuration processes, with significant effort invested in making the interaction with the software engaging and enjoyable.

Configuration rules determine where and how many building components are allowed to be placed. (Cao et al., 2021) proposes four types of configuration rules:

- Composition rules, they define which building components are mandatory or optional in the product architecture.
- Compatibility rules, they define which building components cannot exist simultaneously in the product.
- Dependency rules, they define which building components must belong together in a product.
- Cardinality rules, they define the required or limited number of building components under certain circumstances.

Games primarily use composition and cardinality rules, while compatibility and dependency rules are less common.

The building systems in these games encompass all fundamental features of a configurator: a kit-of-parts product structure and a configuration rule engine. While the goal of these games is not to create physical products, they effectively function as configurators within the digital realm. Unique features such as grids, modular components, and engaging

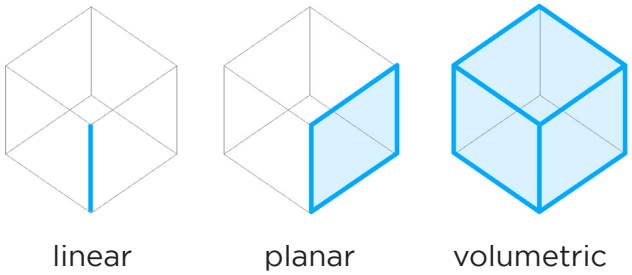


Figure 2.4: Linear, planar, and volumetric elements

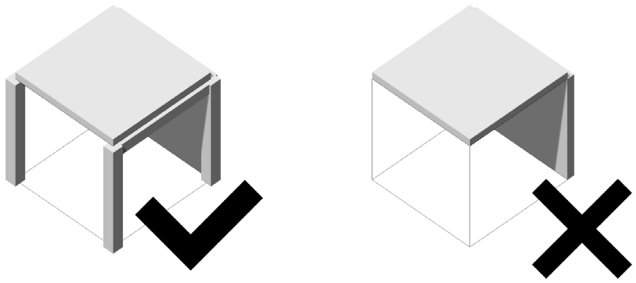


Figure 2.5: Composition rules

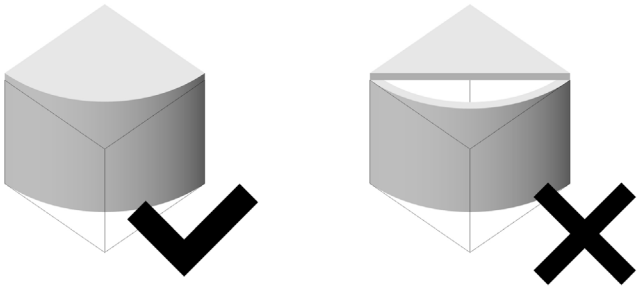


Figure 2.6: Compatibility rules

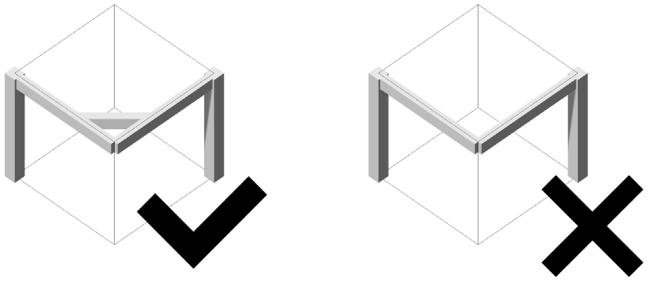


Figure 2.7: Dependency rules

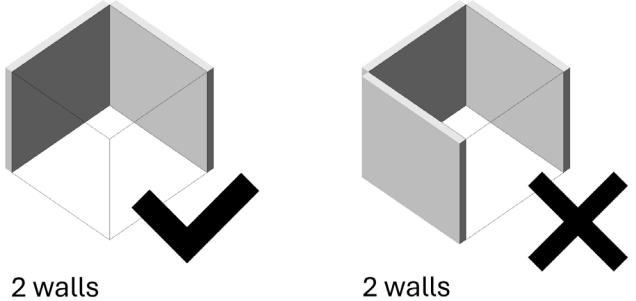


Figure 2.8: Cardinality rules

configuration processes demonstrate potential for creating an integrated configurator with an expanded design space. These engaging and user-friendly configuration processes could enhance cross-organizational collaboration.

However, a major limitation of these games is the lack of proper analytical tools to validate configurations for physical construction. This gap needs to be addressed to adapt these game-inspired principles for use in the construction industry.

2.2 Configuration process

2.2.1 The proposed configurator

As part of this project a configurator is developed that adapts a common building system in games to be able to develop designs which are validated for physical construction. The developed configurator takes the essential parts of the games' building system, the grid, and modular components. These features allow for the creation of configurations of building components (buildings). The configurator is extended with a structural analysis which validates the configurations made in the configurator.

This configurator sets itself apart from existing configurators by adopting modularity on building component level. Existing configurators mostly use larger modules and parametric systems for configuration. In this configurator, planar components such as floors and walls, and linear components such as beams and columns, are placed in a grid to create configurations. By reducing the module size, there is a greater degree of variation that can be achieved by configuring the modules. In other words, the design space is enlarged.

The second objective of the project has to do with integrating the supply chains involved in the construction process. This goal can be achieved by having building components as modules which refer to building products from manufacturers. The creation of such a database of building products and linking the database with the configurator is out of the scope for this project, but more on this topic in chapter 3.

2.2.2 Development in a game engine

The configurator was developed using Unreal Engine 5.3, a 3D computer graphics game engine by Epic Games. Game engines, including Unreal Engine, offer a comprehensive software development environment for creating games, providing essential tools for graphics, sound, and physics. Perhaps the most important tool is real-time computer graphics or real-time rendering. It focuses on producing and analysing images in real-time, meaning without delay. The game engine provides the developer with a 3D environment which the game engine's renderer visualises on your computer screen. Then, it is up to the developer to decide what should exist, what should happen, and how to interact with this 3D environment.

Unreal Engine was chosen as tool to develop the configurator because it is a software development environment for creating games. Configurators are software, and this project's proposed configurator is in many ways similar to games. Additionally, a game engine facilitates the development of user interaction, unlike other 3D graphics or CAD software like Rhino which only support parametric design. Having control over the interaction with the software allows the software to support the enlargement of the design space through a flexible and direct configuration process.

Figure 2.9: Unreal Engine 5's interface

2.2.3 Unreal Engine

Within Unreal Engine, a project encapsulates all information for a piece of software or a game through project files. These files include objects that exist in the 3D environment, such as 3D models and materials, as well as files that define the software's logic and interactions. In Unreal Engine, these logical files are called Blueprints, see figure 2.10. A Blueprint contains information about a specific part of the project. Communication between these Blueprints allows the Blueprints to interact with each other. Blueprint interaction is the most important mechanism for software development in Unreal Engine.

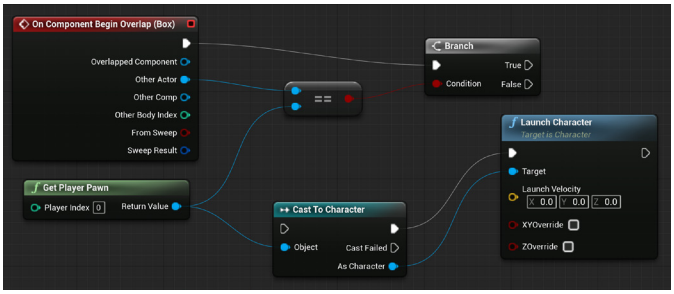


Figure 2.10: Unreal Engine 5's blueprints.

2.2.4 The configuration process

The configuration process is made up from three parts: the grid, the components, and the analysis. It is an iterative process in which each iteration usually starts with the grid and ends with the analysis.

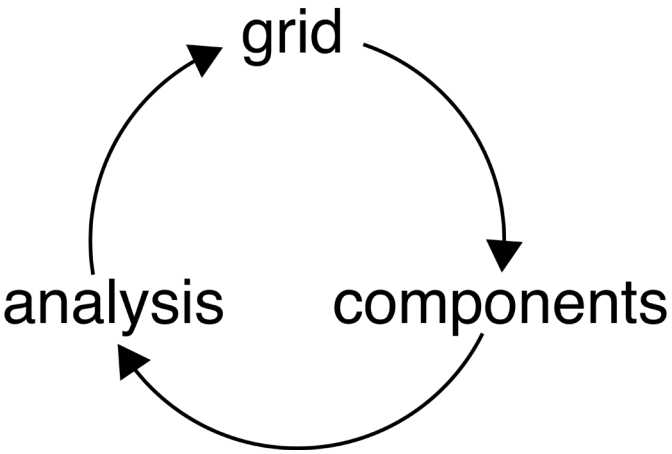


Figure 2.11: Configuration process' iteration cycle

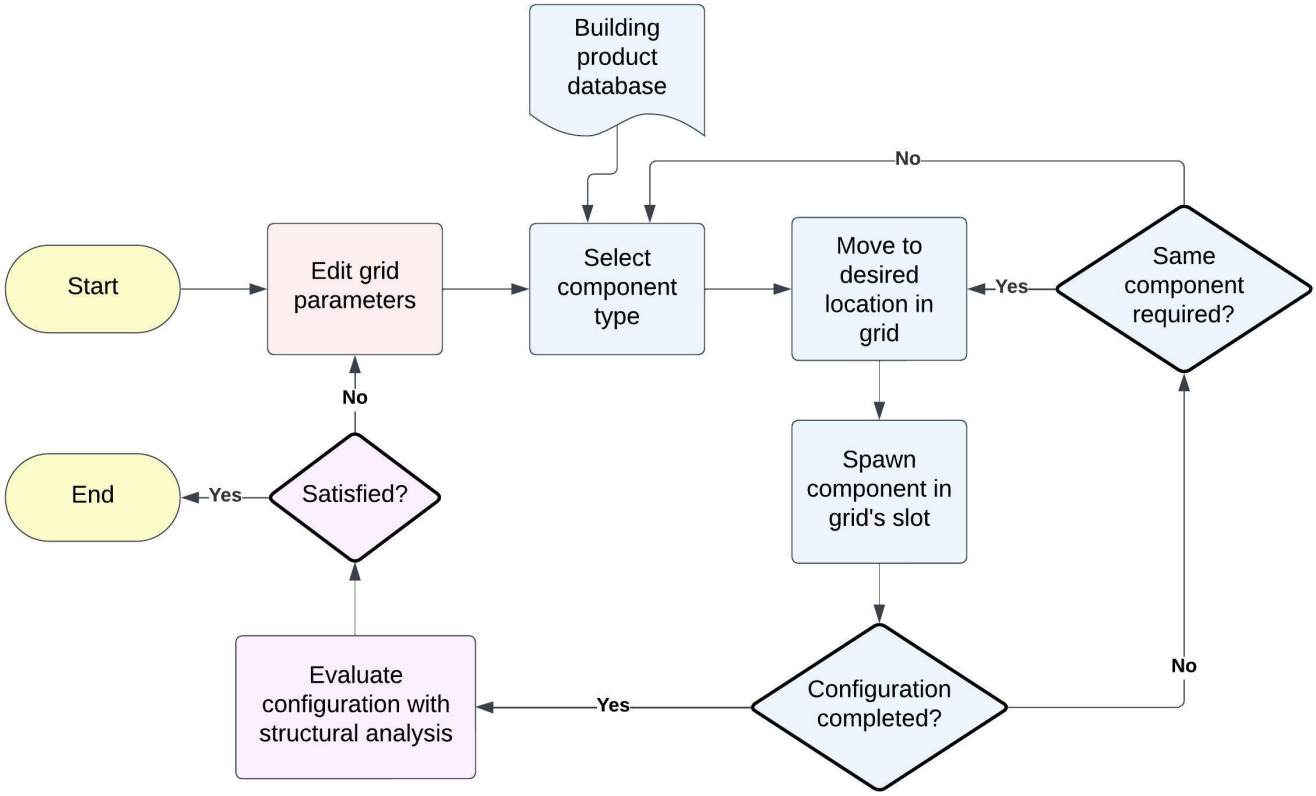


Figure 2.12: Flowchart of configuration process

2.2.5 Grid

The grid in this configurator has its own Blueprint. Fundamental to the grid is a 3D grid of vertices based on input parameters for the number and spacing of vertices in the X, Y, and Z directions. These input parameters can be adjusted to fit the project's requirements.

Figure 2.13: Configurator's grid parameters

These vertices are used to create slots, where components can be placed, see figure 2.15. The slots come in the form of planar and linear elements, faces and edges. Four vertices make up one face. Face slots are created for the foundations, floors, and walls. Next, edges are created. Only two vertices are required to form one edge. Edge slots are created for columns and beams. The grid is visualised by the use of sprites, 2D images that are displayed in the 3D environment. These sprites visualise the vertices and lines connecting the vertices.

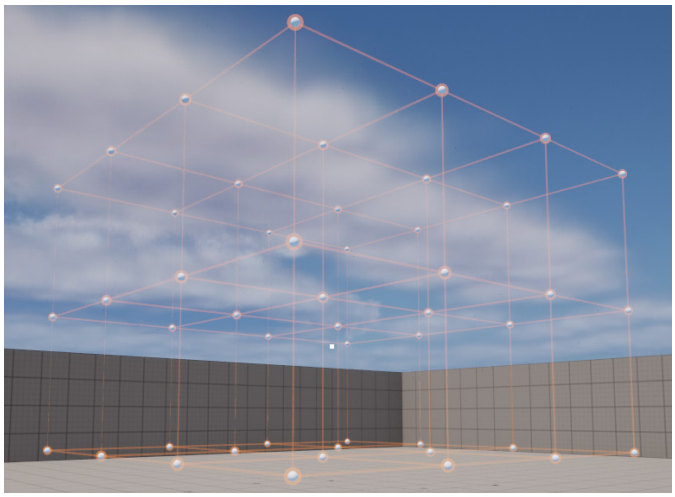


Figure 2.14: Configurator's grid

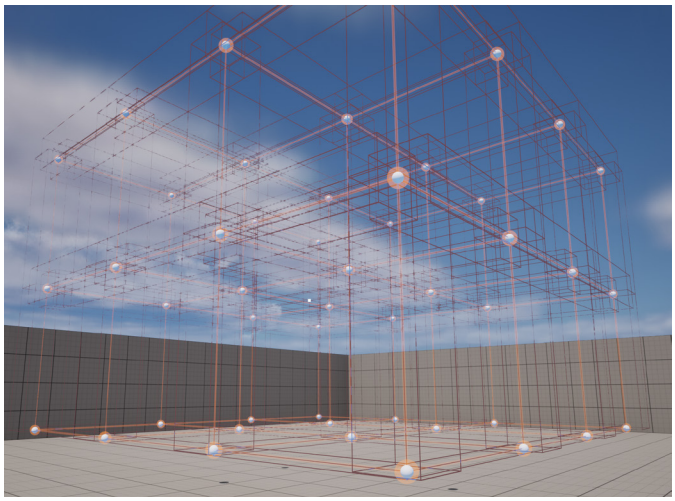


Figure 2.15: Configurator's grid slots

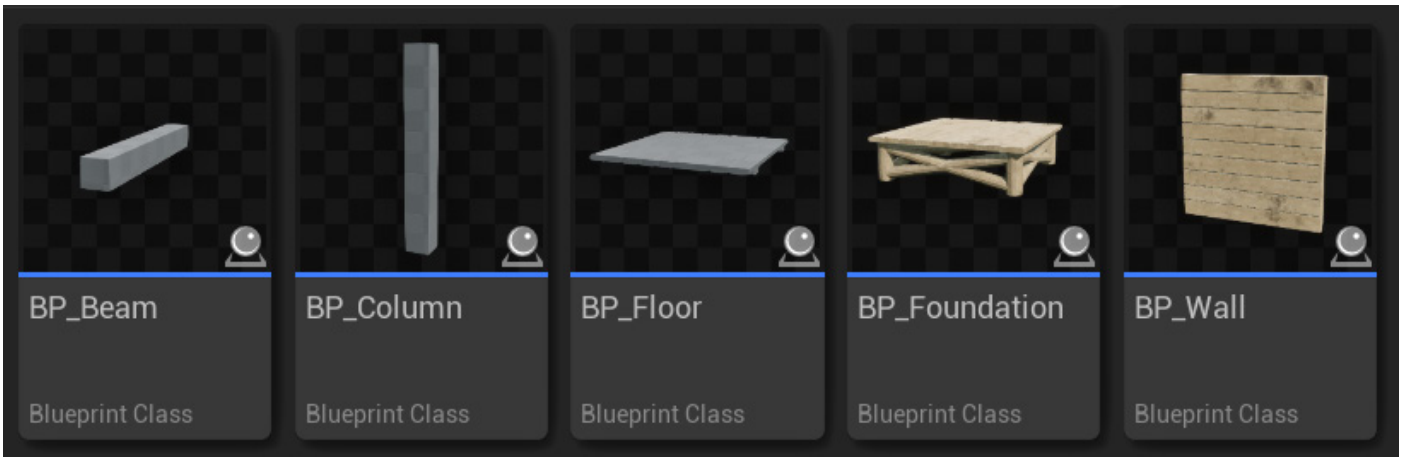


Figure 2.16: Configurator's building components

2.2.6 Components

The configurator utilizes five types of components: foundation, floor, wall, column, and beam, see figure 2.16. These components are stored in a database, and users can select the desired component type using the keyboard's number keys. Once the correct component is selected, users can navigate the grid, and the component snaps to relevant slots within the grid. A left click places the component at the selected location. By placing these components, a configuration for a structure or building is created.

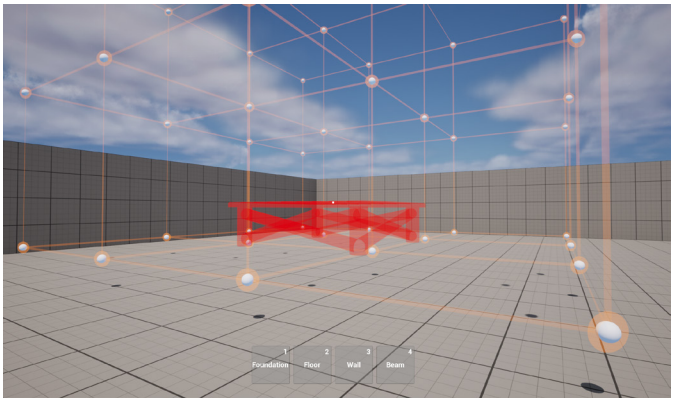


Figure 2.17: Placement of a component when slot is missed

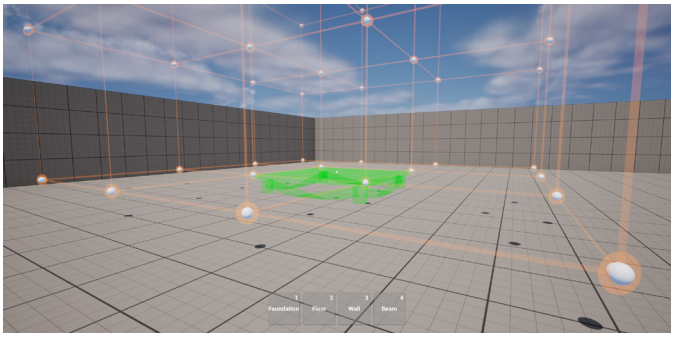


Figure 2.18: Placement of a component snapping to a slot

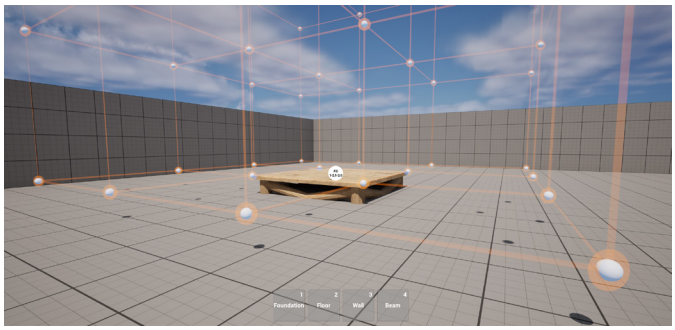


Figure 2.19: Spawning of a component at slot location

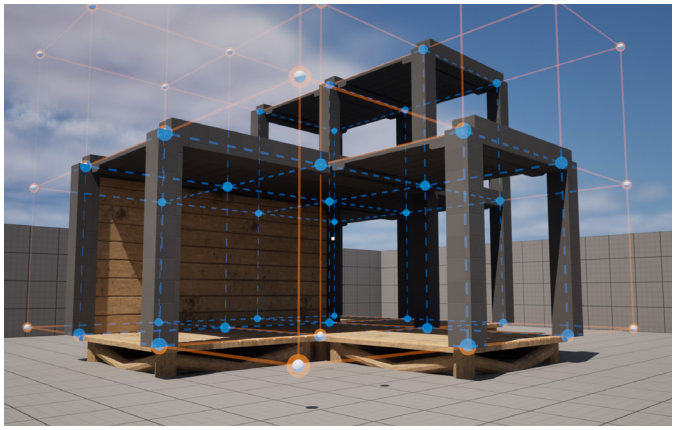


Figure 2.20: Configuration made in the configurator

2.2.7 Analysis

The structural analysis validates configurations by finding how the forces of each component flow through the entire configuration. This information is used to determine the maximum stress occurring at each component.

This begins with the configurator understanding how components are connected. When components are placed adjacent to each other, they may be considered as neighbours if they share vertices. Each component type has specific connection conditions. For example, a floor component can

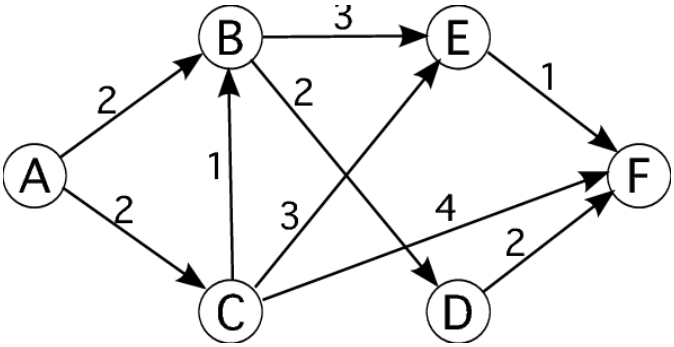


Figure 2.21: A simple flow network with directed weighted edges

only connect to beam components, and they need to share two vertices. A network graph is created where each building component is a node, and the connections between them are edges, see figure 2.21 and 2.22.

This network information is used to determine the shortest path from the foundation to each component. This process is repeated for each foundation, resulting in a list of paths for each component, with distances associated with each path. Then, the foundation is selected that gives the shortest path, see figure 2.23.

The shortest path determines the route of force travel, as per statics in structural mechanics, which states that loads should follow the shortest route to the foundations in statically determinate structures.

The next step involves passing the dead load and live load of one component to the next component in the path. Information about the path and loads is then used to calculate the stress resulting from axial forces and bending. Which in turn determines the required dimensions for each component of a given material. The components are then scaled up or down to the required dimensions.

Repeat

Information from the analysis can be used to reconsider the current configuration. Adjust the grid parameters, remove placed components, and place components to form a new configuration.

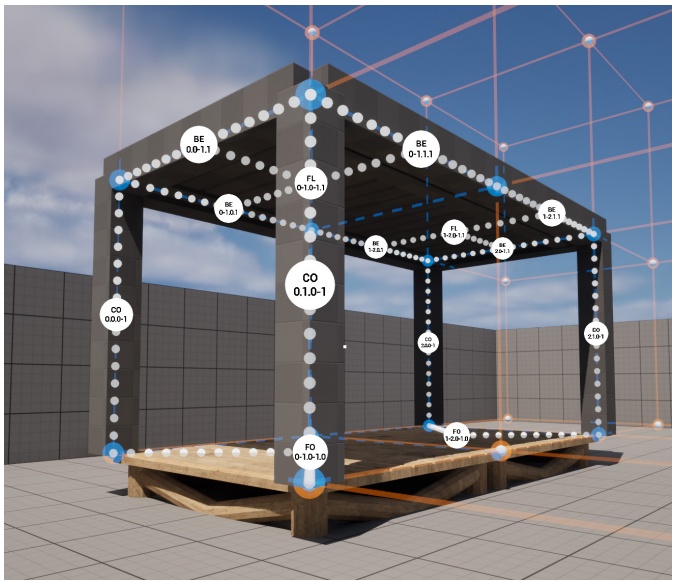


Figure 2.22: Configurator's network graph showing component connections

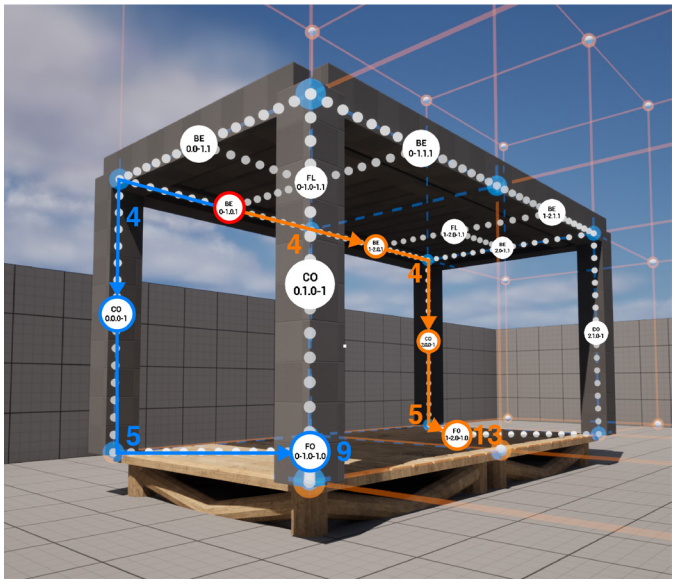


Figure 2.23: Configurator's Dijkstra shortest path algorithm comparing paths to foundations

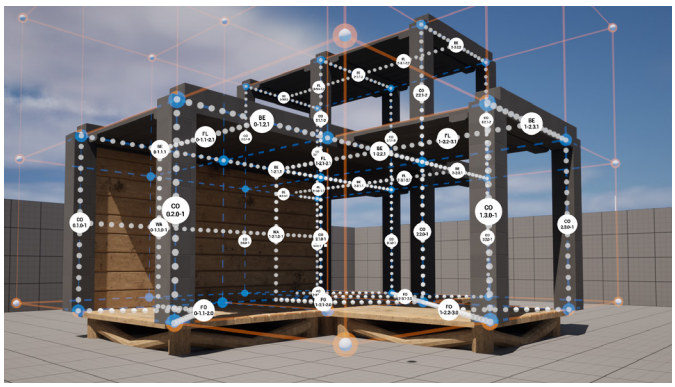


Figure 2.24: Configurator's structural analysis

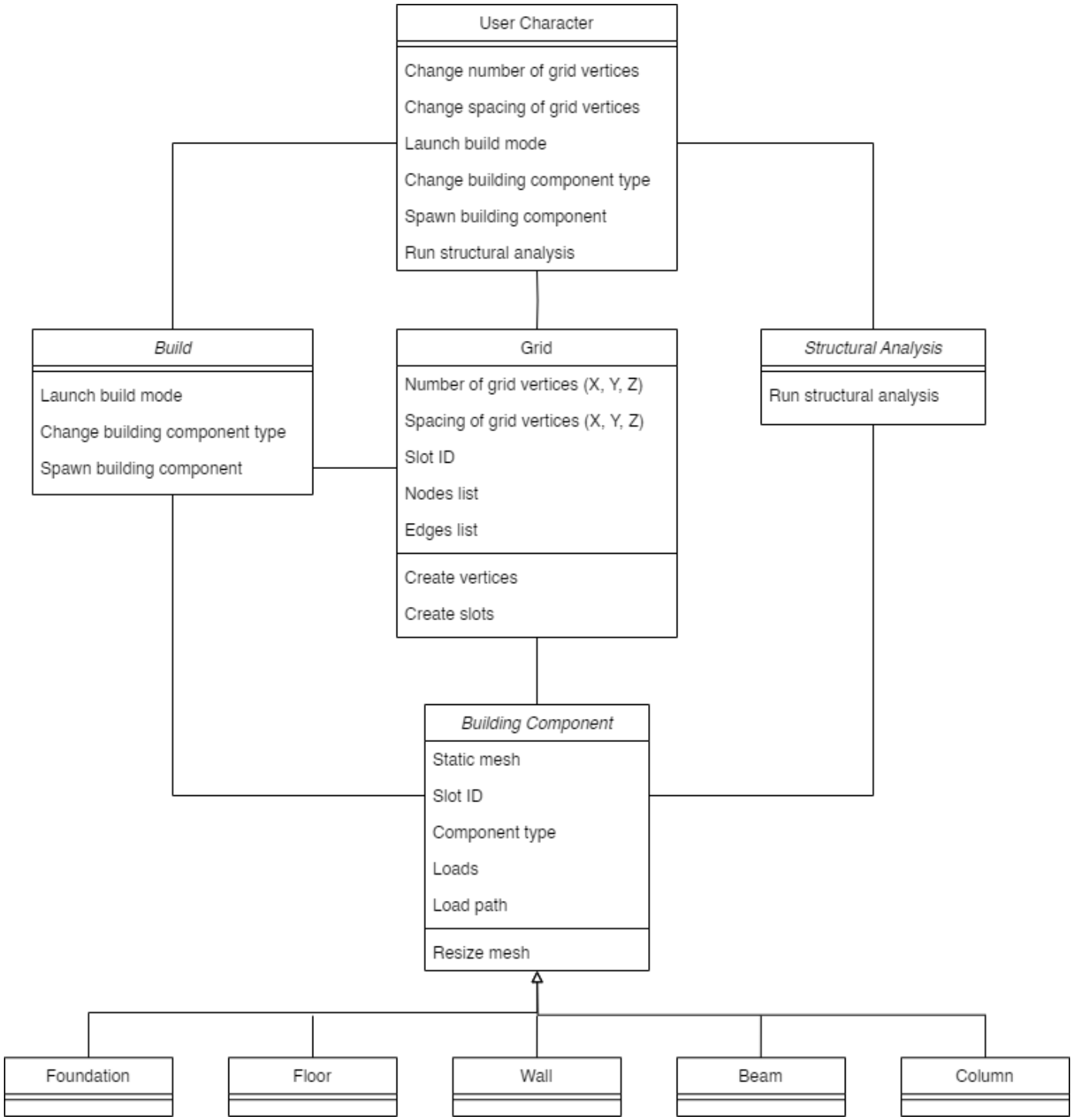


Figure 2.25: UML diagram of configurator's simplified blueprint interaction

2.3 Software development process

2.3.1 Blueprint interaction

This subchapter on the software development process is going to explain how the configurator was developed. The subchapter is divided into subsections for the configurator's most important blueprints: the grid, the build component, and the structural analysis. See figure 2.25 for a UML diagram which explains how these blueprints interact with each other. The user character is the blueprint that the user controls, through this blueprint the user initiate processes in the grid, build component, and structural analysis. Through the build component, the user can choose the component type and place the building component on the grid. When a component is placed, it exchanges information with the grid. Next, the user can run the structural

analysis, which takes information from the building component to make calculations on and changes to the building component.

2.3.2 Grid

Initially, the configurator worked without a grid. However, the decision was made to implement a grid to make the placement of building components more accurate. The grid was developed to be parametric so that it is flexible and can be adjusted to fit the requirements of the building that is to be developed inside the configurator. The grid is implemented in such a way that component placement snaps to the edges and faces of the grid, the grid slots. This implementation was chosen because it resembles the way that the structures of buildings are being designed in current practice. In current practice, structural components such as columns and beams are aligned to the grid lines and intersections of grid lines. Additionally, in the configurator, the grid slots are created from vertices so that neighbouring grid slots can be recognised based on the vertices, more on this in the section in the building process.

The grid is initialised before the software is executed. The algorithm showcased in figure 2.26 is ran every time an input parameter is adjusted. As mentioned in the previous subchapter, a 3D matrix of vertices is made based on input parameters. From the vertices, grid slots are made for the grid's faces and edges. The correct vertices are selected by choosing specific indices based on the parameter for the number of vertices in the X, Y and Z direction. For the foundation faces for example, sets of 4 indices on the bottom layer of the grid are chosen. The sets of indices are used to get sets of vertices. At the average coordinates of the set of vertices, a grid slot is spawned. The grid slot is associated with the vertices and vice versa. Next, the indices are used to make a slot ID for the grid slot. Finally, a box collision is made for the grid slot. This box collision is an invisible box which is meant to collide with the line trace. This allows for component placement, more on this in the next paragraph.

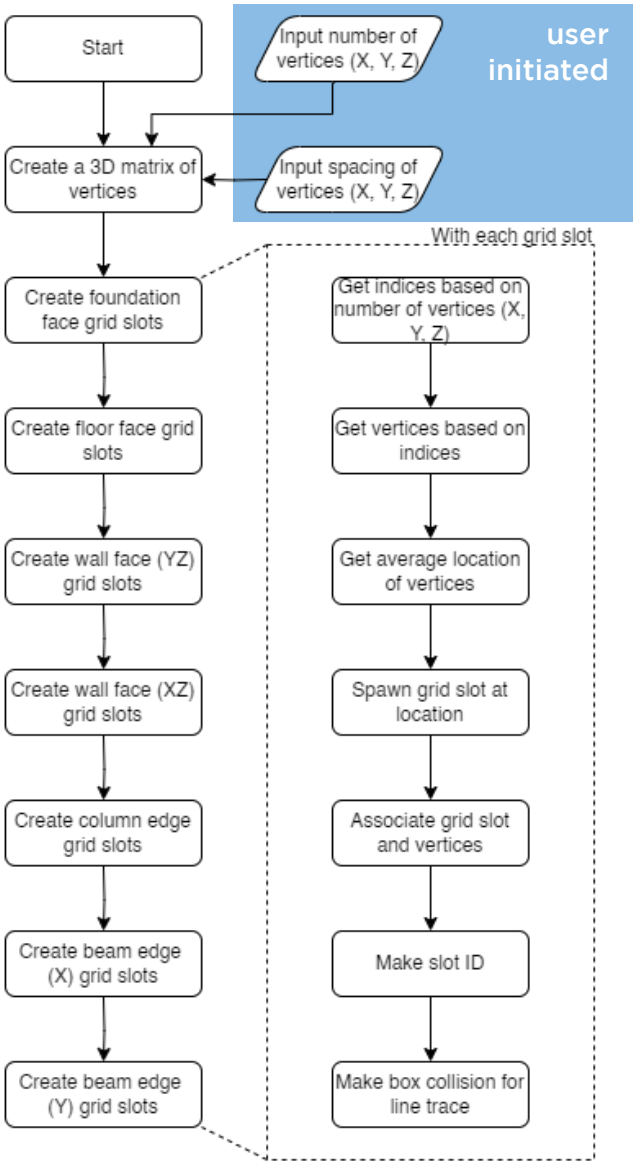


Figure 2.26: Flowchart of configurator's grid's creation process

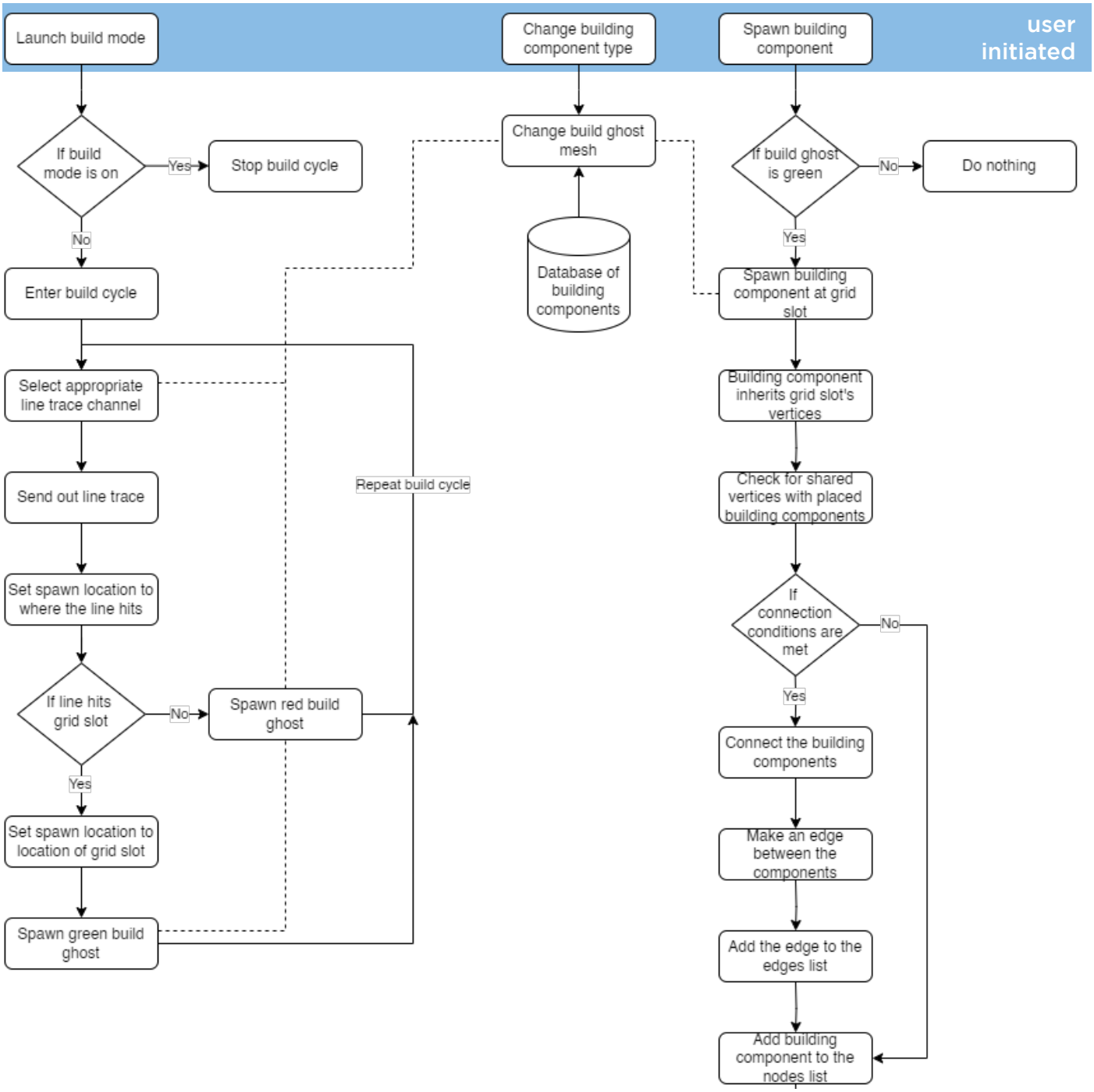


Figure 2.27: Flowchart of configurator's building process.

2.3.3 Building process

The building process concerns itself with component placement on the grid. The configurator's building process is based on building processes found in the games that served as inspiration. The reason being that those games are easy and intuitive to use, and easy to develop in Unreal Engine. The building process is set up in such a way that the user is allowed to place components in the grid wherever they want, there are as few configuration rules as possible. The configurator allows users to make unconventional configurations that are usually undesired, because sometimes unconventional configurations may actually be the best performing configurations. Instead of predefining how the user should configure components, the configurator gives feedback on the configuration and allows users to discover for themselves what kind of configurations they think are appropriate.

Additionally, by snapping components to the grid slots neighbouring components can be found. Identifying neighbouring components and checking connection conditions allows us to create a component connectivity graph. Information on component connection is required to be able to perform a structural analysis.

Process explanation

The building process starts when the 'build mode' is launched. The build mode is launched by a keyboard button press by the user. If the build mode is off, the software enters build mode, if it is already on, the software exits build mode. Once in build mode, the build cycle is entered. It starts with selecting the appropriate line trace channel, this is based on the selected building component type. Because the line trace should only collide with the box collisions of grid slots that belong to the selected building component type. This determines in which grid slots a building component can be placed. Next, the line trace is sent out. A line is sent out straight forward, wherever the line ends or hits something, the spawn location is set. This is where the build ghost will appear. If the line hits a box collision of the grid slot, the spawn location is set to that of the grid slot. This allows the components to snap to that location. A green build ghost is spawned. If the line doesn't hit a grid slot, a red build ghost is spawned. This build cycle is continuously repeated so that the location and colour of the build ghost keeps getting updated. The build cycle ends when the build mode is exited.

User can choose to change the building component type, options for component types exist in a database of building components. Changing the building component type affects the selected line trace channel, the build ghost mesh, and the building component that can be spawned.

The command for spawning a building component is executed when the user left clicks. If the build mode is on and the build ghost is green, a building component is spawned at the targeted grid slot. The building component inherits the vertices from the grid slot it occupies. Then, the building components that are already placed are checked to see if they have any shared vertices with the building component that was just spawned. If they have shared vertices, the neighbouring building components are checked to see if the connection conditions are met.

The connection conditions determine which building components can be connected to which, and how many shared vertices are required to connect them. If the connection conditions are met, the components are considered as structurally connected.

In the case of floor type building components, two vertices are required to connect the floor to a beam. One beam is insufficient for supporting a floor, but currently the software doesn't require two beams to structurally support a floor. Moreover, in the case of a unidirectional spanning floor, all vertices of the floor need to be supported by beams, and the beams should lay cross directional to the span direction. The lack of these structural considerations for floors is a current limitation.

Structurally connected components have a reference to each other. An edge is created between the building components which resembles the connection between them. The edge stores data on what building components are part of this connection, and what the centre-to-centre distance is between the components. The edge is stored in the edges list, and the building component is stored as a node in the nodes list. The edges and nodes in both lists make up a graph. This graph is a network which shows how all of the building components are connected to each other. The last step is setting the building component label. The building component uses the slot-ID from the grid slot it occupies and a type-label which refers to the building component type, to create a unique label for the building component, or node.

2.3.4 Structural analysis: load paths

Different methods of structural analysis were considered for the configurator, Finite Element Analysis, Boundary Element Analysis, and statics of structural mechanics. Finally, a method relying on statics of structural mechanics was chosen because both the Finite Element Analysis and Boundary Element Analysis methods were deemed as computationally too expensive. These methods divide the configuration and components up in voxels and determine stresses at each voxel. This information is relevant if you want to optimize the design of components, but in this case the components do not need to be altered. The only information that is necessary is the maximum stress occurring in a component, so that the strength of that component can be tested against the maximum stress to see if the component yields or not. Calculating just one stress instead of many stresses per component saves time and computer processing expenditure.

Process explanation

The user can run the structural analysis by pressing a keyboard button. The first step of the structural analysis is retrieving all the foundation type nodes in the nodes list. For each of the foundations, the Dijkstra shortest path algorithm is run. This finds the shortest path from each of the nodes to the foundation. These paths are saved to the nodes. Then, the Dijkstra algorithm is run again for the next foundation. The result is for each node, paths from that node to each of the foundations. These paths are compared to find the path(s) with the shortest distance from the node to the foundation. These path(s) are saved as the load path(s) of the node. A new graph emerges from the nodes and their load paths as edges. The edges in this graph are directed, showing how the loads are transferred to the foundation.

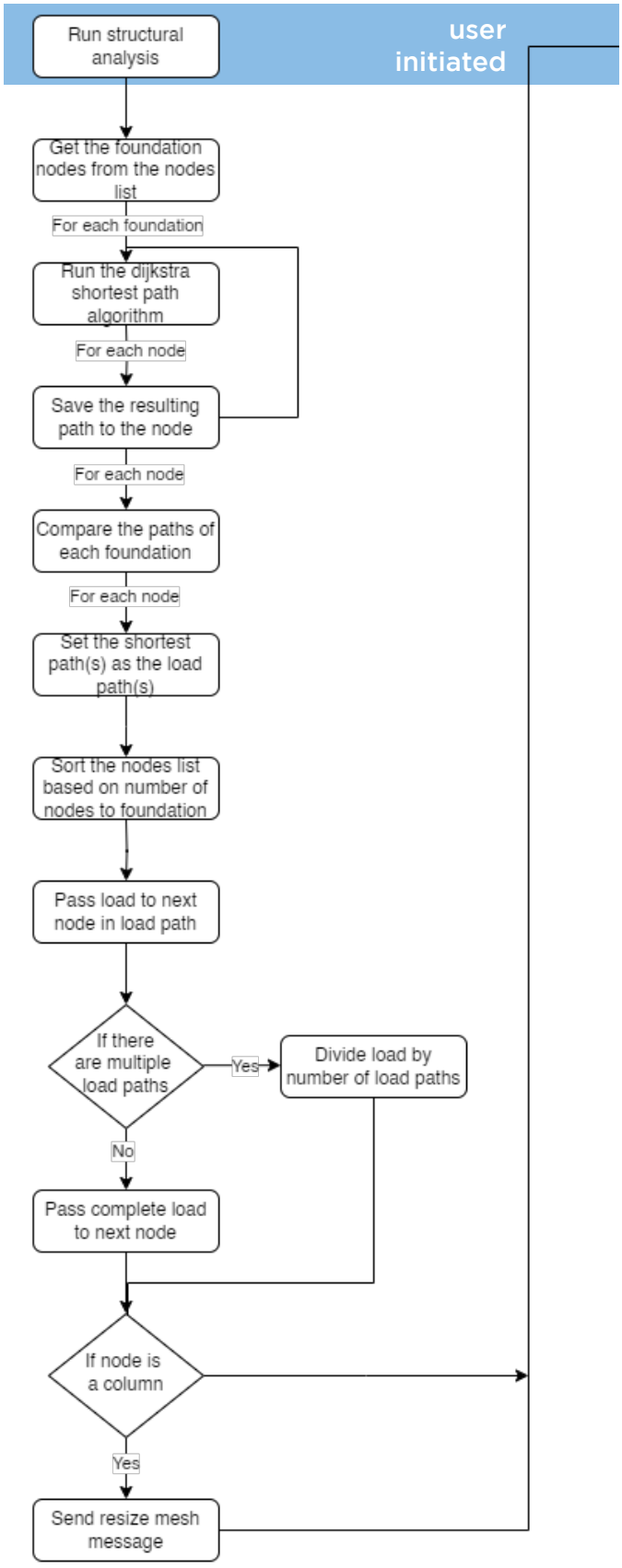


Figure 2.28: Flowchart of configurator's structural analysis, process of creating load paths.

Validation

Figure 2.30 shows an example of a configuration. Figure 2.29 shows the nodes of that configuration, distances to foundations with number of nodes to that foundation, and the next node(s) in the load path. The dimensions of the grid in this example is 400x400x400 cm. That means the node-to-node distances are multitudes of 200 cm and 282.84 cm for diagonal parts of the load path. In figure 2.29, the first node is CO_1.0.0-1, it is the most left column in figure 2.30. The print statement tells us that the distance from FO_2-3.2-3.0 is 2082.84 cm. This can be validated by manually checking the shortest path to FO_2-3.2-3.0. This path moves through nodes: CO_1.0.0-1, BE_1.0-1.1, FL_1-2.0-1.1, BE_2.0-1.1, BE_2.1-2.1, CO_2.2.0-1, FO_2-3.2-3.0. The node-to-node distances are: 400, 200, 200, 400, 400, 482.84. By adding these distances, the total distance is found to be: 2082.84. That means the shortest path algorithm calculated this distance correctly. The print statement shows the distance from CO_1.0.0-1 to FO_1-2.0-1.0 to be 482.84. The shortest path moves directly from CO_1.0.0-1 to FO_1-2.0-1.0, with the node-to-node distance of 482.84. So, the shortest path is also correctly calculated in this case. The last path is between CO_1.0.0-1 to FO_0-1.2-3.0, the print statement shows a distance of 1682.84. This shortest path moves between nodes: CO_1.0.0-1, BE_1.0-1.1, BE_1.1-2.1, CO_1.2.0-1, FO_0-1.2-3.0. The node-to-node distances are: 400, 400, 400, 482.84. Adding them up leads to a distance of 1682.84, so the algorithm calculates the distance correctly once more. Out of all the foundations, FO_1-2.0-1.0 has the shortest path, which means the next node should move towards the foundation. In this case, the next node is the foundation FO_1-2.0-1.0. This means the algorithm is working correctly.

```
CO_1.0.0-1
Distance from FO_2-3.2-3.0 = 2082.842712  N = 6
Distance from FO_1-2.0-1.0 = 482.842712  N = 1
Distance from FO_0-1.2-3.0 = 1682.842712  N = 4
Next node = FO_1-2.0-1.0
CO_1.2.0-1
Distance from FO_2-3.2-3.0 = 1682.842712  N = 5
Distance from FO_1-2.0-1.0 = 1682.842712  N = 4
Distance from FO_0-1.2-3.0 = 482.842712  N = 1
Next node = FO_0-1.2-3.0
CO_1.3.0-1
Distance from FO_2-3.2-3.0 = 2082.842712  N = 6
Distance from FO_1-2.0-1.0 = 2082.842712  N = 5
Distance from FO_0-1.2-3.0 = 482.842712  N = 1
Next node = FO_0-1.2-3.0
CO_0.3.0-1
Distance from FO_2-3.2-3.0 = 2482.842712  N = 8
Distance from FO_1-2.0-1.0 = 2482.842712  N = 7
Distance from FO_0-1.2-3.0 = 482.842712  N = 1
Next node = FO_0-1.2-3.0
CO_0.2.0-1
Distance from FO_2-3.2-3.0 = 2482.842712  N = 8
Distance from FO_1-2.0-1.0 = 2482.842712  N = 7
Distance from FO_0-1.2-3.0 = 482.842712  N = 1
Next node = FO_0-1.2-3.0
CO_2.2.0-1
Distance from FO_2-3.2-3.0 = 482.842712  N = 1
Distance from FO_1-2.0-1.0 = 1682.842712  N = 4
Distance from FO_0-1.2-3.0 = 1682.842712  N = 5
Next node = FO_2-3.2-3.0
```

Figure 2.29: Print statement of configuration's nodes, distances to foundations with number of nodes to that foundation, and the next node(s) in the load path

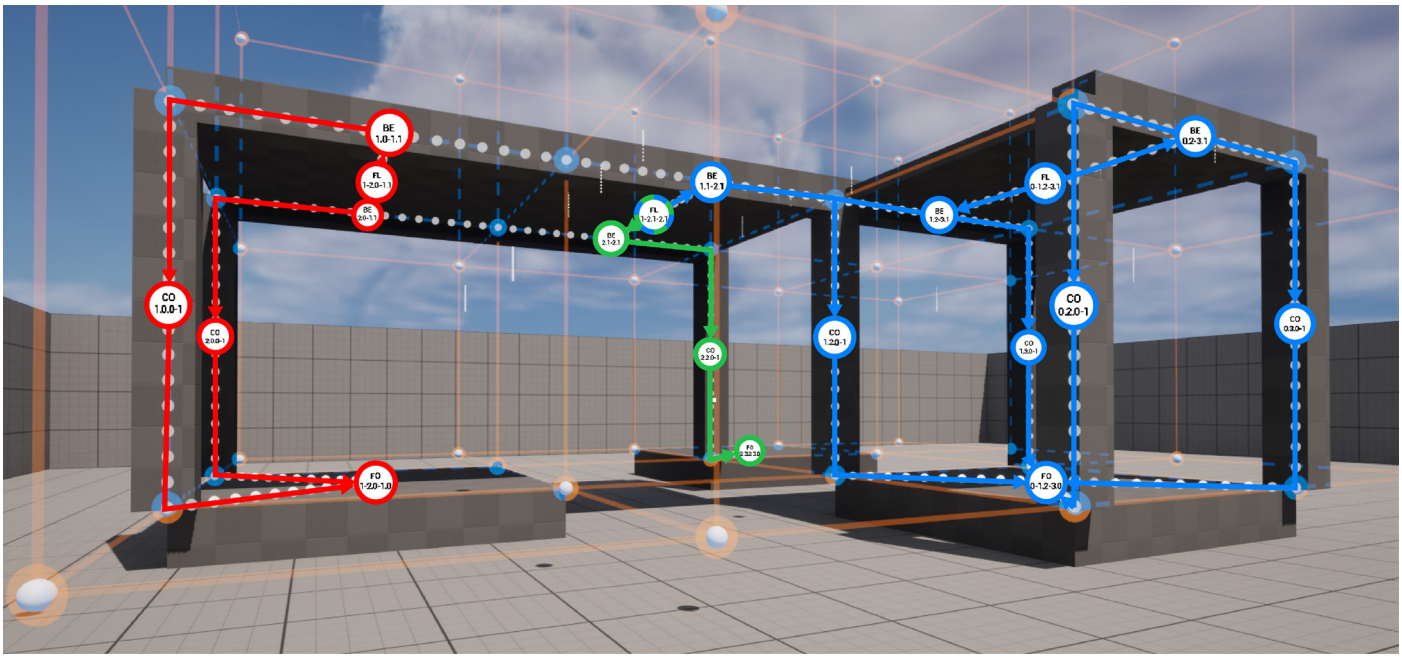


Figure 2.30: A configuration and load paths made in the configurator

Process explanation

Next, the nodes list is sorted based on the number of nodes between the concerned node and the foundation. Nodes which are close to the foundation are at the start of the list, and nodes far from the foundation are at the end of the list. Loads are passed from the concerned node to the next node in that node's load path. This happens in the reversed order from the sorted nodes list, nodes furthest from the foundation first, and closest to the foundation last. The order matters because loads are passed cumulatively, if a node passes its loads before knowing what its imposed load is, it passes an incorrect value to the next node. In the case that the node has multiple load paths leading to foundations, the passed load is divided by the number of load paths. The result is an imposed load on each of the nodes, based on the dead loads and live loads of all the nodes that came before it.

Validation

Figure 2.31 shows the imposed loads of each node resulting from the structural analysis overlayed on top of the configuration. Each floor has a dead load of 13.75 and a live load of 48, these add up to 61.75 to be passed to the next node as imposed load. A beam only has a dead load, of 2.75. A column also only has a dead load, of 2.75. The red load path has 1 floor, 2 beams, and 2 columns transferring loads to foundation FO_1-2.0-1.0. This means the total imposed load on FO_1-2.0-1.0 should be: 61.75 + (2 * 2.75) + (2 * 2.75) = 72.75. Thus, the total imposed loads that get transferred to the foundation is correct. However, there is an issue with the load transfer from the floor to the beams. All of the loads resulting from the floor gets transferred to just one beam. The reason is that the Dijkstra shortest path algorithm gives each node one next node to transfer their loads to. In the case that there are two load paths that both have the shortest path leading to the foundation, also just one load path, with just one next node is chosen. This is what is happening in the load transfer from the red floor to the beams. Floor FL_1-2.1-2.1 is connected to two load paths. Because the load paths are created in two different instances of the Dijkstra algorithm, the same issue does not occur. The floor shares its loads between the node paths as it should.

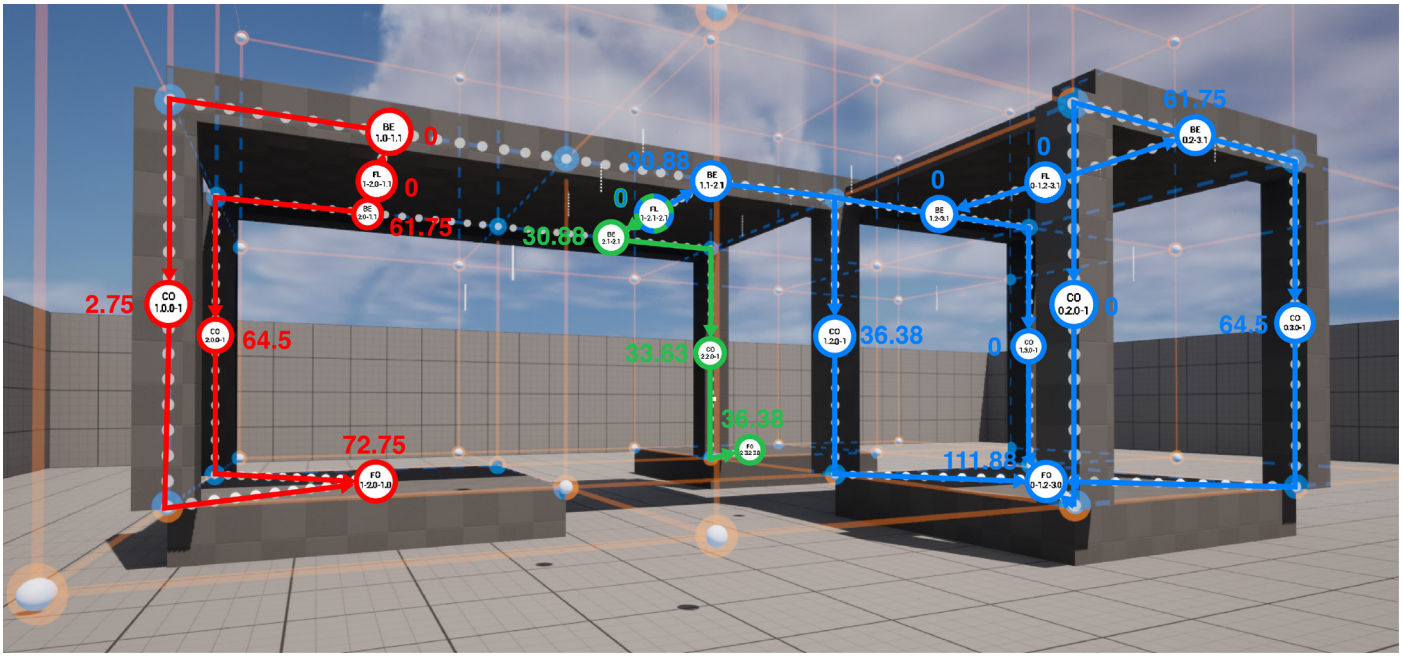


Figure 2.31: A configuration's load paths and nodes' imposed loads

2.3.5 Structural analysis: column resizing

The next step is sending a message that starts the resizing algorithm for columns. The blueprint for columns receives this message. Then it gets the columns width and depth and multiples them to find the column's profile surface area. The column's dead-, live-, and imposed load are added together. This total load is divided by the column's surface area to find the stress occurring in the column as a result of compression. From the database of building components, material properties are retrieved, in this case the compressive strength. In the configurator the characteristic strength properties of combined glulam (GL28c) are used for the calculations. In this case, the compressive strength perpendicular to the grain of GL28c, which is 2,7 N/mm^2. The column's compression stress is divided by the material's compressive strength. If the resulting factor is smaller than 1, the column can handle more compression stress before yielding. If the factor is greater than 1, the column yields and needs to be stronger to handle the compression stress. This factor is clamped so that values lower than 0,5 will be rounded up to 0,5, and values higher than 1,5 will be rounded down to 1,5. The clamping is done to prevent extreme scaling that is unrealistic. Next, the clamped factor is used to scale the column's width and depth. This is a rough estimation, because if the column is scaled, it's own weight or dead load changes. That means that columns that are too strong should be scaled down even more, and columns that are too weak should be scaled up even more.

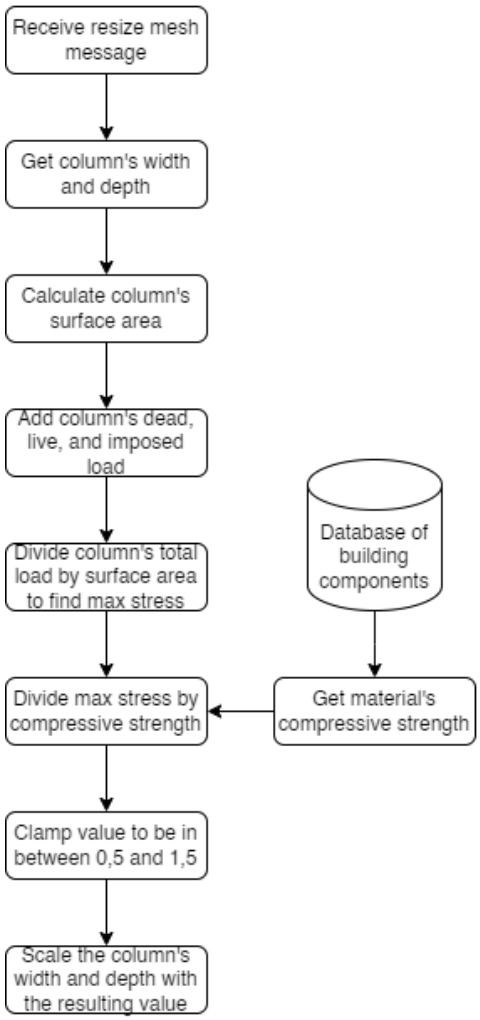


Figure 2.32: A configuration's load paths and nodes' imposed loads

Current project's configurator

Validation

Each column starts with dimensions of 500x500 mm, that equals to 250000 mm^2 of surface area. As shown in figure 2.33, the red column CO_2.0.0-1 has an imposed load of 64.5 kN, and dead load of 2.75 kN. The total load is 67.25 kN. This means the stress is equal to (67.25 * 10^3) / 250000 = 0.27 N/mm^2. The compressive strength perpendicular to the grain of GL28c is 2,7 N/mm^2. That means the scaling factor is 0.27 / 2.7 = 0.1, figure 2.34 shows the columns rescaled using this method. Thus, the width and depth of the column is reduced to the dimensions of 50x50 mm, with a surface area of 2500 mm^2. That means the stress is now 26.9 N/mm^2. The factor should now be 1, but instead is 26.9 / 2.75 = 9.78. This means the column is too weak now, so it scaled down too much. The reason is that the scaling factor is applied to both the width and depth of the column, but instead should be applied to the surface area of the column. Which means the scaling factor for the width and depth should be sqrt(0.1) = 0.32 instead of 0.1. Resulting in dimensions 158x158 mm, surface area of 24964 mm^2, stress of 2.7 N/mm^2, thus a factor of 1. This is the correct method.

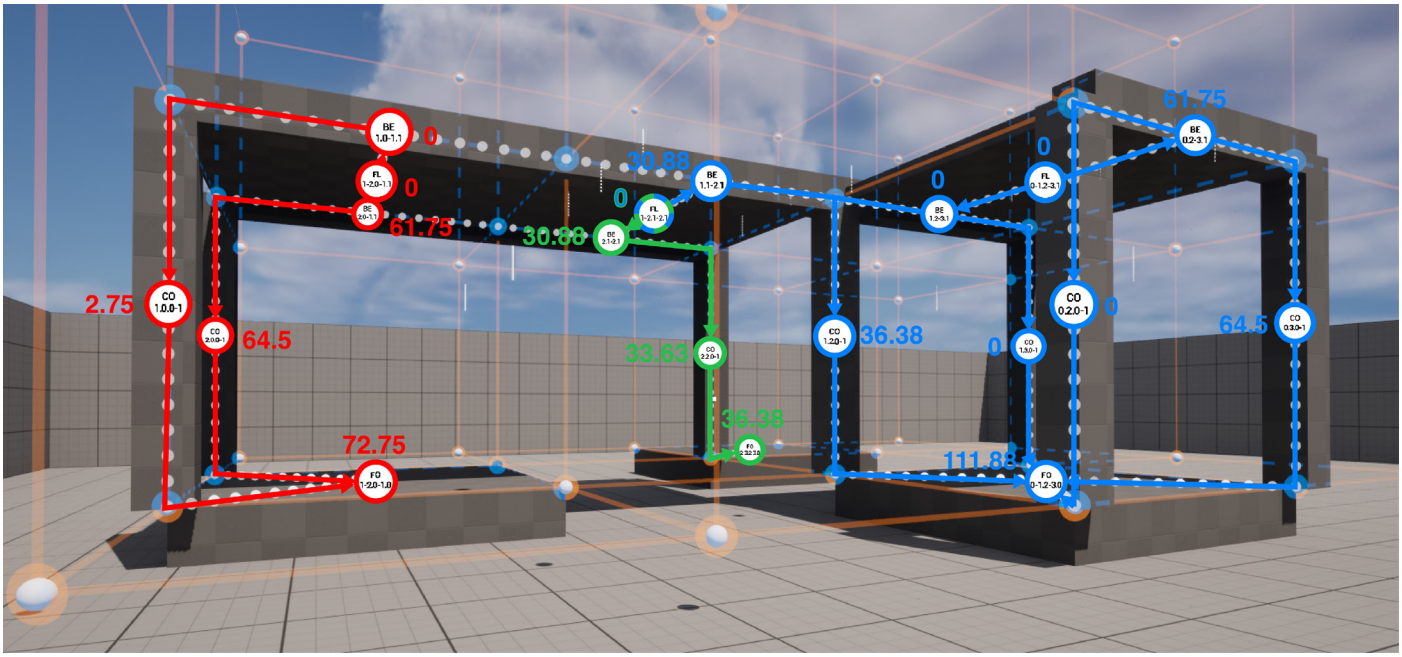


Figure 2.33: A configuration's load paths and nodes' imposed loads

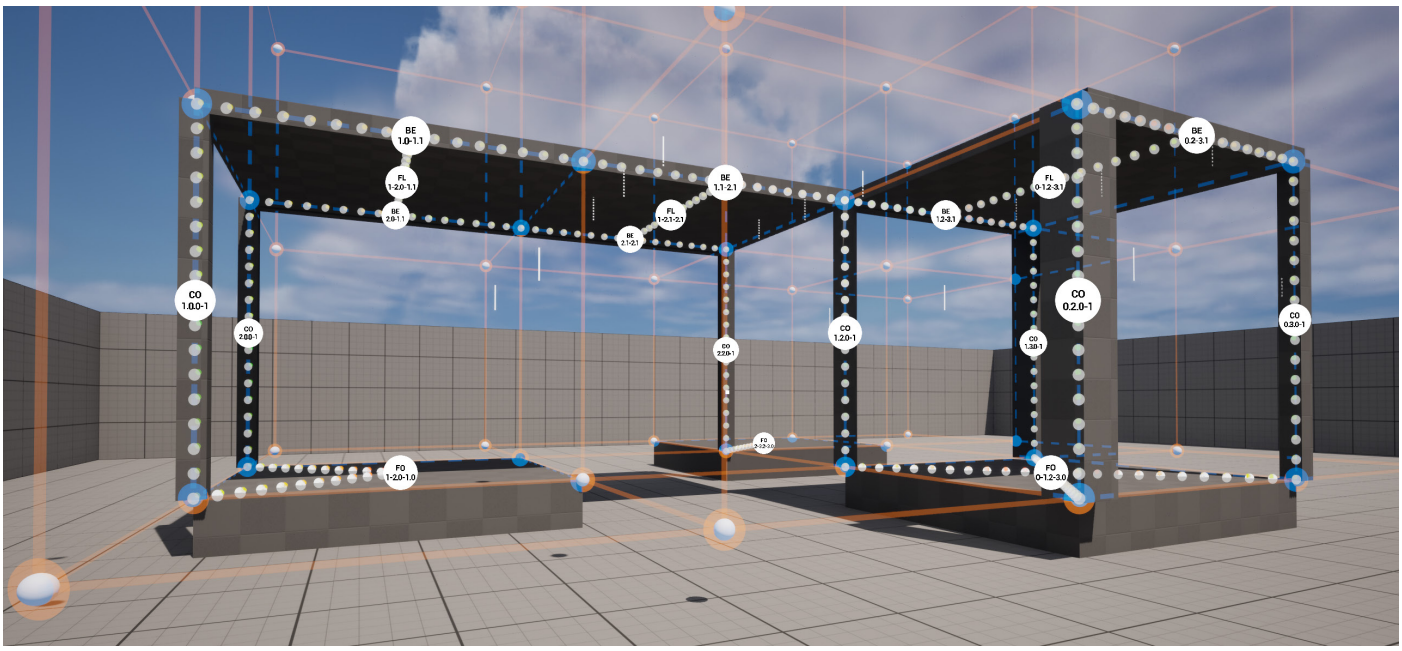


Figure 2.34: A configuration with rescaled columns and beams after running the structural analysis

2.3.6 Structural analysis: bending

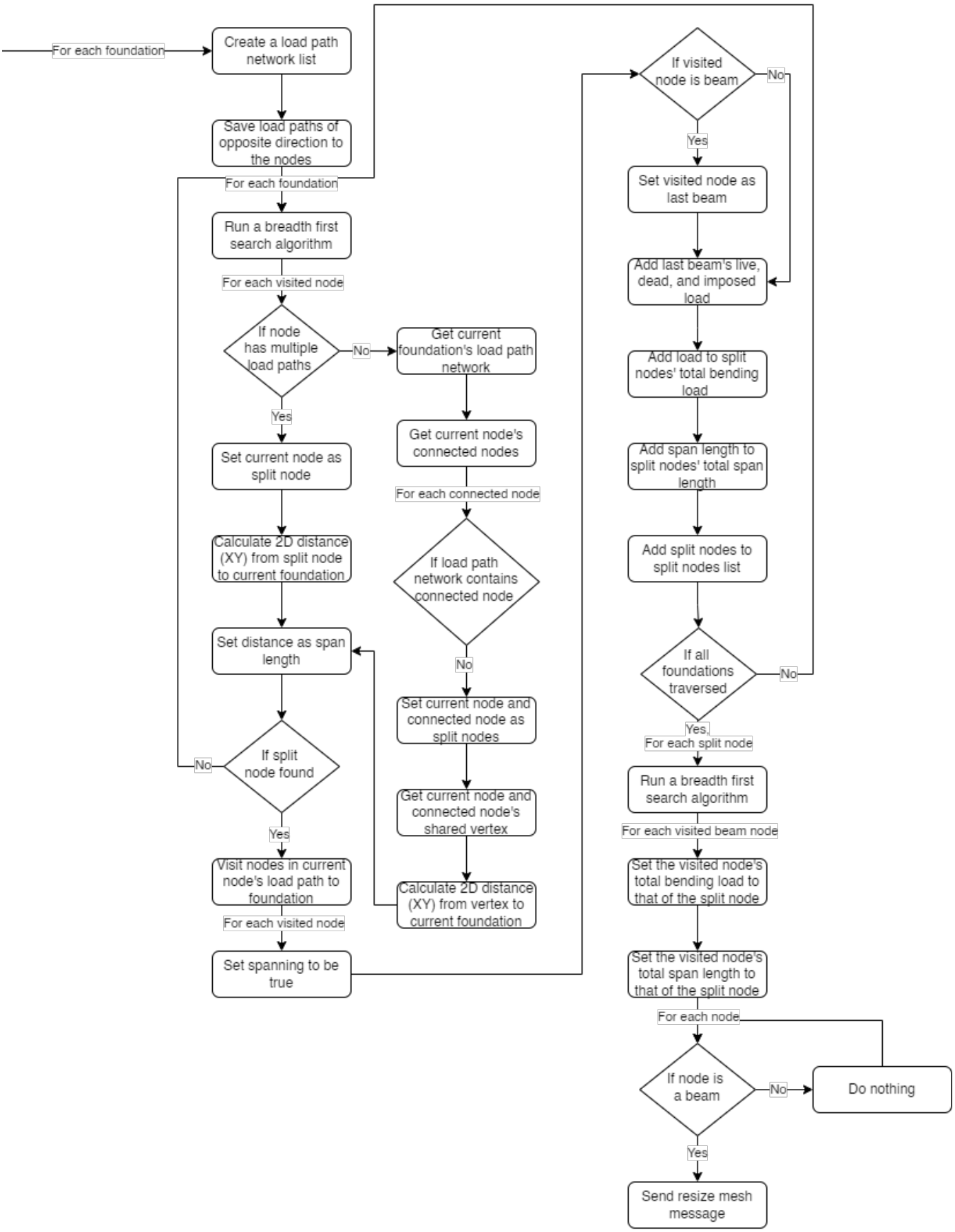


Figure 2.35: Flowchart of configurator's structural analysis, process of testing beams on bending

After resizing the columns, the structural analysis continues with the calculations on bending. The first step is creating a load path network list for each foundation. The load path network contains all the load path edges that lead to one specific foundation. Next, load paths that lead away from the foundation are saved to the nodes. This step allows the algorithm to be able to perform a graph traversal starting at the foundation, and ending at the outer nodes. That is exactly what the next step does, it uses a breadth first search algorithm for the graph traversal. This algorithm starts by visiting the foundation node, then it traverses to the neighbours leading away from the foundation and visits those nodes, next it traverses and visits their neighbours. The traversal continues until it reaches a node which has no load paths leading away from the foundation, this node marks the boundary of the network. It is an outer node which transfers its loads to the concerned foundation.

When visiting nodes during the graph traversal, operations are performed on the visited node to find out if the foundation's network is connected to another foundation's network. The nodes which connect these networks are named split nodes, because it is the point where the load paths split and go different directions, each to their foundation. Thus, the goal is to find out if the visited node is a split node. If the visited node has multiple load paths (leading to a foundation), it is set as a split node. The 2D distance in the XY plane is calculated from the centre of the split node to the foundation of the network currently being traversed. This distance is set as the span length. If the visited node does not have multiple load paths, each of the nodes connected to the visited node will be checked to see if they are part of the current foundation's network. If a connected node is not part of the current foundation's network, a split is happening in between the visited node, and this connected node. Both of the nodes are then set as split nodes, and the 2D distance in the XY plane is calculated from the nodes' shared vertex to the current foundation. This distance is then set as the span length.

If the visited node is found to be a split node, their load path will be followed to the foundation. Each node in this load path is set to be spanning. The beam node which is closest to the foundation is set as the last beam. The last beam's live, dead, and imposed load is added to the split nodes' total bending load. The total bending load is the total load on the beams in this load path. The last beam's loads cover this total load because loads of beams before it are passed to the last beam. After adding to the split nodes' total bending load, the span length is added to the split nodes' total span length.

However, this total bending load and span length is only half of the load that is part of this span, because another foundation's network is involved in the span. So, when the algorithm is run again for the other foundation's network, the split nodes' total bending load and span length will be completed by adding to them with the load and span length values relevant for this foundation's network. The split nodes now contain the correct total bending load and span length, the other nodes do not. So, another breadth first search algorithm is performed for each split node. The algorithm starts at a split node and moves down through the load path to the foundation. Each node visited during this graph traversal gets their total bending load and span length set to that of the split node the load path starts at.

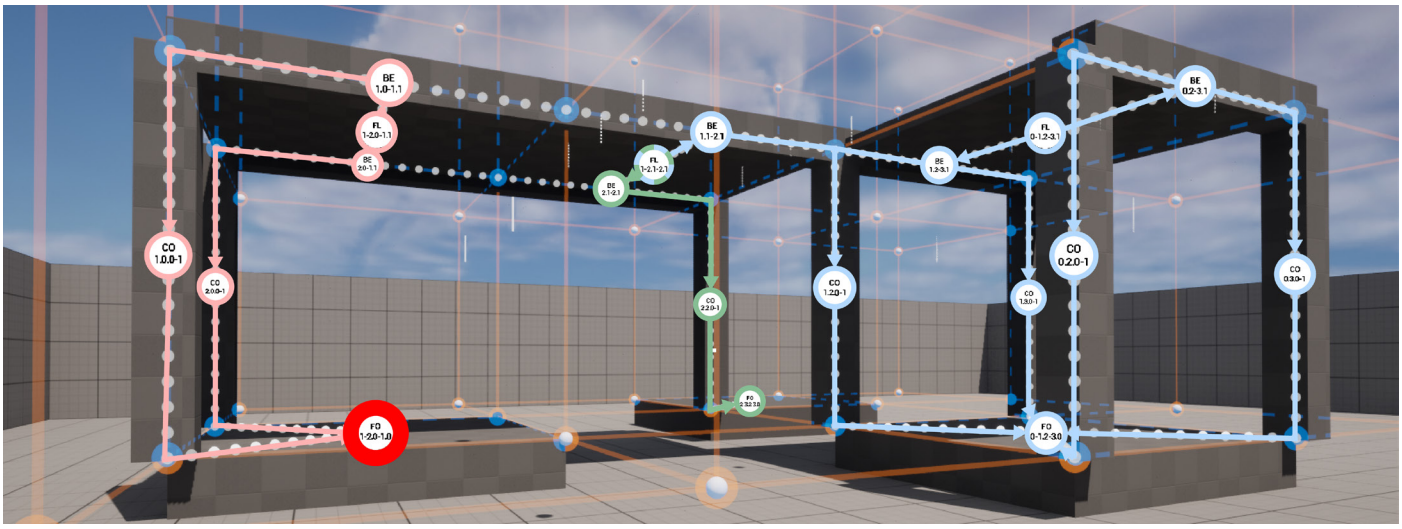


Figure 2.36: Breadth first search graph traversal
Traversal of red load path starts by visiting the foundation.

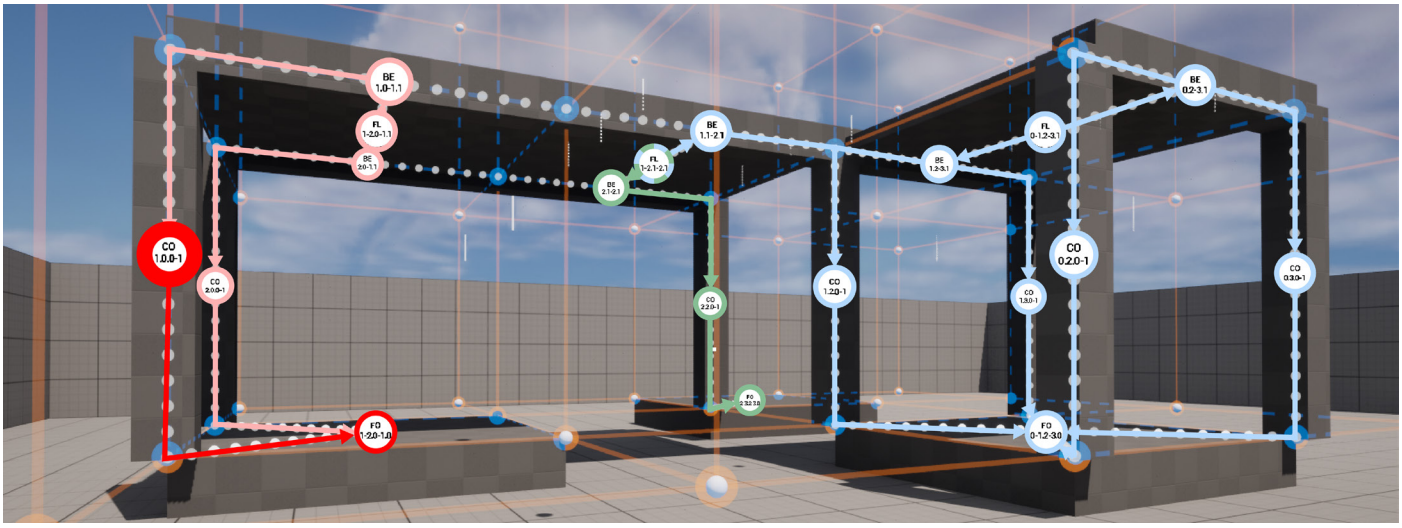


Figure 2.37: Breadth first search graph traversal
The foundation's first neighbouring node is visited.

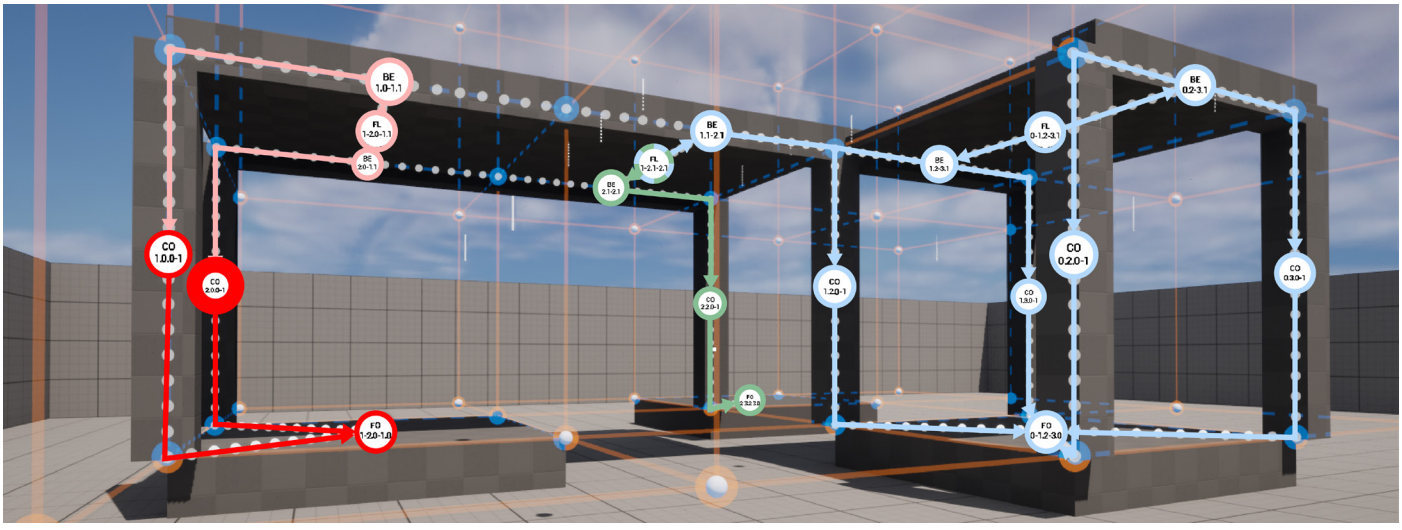


Figure 2.38: Breadth first search graph traversal
The foundation's second neighbouring node is visited.

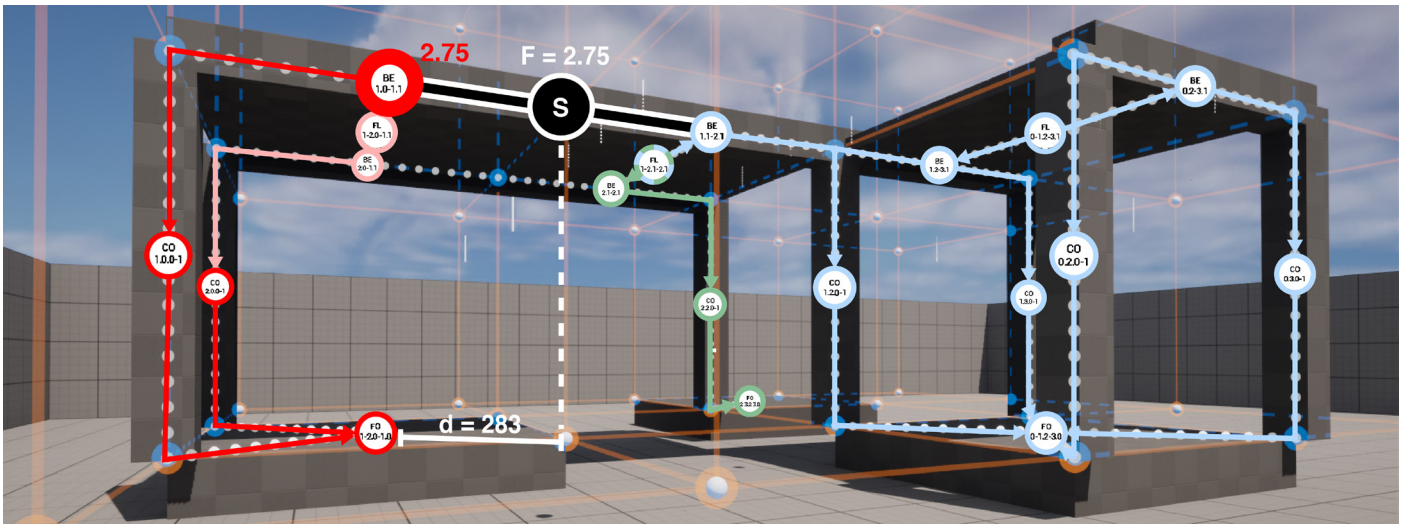


Figure 2.39: Breadth first search graph traversal

The first columns neighbour is visited. This node is identified as a split node, the split happens in between BE_1.0-1.1 and BE_1.1-2.1. For the nodes part of this split, the span length is set as the XY distance from the split to the foundation. The bending load is set as the total load acting on the last beam BE_1.0-1.1.

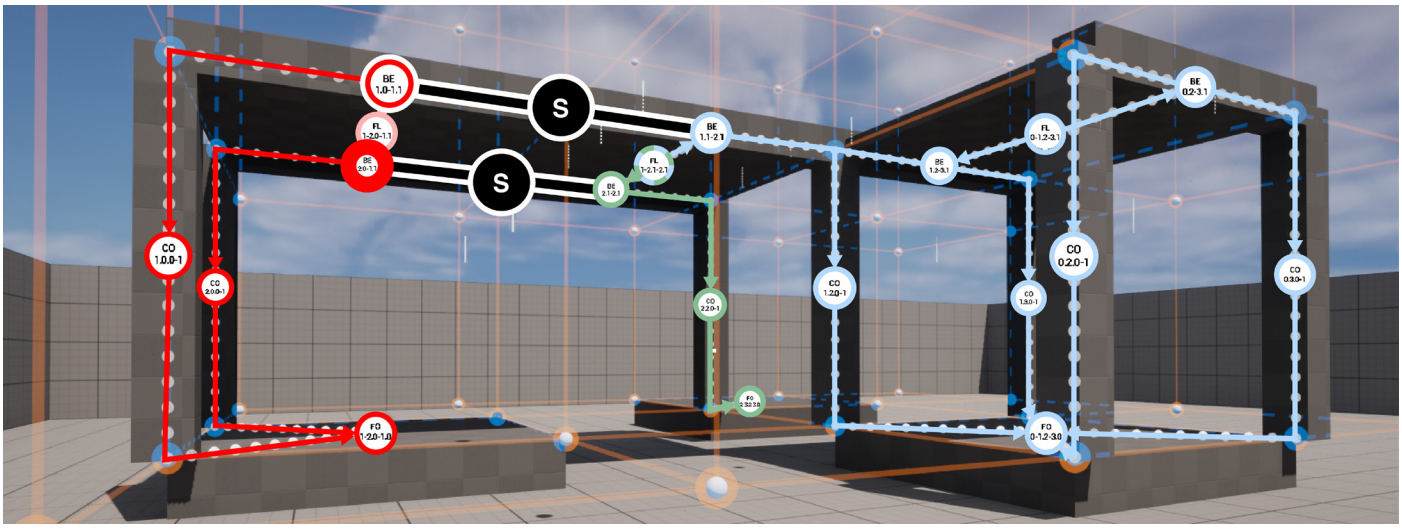


Figure 2.40: Breadth first search graph traversal

The second columns neighbour is visited. This node is also identified as a split node. For the nodes part of this split, the span length and bending load is set.

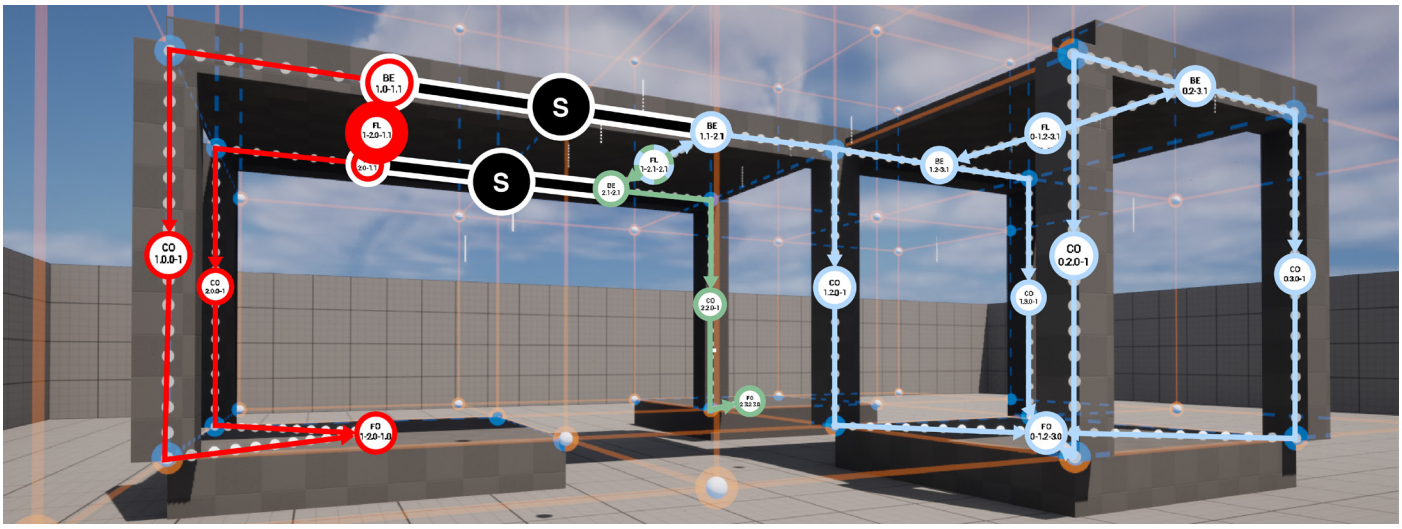


Figure 2.41: Breadth first search graph traversal

The first beams neighbour is visited. All nodes that are part of the load path network of the red foundation have been visited.

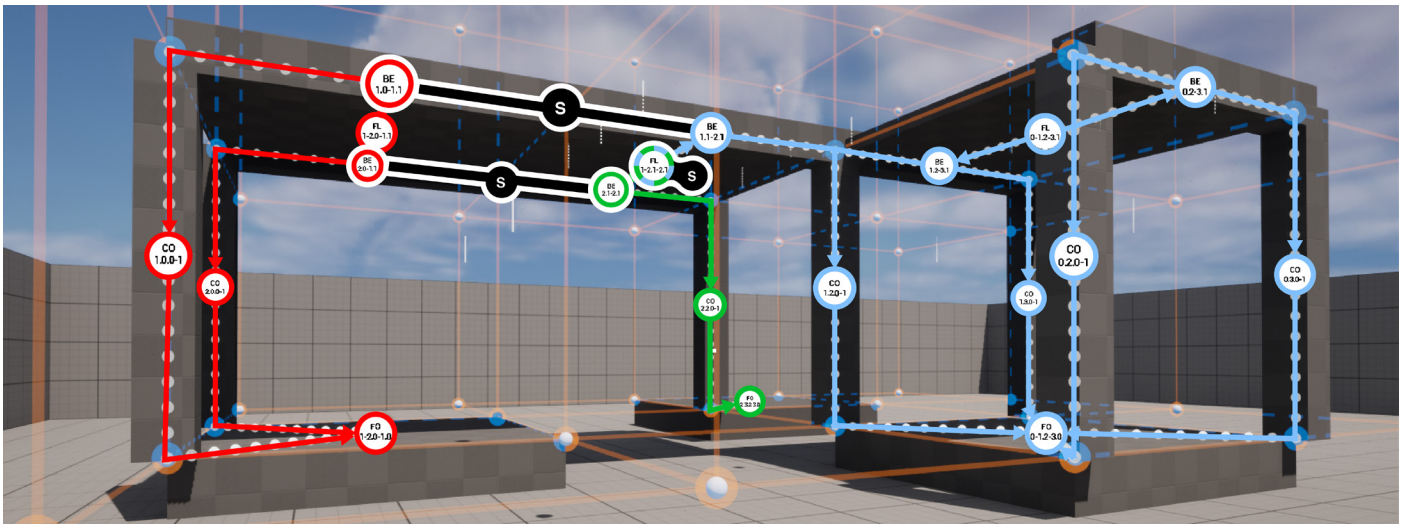


Figure 2.42: Breadth first search graph traversal

The breadth first search traversal is applied to the green foundation's network. The split node at BE_2.1-2.1 is found, span length and bending load is added to the nodes part of this split. Another split node is found at FL_1-2.1-2.1, this node has two load paths.

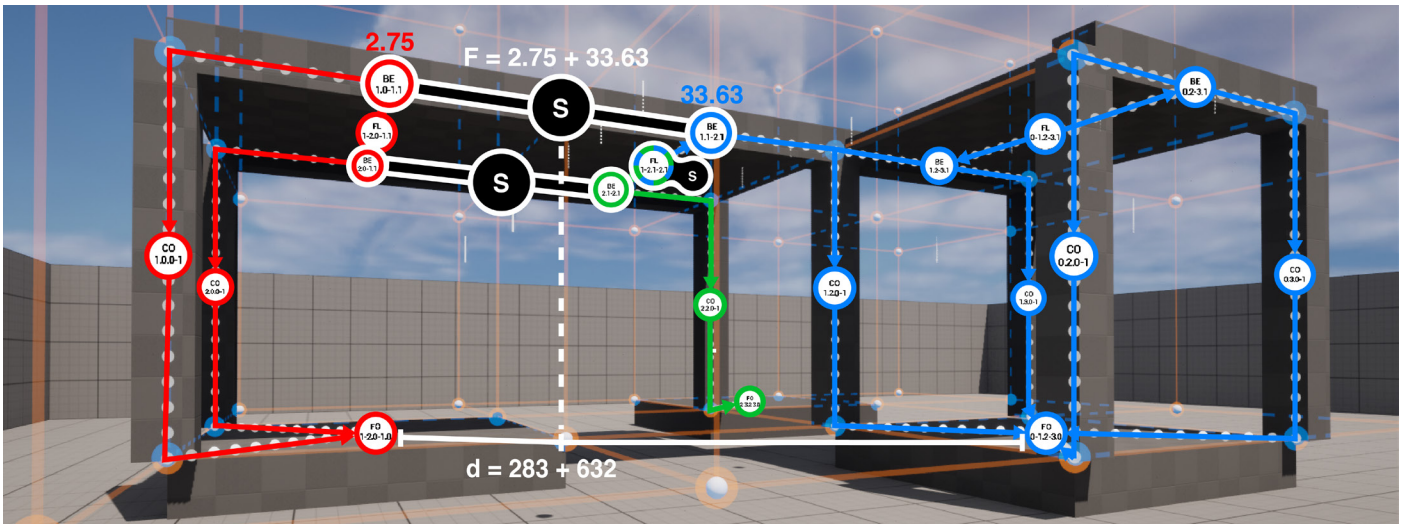


Figure 2.43: Breadth first search graph traversal

The breadth first search traversal is applied to the blue foundation's network. The split node at BE_1.1-2.1 is found, span length and bending load is added to the nodes part of this split.

2.3.7 Structural analysis: beam resizing

If the visited node is a beam, a message is sent to start the algorithm that resizes beams. When the blueprint for beams receives this message, the first step is getting the beam's width and height. This is used to calculate the beam's section moment of inertia. Next, the total bending load is retrieved for the span of which the beam is a part of. This span's total length is also retrieved. Both the bending load and span length are used to calculate the span's maximum moment. Then, the distance from the beam's neutral axis to the height of interest is determined. In this case, we are interested in finding the maximum stress, which occurs at the top and bottom of the beam's profile, at y=h/2. The span's maximum moment, height of interest, and the beam's section moment of inertia are used to calculate maximum stress occurring in the beam as a result of bending (My/I). From the database of building components, material properties are retrieved, in this case the bending strength. In the configurator the characteristic strength properties of combined glulam (GL28c) are used for the calculations. In this case, the bending strength parallel to the grain of GL28c, which is 28 N/mm^2. The maximum occurring bending stress is divided by the material's bending strength. Resulting factors greater than 1 are too weak, factors smaller than 1 can handle higher stresses. This factor is clamped so that values lower than 0,5 will be rounded up to 0,5, and values higher than 1,5 will be rounded down to 1,5. The clamping is done to prevent extreme scaling that is unrealistic. Next, the clamped factor is used to scale the beam's width and height. Same as with the resizing of the columns, the effect of scaling dimensions on the dead load of the component is not take into consideration.

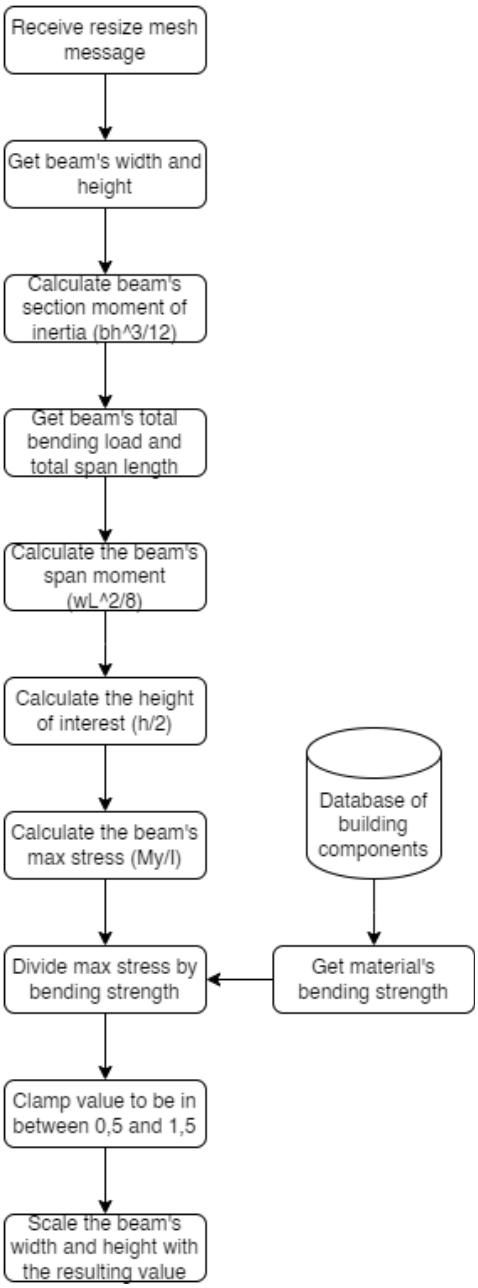


Figure 2.44: Flowchart of configurator's structural analysis, process of resizing beams

Validation

Each beam starts with profile dimensions of 500x500 mm, that equals to a section moment of inertia of: (500 * 500^3) / 12 = 5.21 * 10^9 mm^4. The red beam BE_1.0-1.1 has a total bending load of: 2.75 + 33.63 = 36.38 kN, see figure 2.45. Its total span length is: 282.84 + 632.46 = 915.3 cm. The total span length and total bending load give us a bending moment of: (36380 * 9.15) / 8 = 41609.6 Nm, or 4.16 * 10^7 Nmm. The height of interest is: 500 / 2 = 250 mm. The beam's max bending stress is: (4.16 * 10^7 * 250) / (5.21 * 10^9) = 2.00 N/mm^2. The bending strength parallel to the grain of GL28c is 28 N/mm^2. That means the scaling factor is: 2 / 28 = 0.07. So, the algorithm downscales the width and height of the beams to 35x35mm, this profile has a section moment of inertia of: 1.25 * 10^5 mm4. The resulting bending stress is: 5821.57 N/mm^2. So, the factor is 5821.57 / 28 = 207.91. This means that the 35x35 mm profile is way too slender. The problem is similar to that of the column resizing. The factor does not apply to the width and height of the beam's profile, it applies to the maximum allowable stress. The maximum allowable stress is 28 N/mm^2, this means that: 28 = (4.16 * 10^7 * (a / 2) / (a^4 / 12). With some algebra it is found that: a = 207.31 mm. Thus, the scaling factor should be 207.31 / 500 = 0.41. When we test these dimensions of 207x207 mm, its section moment of inertia is 1.53 * 108 mm^4. The beam's maximum bending stress is 28 N/mm^4. So, this method is correct.

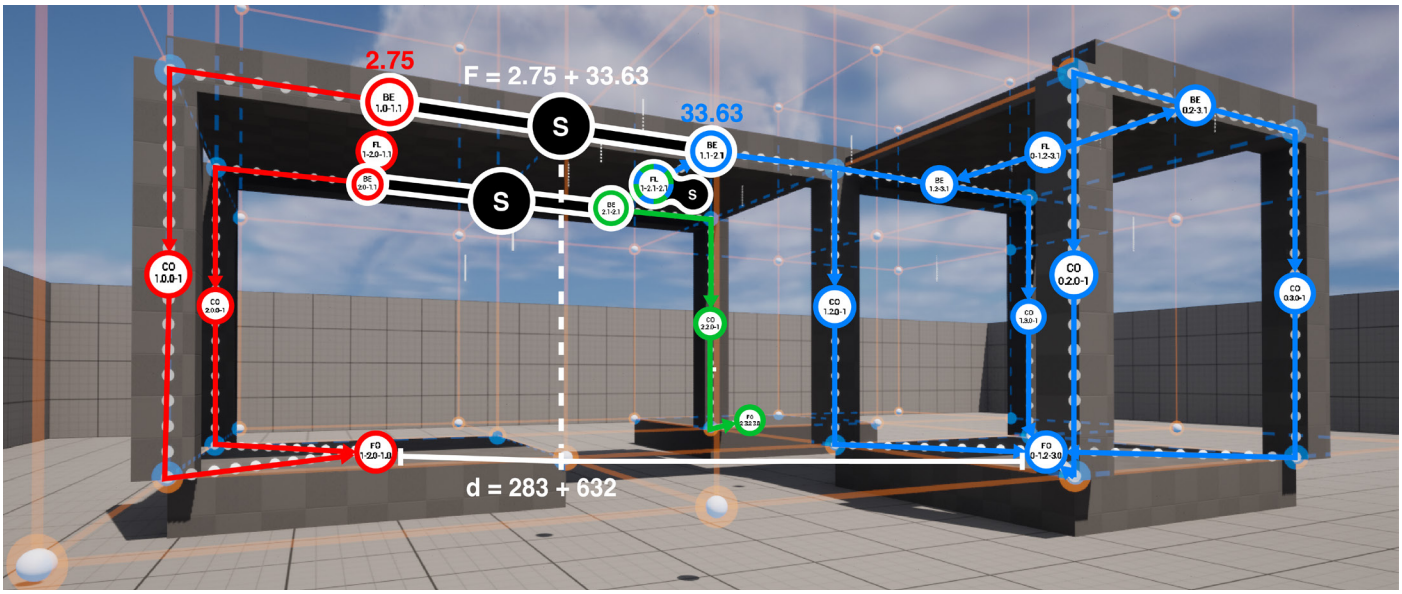


Figure 2.45: Breadth first search graph traversal

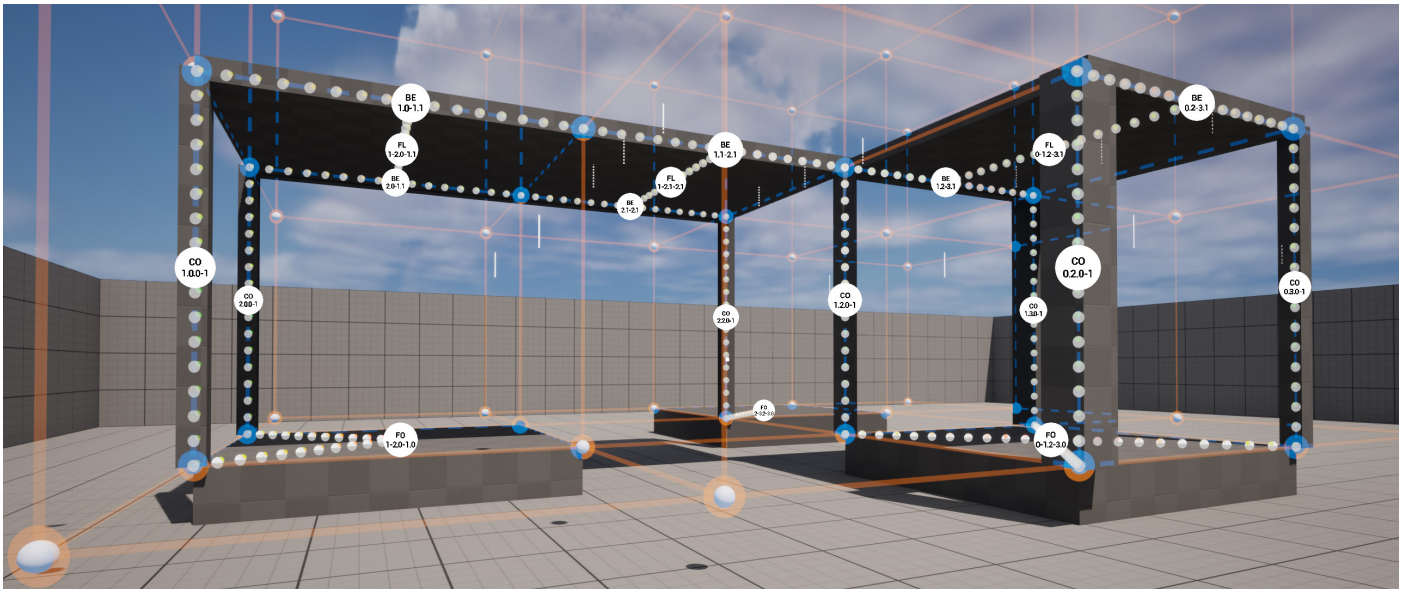


Figure 2.46: A configuration with rescaled columns and beams after running the structural analysis

3. Future development

The research objective is to create an integrated configurator with an extended design space. The developed configurator from the previous chapter has some of the core features implemented but requires refining and additional development before the research objective is fully achieved. This chapter explores potential advancements in order to realise this, detailing where future developments could lead.

3.1 Implementing the building product database

3.1.1 Building product database's revelance

The developed configurator has the infrastructure that allows the implementation of a building product database. It uses building component types to build with. Taking the configurator to the next level by building with digital assets of real building products allows the configurator to integrate across construction phases. Utilisation of building products can ensure product manufacturability when the digital assets include manufacturing constraints. Decoupling the database of building products from the BIM software ensures that manufacturing stays possible because only the manufacturer of that product can edit the BIM family. Ensured product manufacturability limits design issues which require reorganisation efforts. Additionally, the reuse of building product models saves time compared to building projects from scratch as is currently common practice. Both aspects ultimately save time and consequently money.

3.1.2 Difference from existing BIM databases

The building product database provides a common environment where a diverse range of building products from different manufacturers are presented. The building products come in the form of BIM families. Many manufacturers already have BIM families available. The added value of the database to the user of the BIM software is that they have one place where many building products can be viewed and compared. It will become easier to implement appropriate building products and their BIM families in projects. For the manufacturers it means their products get more exposure and a good chance to be included in the design of new buildings, which may boost their sales. These BIM families range from structural elements, façades, and roofs, to mechanical, electrical and plumbing systems. Everything that can exist in a building can be found in the database. BIM databases are currently underutilized, but there are some existing databases. Some of which even have a plugin for BIM software which allows integration within the software. What sets this database apart from existing databases is the interactability between the database and the configurator. Building products are recommended on the specific use case within the project's design. The way this happens will be detailed in the configuration process section.

3.1.3 Creating entries in the database

The buildings products in the database are organised into building component types such as those in the developed configurator: foundations, floors, walls, columns, and beams. Each building product has data on connection compatibility, if two building products are compatible with the same type of connection then they are also compatible with each other. Another type of data is about the products geometric properties. For the geometric properties an important distinction is between static and dynamic geometry. Static geometry does not have the ability to change, its dimensions are set and unmodifiable. In contrast, dynamic components have the ability to adjust their shape and/or size according to grid or configuration conditions. The size of the grid slot will determine what building products can fit in that grid slot. Dynamic components will fit in a grid slot more often than static components, but static components may be cheaper and more economical to produce. BIM families already have the potential to be dynamic and include module size variations. BIM families provide module size variations through parameters on the height, width, and length of the component.

Another point of data is on the materials that are used in the building product. Additionally, each building product has physical properties, of which the mass is an important one. Another type of physical properties refer to the structural performance

	Milieueffectcategorie	Equivalent eenheid	Methode	
Emissies	Klimaatsverandering – GWP 100 j.	CO_2 eq	CML2-baseline	Nationale Milieudatabase
	Aantasting ozonlaag – ODP	CFK-11 eq	CML2-baseline	
	Humane toxiciteit – http	1,4-DCB eq	CML2-baseline	
	Zoetwater aquatische ecotoxiciteit – FAETP	1,4-DCB eq	CML2-baseline	
	Terrestrische ecotoxiciteit – TETP	1,4-DCB eq	CML2-baseline	
	Fotochemische oxydantvorming – POCP	C_2H_2 eq	CML2-baseline	
	Verzuring – AP	SO_2 eq	CML2-baseline	
	Vermesting – EP	PO_4 eq	CML2-baseline	
Uitputting grondstoffen	Uitputting abiotische grondstoffen – ADP	Sb eq	CML2-baseline	
	Uitputting fossiele energiedragers	Sb eq	CML2-baseline	
	Uitputting biotische grondstoffen – BDP	mbp	TWIN	NIBE
Landgebruik	Landgebruik	PDF*m2yr	Eco-indicator '99	
Hinder	Hinder t.g.v. stank	OTV m3	CML2-baseline, inverse OTV	
	Hinder t.g.v. geluid door wegtransport	DALY	Müller-Wenk	
	Hinder t.g.v. geluid door productieprocessen	mbp	TWIN	
	Hinder t.g.v. licht	mbp	TWIN	
	Hinder t.g.v. kans op calamiteiten	mbp	TWIN	

Figure 3.1: NIBE environmental impact categories

of the product. These are characteristic values of maximum stresses for compression, tension, bending, torsion, and shear that are allowed to occur in the product. Physical properties provide the data necessary for analysis.

Other data refers to the environmental impact of the building product. NIBE is an organization which concerns itself with assessing the environmental impact of building products, their method and data types will be taken as inspiration. Life cycle assessment is the method NIBE chooses for environmental assessment. A LCA can be defined as the "collection and assessment of all inputs and outputs and possible environmental effects of a product system throughout its life cycle." NIBE assesses environmental impact in four different categories: emissions, resource depletion, land use, and nuisance, see figure 3.1.

NIBE takes the data from all four environmental impact categories and calculates a shadow cost for them, see figure 3.2. These are the (prevention) costs required to reach the environmental goal for a given environmental effect set by government and international organizations. Adding all the shadow costs gives the building product one

Milieueffect	Milieukosten	Bron
global warming (GWP100)	€ 0,05 / kg CO2 eq.	CE
ozone layer depletion (ODP)	€ 30 / kg CFC-11 eq.	CE
human toxicity	€ 0,09 / kg 1,4-DB eq.	TNO
aquatic tox. fresh water	€ 0,03 / kg 1,4-DB eq.	TNO
terrestrial toxicit	€ 0,06 / kg 1,4-DB eq.	TNO
photochemical oxidation	€ 2 / kg C2H4 eq.	CE
acidification	€ 4 / kg SO2 eq.	CE
eutrophication	€ 9 / kg PO4-3- eq.	CE
exhaus biotic	€ 0,042202 / mbp	NIBE
exhaus abiotic	€ 0,16 / kg Sb eq.	TNO
exhaus Energy	€ 0,16 / kg Sb eq.	TNO
Eco99 EQ Landuse	€ 0,20482 / PDF*m2yr	NIBE
malodorous air	€ 0,0000000233 / OTV m3	NIBE
Roadnoise	€ 321,946 / DALY	NIBE
hinder geluid	€ 0,00000149 / mbp	NIBE
hinder licht	€ 0,024005 / mbp	NIBE
hinder calamiteit	€ 0,024005 / mbp	NIBE

Figure 3.2: NIBE shadow costs

value that reflects the cost of the environmental impact.

The results from the LCA, but especially the total shadow costs allow the user to compare available building products on their environmental impact. The environmental impact assessment should be carried out by a single organisation such as NIBE. It should not be the responsibility of the manufacturer because there would be a conflict of interest, with manufacturers benefitting from advantageous assessments on their products. It also shouldn't be carried out by

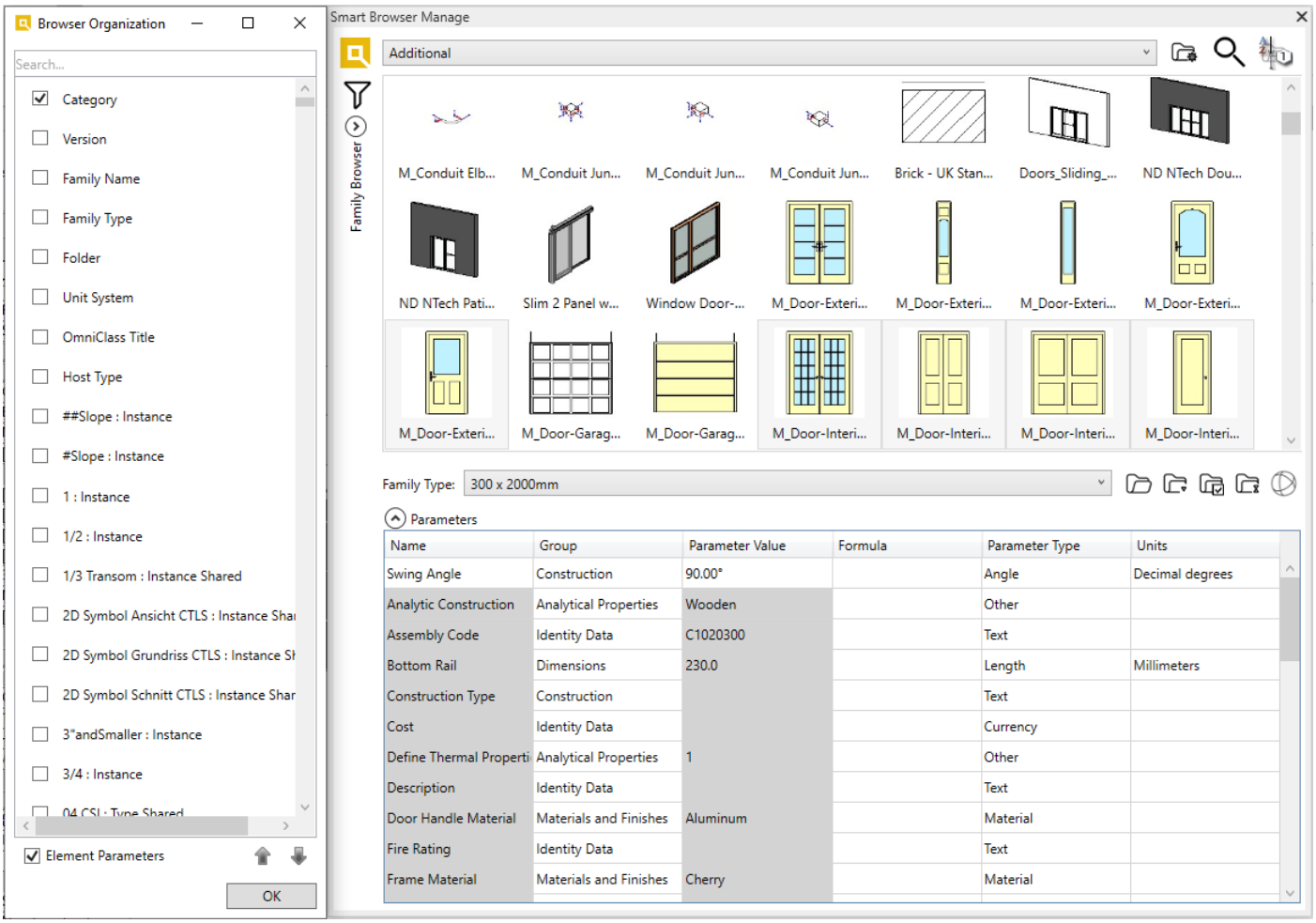


Figure 3.3: AGACAD configurator's BIM family manager

multiple organisations because the method of assessment needs to be consistent for products to be able to be compared against each other.

The last data is the price of the building product given by the manufacturer. Entering building products into the database first requires the selecting of a building component type. Next, the modelling of (parametric) geometry. The modelling can happen in the configurator or any other BIM software that support the creation of BIM families. The models need to be checked to confirm that they are consistent with the other building products in the database, with special regard to the alignment and implementation of size variations. Next steps are selecting connections that are compatible with the building product, giving materials to the geometries in the model, and entering values for the physical and environmental impact properties.

3.1.4 Configuration process

The configuration process starts with modelling the grid, see figure 3.4. The next step is the placement of generic building components in grid slots, see figure 3.5. These components have a building component type, but not a building product assigned to them. This means a generic component knows for example that it is a column, but not what type of column it is. Generic components hold some generic property values that resemble the average of the building products of that component type. Properties such as weight are required to perform a structural analysis. The structural analysis gives, for each generic component, information on the maximum stresses occurring as a result of structural forces (compression, tension, bending, torsion, shear), see figure 3.6. These maximum occurring stresses can be tested against the maximum stresses the building products can withstand, their strength. Building products with a strength too low will get filtered out of the

options for replacing the generic component. The user can choose a replacement from the remaining building products, see figure 3.7. This process is repeated for each of the placed generic components. After replacing the generic components, the structural analysis is run to check if the chosen components are sufficient for the new configuration. This is necessary because by replacing components, the weights of the new components change the maximum stresses occurring in each component. Components that are too weak are highlighted in the configurator; these components need to be replaced once more. The risk is that this process of replacing and analysing has to be repeated many times before arriving at a sufficient configuration. To combat this, components may be chosen that are a bit stronger than is expected to be necessary. This will reduce the chance that the components need to be replaced.

3.1.5 Dynamic components responding to analysis

Implementation of dynamic components that can respond to analysis does not require iteration in the product selection process. Rather, the dynamic components automatically change their geometry according to the requirements resulting from the structural analysis. The resizing of columns and beams in the developed configurator is an example of dynamic components that respond to analysis. For planar elements such as floors, the width and depth respond to the grid slot it occupies, that leaves the thickness to respond to the structural analysis. For linear elements such as beams, the length is decided by the grid slot it occupies, and the profile of the linear element responds to the structural analysis.

3.1.6 Stock-constrained optimization

Another approach to speed up the building product selection process is by automating this process using an optimisation algorithm. The optimisation will ensure that the building products are chosen with strengths that most closely match the requirements of the structural conditions to reduce material usage. This kind of problem is called a stock-constrained optimisation. An example of such an optimisation is that of Warmuth et al. in 2020, they did research on the stock-

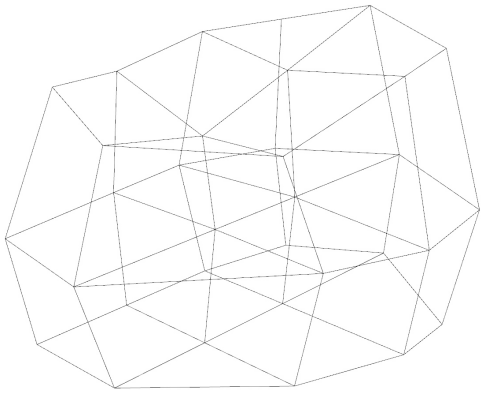


Figure 3.4: Grid modelling

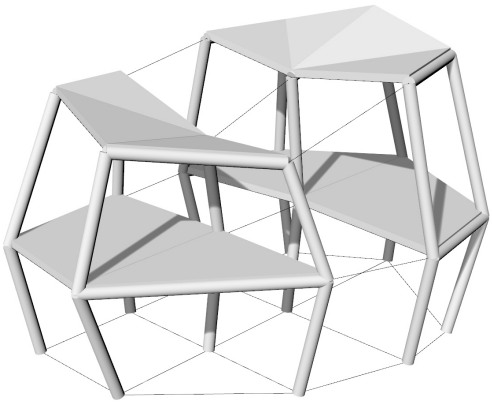


Figure 3.5: Component placement

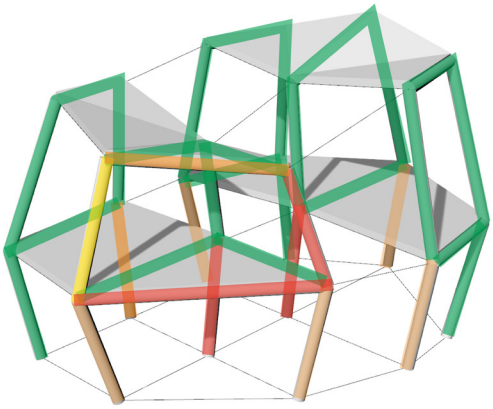


Figure 3.6: Structural analysis

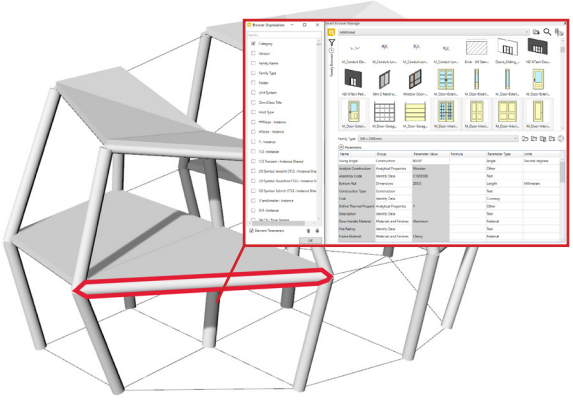


Figure 3.7: Replacing generic component with building product from database

constrained design of truss systems. In this case the members that form the truss are part of the stock, and the result of the optimisation is a building product, the truss. The configurator and building product database offer the opportunity to perform a stock-constrained optimization with a stock of building products, the result being the structural design of a building. Application of such an optimisation has the potential to reduce the material usage in the structure of a building. Moreover, it offers a gateway into the use of reused building components in the design of new buildings.

3.2 Benefits of the proposed configurator

In this subchapter, benefits of the proposed configurator will be examined. Additionally, the configurator will be compared to BIM as this is the prevalent software type used in the construction industry, and in order for the configurator to be commercially viable, it needs to be able to compete with BIM.

3.2.1 Main benefits of the proposed configurator

Because manufacturing and assembly constraints are integrated in the early design phase, early design choices are evaluated on manufacturability. The configurator's building product database holds manufacturing constraints and ensures that manufacturing is possible, and the structural analysis holds assembly constraints and ensures that the assembly of those building products is possible. In response to the structural analysis, the configurator guides the user in building product selection or updates dynamic components automatically. The guidance from embedded expert knowledge makes it easier to make the correct decisions, and automation minimizes the need for manual involvement altogether. Consequently, design issues are avoided that require reorganisation efforts and put pressure on the supply chain.

For example, a designer orders 10 floor systems from a manufacturer, but later figures out that these floor systems are too weak for the design. So, the designer contacts the manufacturer to change the order to 10 stronger floor systems. The manufacturer has already started production on the initial order. Changing the order now delays the initially discussed delivery date, moreover these stronger floor systems cost more, and the manufacturer wants compensation for the late

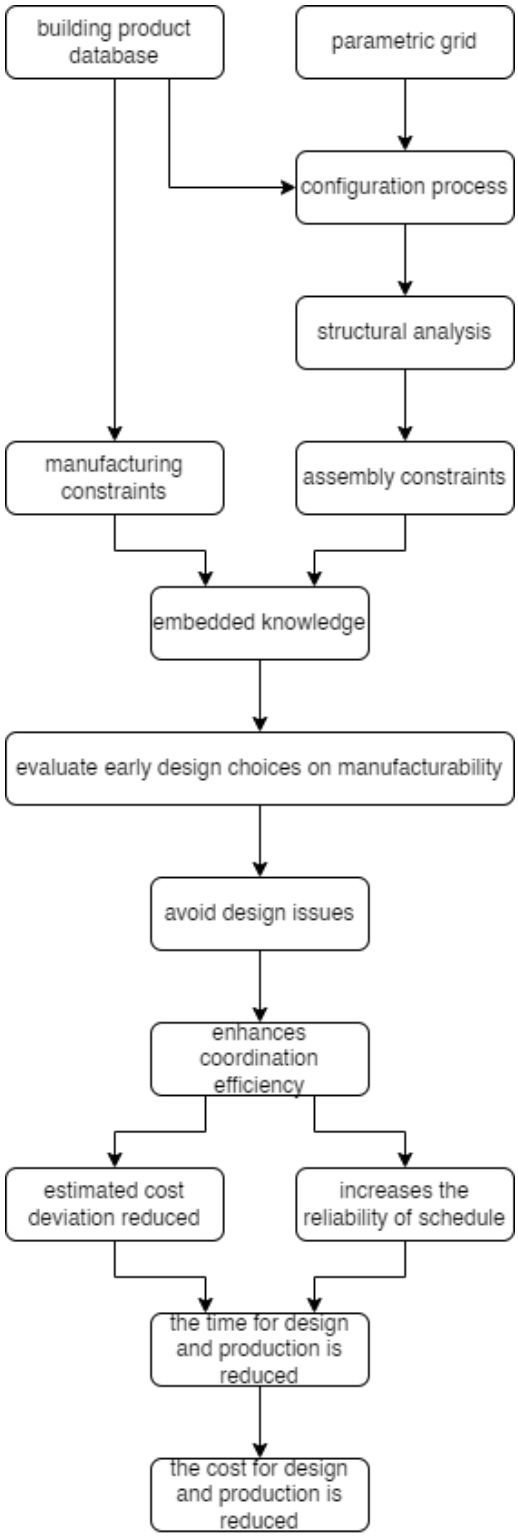


Figure 3.8: Characteristics of the developed configurator and resulting beneftis

notice of change of order. The delay of arrival of the floor systems may also have an impact further along the supply chain, such as in the building process.

Thus, these design issues result in increased time and cost for design and production. Avoiding these design issues enhances the coordination efficiency between designer and manufacturer, and increases the reliability of schedule. Also, because design issues are avoided and because information on building products are available in the early design phase, the estimated cost deviation is reduced. Consequently, the time and cost for design and production is reduced.

3.2.2 Additional benefits of the proposed configurator

The building product database preserves and reuses knowledge on the building products for next projects, whereas in BIM projects are

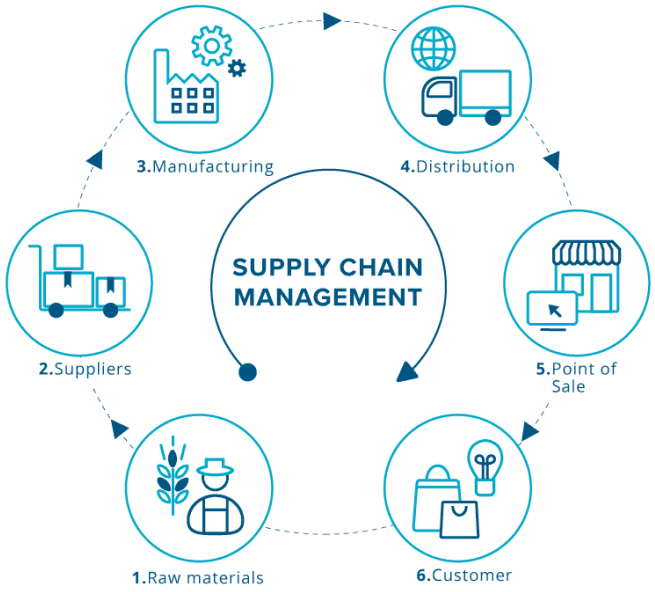


Figure 3.9: The building product supply chain

often built from scratch. This saves time in the design process. Additionally, the building product database has the potential to develop construction documents efficiently, because the building products hold most information necessary to create these documents. This minimizes the need for manual involvement when creating these documents, consequently it also saves time in the design process.

Another benefit is that the software is easy and quick to use. This may save time while handling the software and leads to the potential for engaging with clients, end-users, and other non-engineering professionals in the decision-making processes. The software is easy because of its limitations. The grid discretises the solution space which makes placement of components quick and precise, instead of being able to place components anywhere, components can only be placed in the slots of the grid. This avoids alignment issues, where components are sometimes just placed millimetres from the right spot. However, in conventional BIM, these misalignments may cause issues with functionalities such as analysis because the components are not considered as connected. Additionally, the building product database gives the user an overview of ready to use digital assets. In other words, there are a limited number of building components that go into a limited number of slots for those components.

Additionally, the configurator supports an overarching modularity. Meaning that instead of supporting one generation of product from one company, it is able to support almost every building product. Although the building products may not be compatible with each other, the building products are compatible with the software. The modules are the building products from the database and because the grid is parametric, it can be adjusted to the size or shape of almost any building product. This enables great potential for reuse of building components.

Lastly, because embedded expert knowledge and automation make it easier to make the correct decisions, they avoid design issues but also lead to better designs. In this application of structural design, this means that the selection of building products more closely aligns with the requirements set out by the structural analysis. As a result, material use is reduced.

3.2.3 Comparison to BIM

Theoretically, almost everything that can be done in the configurator can also be done in BIM software like Revit. The main distinction between BIM and the configurator is that the configuration process is different, the way in which the building products are placed and configured. The configurator's configuration process relies on placement on a parametric grid, in BIM the user has more freedom in placement. Consequently, the configurator

understands how building products are connected through the connectivity graph, whereas in BIM more manual involvement is required to make the software understand building product connectivity. This information on connectivity is utilised in the structural analysis. For this reason, the process leading up to and including the structural analysis is more streamlined for the configurator. Additionally, it is uncommon for BIM's design software to have an integrated structural analysis. This means that the building needs to be remodelled in the software for structural analysis, or exported to that software. These hurdles make structural analysis in the early design phase a tedious process in BIM. Therefore, it is unlikely that the user is continuously testing the design by means of structural analysis, so assembly constraints are not applied to the design choices.

A building product database can also be implemented in BIM through plugins for databases of BIM families of building products. However, it is not common practice to apply building product families in BIM, rather projects are built from scratch. For this reason, manufacturing constraints are also not applied to design choices. The exact reason for the lack of BIM family adoption in BIM is unclear, but it is conceivable that it has something to do with BIM's complexity. The configurator aims to simplify the adoption of BIM families in projects by better integrating it into the software, and thereby reducing the required manual involvement. By reducing the software's complexity, non-engineers can also use the software and engage in the decision-making process.

Thus, because manufacturing and possibly also assembly constraints are not applied to design decisions in BIM, early design choices are not evaluated on manufacturability. This means that design issues are not avoided that require reorganisation efforts and put pressure on the supply chain. Consequently, many of the configurator's benefits do not apply to BIM software such as Revit.

3.2.4 The configurator's aspirations

The configurator aims to be easy-to-use software with the targeted audience being designers and non-engineers such as the designer's clients and end-users. The configurator focuses on structural design in the early design phase, but its scope extends to other aspects of the design of buildings, such as climate design. Each aspect influences another, so the integration of multiple aspects into the configurator avoids designers having to switch back and forth between software. Instead, it allows the configurator to act as a common environment.

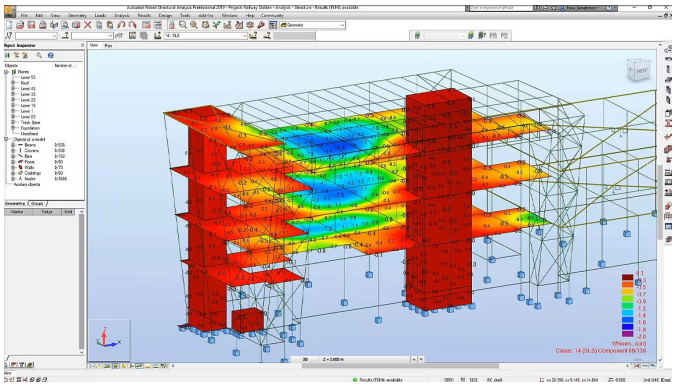


Figure 3.10: Autodesk's Robot Structural Analysis tool for Revit

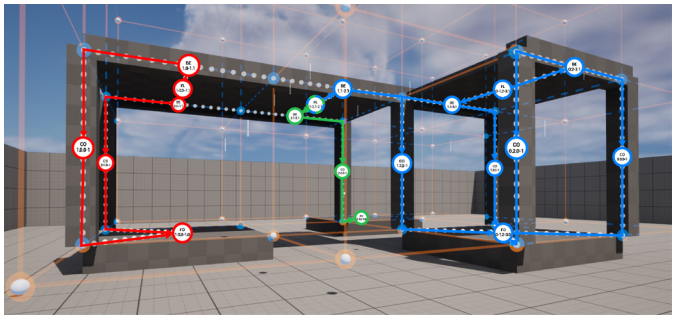


Figure 3.11: Structural analysis of the configurator

The configurator's strategy for becoming easy-to-use is to reduce the requirement for manual involvement of the user. The configurator achieves this by embedding expert knowledge into the software, through the integration of a building product database and structural analysis. The main goal of the configurator is to make the application of building products' BIM families and structural analysis common practice so that early design choices are evaluated on manufacturability. This avoids design issues that require reorganisation efforts and put pressure on the supply chain. Consequently, the configurator reduces the time and cost for the design and production phases, allowing users of the configurator to develop buildings faster and cheaper.

4. Discussion

The previous chapter ended with the benefits of the proposed configurator over BIM. In this discussion the research will be placed in context and the limitations will be discussed.

The designs that can be created in the configurator are limited by options of building products in the database. For this reason, the configurator has a dependency on manufacturers to fill the database with their products. To ensure that building products are compatible with the configurator and can be compared with each other, they need to be checked so that they fit the same format. This includes the way their geometry is modelled and how their structural and climate impact properties are determined.

The possible designs are also limited by the configurator's grid. Building products can only exist on the faces and edges of the grid. Complex grids can allow for complex designs, but the design of a complex grid may also complicate the design process. This is in opposition to the configurator's goal of being easy-to-use. Despite complex grids being possible, a grid similar to the one developed in the configurator does not support curved building products and buildings.
The way in which the configurator implements the building product database and grid makes a clear distinction between vertical and horizontal elements, columns and beams. The grid even distinguishes between the rotation of horizontal elements. In reality, this distinction is not always so clear. Because of this distinction, the configurator will have difficulty with implementing diagonal elements that are somewhat in between a column and a beam.

Other limitations have to do with how the connection conditions are implemented for placed building components. These conditions determine how components can transfer loads to each other. Connections for floor components currently don't consider the span direction of the floor and can be supported by just one beam, which is usually not possible in reality. Moreover, the connection conditions don't take the compatibility with structural connections into consideration. It is assumed that there always exists a structural connection that can be used to connect two building components. However, this is not evident.

Moreover, the structural analysis is very rough and limited to simple configurations. One of the reasons is that structural connections are not considered in the structural analysis. Instead, it is assumed that all connections are pin joints and that there are no horizontal forces. Additionally, the structural analysis can only check axial compression of columns and bending in beams, excluding bending resulting from a cantilever. After checking the stresses in columns and beams, they are resized. The resizing currently doesn't influence the weight of the resized component. The analysis should consider this, because it will influence the required dimensions for that component and other components in the configuration. Furthermore, the structural analysis doesn't consider how the loading and bending of one component affects the rest of the configuration. A method of structural analysis closer to a Finite Element Analysis may be more appropriate to consider the interdependency of building components in a configuration.

Lastly, it is assumed that the lack of application of building products' BIM families in BIM is due to BIM's complexity. As well as that for the same reason there is a lack of application of structural analyses. The configurator needs to be put to the test to see if reducing the required manual involvement leads to increased application of building products' BIM families and structural analysis in projects. Only after testing, it can be concluded whether the configurator's projected benefits do apply.

5. Conclusion

5.1 Conclusions

5.1.1 Research question:

How can the design space of integrated construction configurators be enlarged?

An integrated construction configurator integrates across all phases of construction, planning, design, and production. It evaluates design choices on manufacturability. An enlargement of the design space can be achieved by implementing a smaller module size. Many existing configurators work with modules which are room sized. This project transitions from room sized modules to building component sized modules. Changing the modules of the configurator to building components can also serve the goal of integration, since it has the opportunity to evaluate design choices on manufacturability. When the digital building components are based on physical building products from manufacturers that include manufacturing constraints, the manufacturer can ensure the manufacturability of the component.

The developed configurator took an approach that is based on building components and inspired by games with building systems. These games have in common: a 3D grid, a database of building components, and a configuration rule engine which allows the building components to be configured on the grid. The configuration process is as follows: it starts with designing the grid. Then, components from the building component database are placed on the grid to make a configuration. Next, the configuration is evaluated by means of analysis, in this case a structural analysis. The results of the analysis guide the user in improving the configuration.

The configurator is developed in Unreal Engine, a software development environment. In Unreal Engine different parts of the software are segmented into Blueprints, these are the scripts of Unreal Engine's visual programming language.

The grid is one of those Blueprints. It is made by a 3D matrix of vertices. The vertices are used to make slots, a place in the grid where a component can be placed. There are linear slots, or edges, which are made with 2 vertices. There are also planar slots, or faces, which are made with 4 vertices.

There are 5 types of components in the building component database: beams and columns which fit in the edge slots, and foundations, floors, and walls, which fit in the face slots. In the configurator, when one of these components is selected, the software understands which slots this component can be placed in. All the user needs to do is point at one of the slots and the component will snap to that location.

When a configuration is made, the software recognises which components are neighbours. Connection conditions determine whether neighbouring components are seen as physically and structurally connected. The components and connections between components form a network graph.

To determine the load path for the structural analysis, the Dijkstra shortest path algorithm is implemented to find the shortest path from each of the components to the closest foundation. The live loads and dead loads of each component are passed along the load path to determine the loads acting on each component. The loads are used to find stresses in columns and beams resulting from axial compression and bending. The stress in the component is used to determine the required dimensions for that component.

The developed configurator has the infrastructure that allows the implementation of a building product database, but a database of actual building products has not been realized. A building product database is proposed which is based on BIM families and can serve as a common environment where a diverse range of building products from different manufacturers are represented. It aims to increase the adoption of BIM families in projects. Adoption of building products' BIM families can ensure product manufacturability by integrating manufacturing constraints in the early design process. This building product database sets itself apart from other BIM databases by integrating and interacting with the design software. Building products are recommended based on the dimensions of a specific location in the grid and conditions resulting from structural analysis.

To sum up, the design space of integrated construction configurators can be enlarged by developing a configurator which implements modularity on building component level. The configuration process consists of 3 parts:

the grid, the components, and the analysis. The grid defines the design space in which components can be placed, components from a building product database ensure product manufacturability, and analyses ensure that the assembly of components is possible.

5.1.2 Sub-question:
What are the benefits of this type of integrated construction configurator compared to prevalent BIM software?

BIM's complexity limits the possibility for non-engineers to engage in the decision-making process. The design process of the configurator is simplified by reducing the need for manual involvement. This is achieved by integrating the building product database and structural analysis into the configurator. The aim is to increase the application of BIM families and structural analysis. The building product database preserves and reuses knowledge on the building products for next projects. Whereas in BIM, instead of using ready-to-use BIM families, projects are usually built from scratch. Increased application of BIM families and structural analysis can help integrate manufacturing and assembly constraints into the design process. By doing so, early design choices are evaluated on manufacturability. This can avoid design issues that require reorganisation efforts and put pressure on the supply chain. Consequently, the coordination efficiency between designer and manufacturer is enhanced, there is an increased reliability of schedule, and estimated cost deviation is reduced. As a result of these benefits, the configurator reduces the time and cost for the design and production phases, allowing users of the configurator to develop buildings faster and cheaper.

5.2 Recommendations

To be able to accurately assess the potential of this type of construction configurator which is based on building component level modularity, more research and development is required on:

- The development of a building product database.
- Implementation of a detailed structural analysis, in which the stresses in one component affects the rest of the configuration (like in FEM).
- Implementation of additional analyses such as indoor climate analysis.

Other areas of research that could benefit the field are:

- Developing a component-based configurator for collaboration, such as active real-time collaboration between multiple stakeholders.
- Developing a component-based configurator for reuse, by configuring with used components, or configuring for reusability, such as design for disassembly.
- Optimising configurations, using criteria (from analyses) to determine the optimal configuration. Opportunities lie in application of reinforcement learning to guide the user in component placement, and stock-constrained optimisation to optimise building product selection (opportunity to include reused components).

6. Reflection

6.1 Topic

Why did you choose this topic?

Years before I started this master thesis, during a course called 'Circular Product Design' I had an epiphany. The course had lectures on how building products could be reused by implementing material passports, having databases for used building products, and doing stock optimisations to efficiently implement the used building products in a design. One of these lectures was given by my main mentor Stijn Brancart. In the period in which I followed this course, I was playing a game called Valheim. This game has a building system which uses a database of building products which can be configured to develop structures. I put the two together and realised that the game was a solution to problems in the building sector that were put forward in the lectures of the 'Circular Product Design' course. At this point an idea was conceived, a software supplemented with a database of building products, in which these building products can be configured to make actual buildable structures. I wrote the idea down and then stopped engaging with it. Until, when looking at the available topics for the master thesis, I found the topic 'Discrete Timber' at the Structural chair with Stijn Brancart. Immediately it made me think of this idea I had, and I was motivated to proceed with this topic.

What is the relation with the 'Building Technology' master track and 'Architecture, Urbanism, and Building Sciences' programme?

This project is about developing a software which architects, or anyone for that matter, can use to design buildings. It is about innovating, not what we design, but how we design. It is about reinventing the design process. 'Building Technology' is the discipline which bridges the gap between the architects, that design, and the engineers, which make the design buildable. That is exactly what this software does. It embeds technical know-how into the software so that whatever the architect designs is ensured to be buildable. This design process for buildings, which the software facilitates, is of course very much related to the 'Architecture, Urbanism, and Building Sciences' programme.

6.2 Approach

Why was this approach chosen for the project?

The approach consists of 3 parts. The first part is the literature research. The literature research was performed to understand the context of configurators. To understand what a configurator is, where its origins lie, what configurators are already existing, what the problems are in those configurators, and what possible solutions the literature has to offer. This information is used to define the problem statement and objectives of the project. Without it, I wouldn't know what direction the project should take.

This brings us to the second part of the project, the software development. The information from the literature is used to propose a new configurator. I decided that the configurator would need to get developed to be able to test the proposal. The development of the configurator was a kind of research by design. By developing the configurator, it became clearer how the software can and should function.

The third part of the project is concerned with proposing future developments. Because software development takes a long time for this kind of project, I decided that this proposal was required to give insight into how the configurator could function were it fully developed. This part of the project is meant to bridge the gap between the developed configurator and the proposed configurator resulting from the literature research.

Did the approach work out or not?

I think the information from the literature research gave me a structured way of thinking about configurators. It made me understand the building blocks that make up a configurator. It also helped to validate and guide the problem statement and objectives of the project. What it didn't do was guide me in the development of the software.

I had to learn how to use Unreal Engine to develop the configurator. Even for learning this software there was almost no documentation. I learned in a fragmented manner from YouTube videos, forums, by asking questions at the university and faculty VR lab's, and most of all by trial and error. Despite the difficulty in working with Unreal Engine I think it was the right choice.

Reflection

I don't think this type of configurator could have been developed in Rhino, for example. Additionally, it does make sense to developed the configurator in the software that some of the games which served as inspiration were made with.

The features I chose to develop are I think the minimum requirements for the project. I developed the core software architecture that allows building components to be placed on a grid, supplemented by a structural analysis. I think the structural analysis is also a fundamental part of the project because my topic is in the Structural chair. Moreover, because it highlights one of the strengths of this configurator, having information on the assembly of components. I did run into some issues with time constraints as I would have liked to have spent more time on the structural analysis as well as some other features.

How can research by design be implemented in a scientific manner for software development?*

The research approach did make it difficult for me to answer my research question in a scientific manner. Because by developing the software I didn't get any quantifiable results. I can't proof that the way that I chose to develop this configurator is the correct way. I can only explain how the configurator works and why I made the decisions that I did. What made it even more difficult was that the developed configurator is part of a larger concept that couldn't be completely developed. For this reason, it is hard to say whether the developed and proposed configurator is the solution to the problems brought forward in the problem statement.

The same issue occurred when proposing future developments. I try to use my imagination in combination with logical reasoning to determine if certain features of the configurator will solve problems from the problem statement. There are no calculations which can be done, there is no literature or precedents to give insights, the only way to know for sure is to develop the configurator with these features.

Did the approach consistently result in new information relevant to the research question?*

Throughout most of the process I worked on software development. The new information that was revealed during this process mostly had to do with how features should be coded in Unreal Engine. There wasn't a lot of new information on what the software should be during this process. This information on coding was too detailed and not really relevant for my research. However, it did result in a tool that helps answer my research question.

How did your research method differ from the methodical line of approach of the graduation studio?

My research method was very different from most other students graduating in 'Building Technology'. That is because my topic is very different. Most other students develop a design or recommendations for design. I developed a design tool. Topics closest to mine are design tools in the field of computational design. The difference is that most of those tools don't handle any interaction with the user, they don't let the user decide. They take data and process this data to give a recommendation or determine the design. For this reason, my approach was more focused on software development and less on calculating or working with data.

Did you encounter moral/ethical issues or dilemmas during the process? How did you deal with these?

There exists a moral/ethical issue which has to do with automation. The configurator has knowledge embedded into the software which could mean that less engineers are required to check the design. Moreover, the design process is simplified, it is based on games that children intuitively learn how to use. This also means that cases could exist where instead of the architect designing the building, the end-user would be directly responsible for designing the building. So, it could cause a loss of jobs. I do think the benefits outweigh the drawbacks, so it is worth developing the software. Such a tool could make buildings cheaper, more environmentally friendly, and in general improve designs for buildings.

How do you reflect on the feedback of your mentors?

The feedback that I got had mostly to do with controlling the scope and method of the project, making clearer what the added

value is of the configurator, and what features should or shouldn't be added to the software. I think the feedback helped guide me in the direction that I took the research and software development. It did not help much with the software development itself though. Unreal Engine was a software that both my mentors are unfamiliar with, so I had to figure this out myself.

How was your mentors' feedback translated into your work?
The scope of the project became developing a configurator that is able to validate a configuration with a rough structural analysis. Making clear what the added value is of the configurator was translated in the report by on the one hand developing the configurator and explaining how it works, and on the other hand by explaining what still needs to be developed and how it can work. Then explaining the benefits that the different features of the configurator provide. Finally, I didn't implement the features that my mentors said were out of scope, but also haven't implemented the features that my mentors would have liked to see. This has to do with time constraints, and me choosing to focus on the core of the project first.

What did you learn from your own work?
First of all, I learned to use the most complex piece of software that I know, Unreal Engine. Secondly, I learned more about software development in general. Moreover, I also learned about design software, about how small choices on software development can have a big impact on the design process which the software provides. Also, I learned about controlling the scope of a project. Lastly, I learned a bit about how to get an abstract idea across, about translating an idea that only lives in the mind to other people through text, speech, and imagery.

6.3 Results

How did the preliminary results of the research and design come to be (product, process, planning)?
Most of it is a result of software development. For most of the features I had a rough idea of how I wanted it to work. Then, I would start with looking online at ways that others had implemented this feature, or I would ask people at our VR labs that had experience with Unreal Engine. Once I knew how the feature needed to work, I would write code, then test, then troubleshoot and make edits to the code, and repeat until the feature was developed. The last step is then to explain and contextualize those features which I had developed, by writing.

To what extent has the projected innovation (research objective) been achieved?
My research objective is 'to discover how an integrated construction configurator which enlarges the design space should function, and to develop (part of) the integrated construction configurator'. I succeeded in creating a construction configurator although it might not be an integrated one. An integrated construction configurator supports the design process through all stages, planning, design, and production. The construction configurator which I developed can't really support the production phase since it does not yet offer the information required to build the configuration. The configurator was meant to act as a proof of concept, showing the potential of this type of configurator. I'm not sure if the configurator was developed far enough to make the benefits of this type of configurator clear. Also, I don't think I haven't exactly figured out how this type of configurator should function, but I have provided information on how it can function. I'm hoping at the very least that my project inspires some so that more research will follow.

How do you assess the academic and societal value, scope, and implication of your graduation project, including ethical aspects?
My project by itself, the configurator that I built, won't have a large impact. If my project attracts attention to the concept of such a configurator and if it were fully developed, it could have a big impact. I believe this type of configurator has the potential to revolutionize the way in which we design buildings. Ultimately making buildings cheaper and more sustainable, it could help in overcoming the housing crisis. This happens by having information on production available early in the design process, it helps to reuse, standardize, validate, and iterate more efficiently.

Reflection

To what extent are the results applicable in practice?
The configurator itself could only be useful in validating a simple structure in the planning phase, or just quickly exploring design options. Other applications of the results are in the domain of the software development of architectural software. It provides examples and ideas for developing new software, or extending existing software with new features.

How do you assess the value of the transferability of your project results?
For anyone familiar with the development of software, this project should provide enough information so that this configurator can be remade. It also provides ideas for future development.

7. References

Abanda, F. H., Tah, J. H. M., & Cheung, F. K. T. (2017). BIM in off-site manufacturing for buildings. Journal of Building Engineering, 14, 89–102. https://doi.org/10.1016/j.jobe.2017.10.002

Barman, S., & Canizares, A. E. (2015). A survey of mass customization in practice. International Journal of Supply Chain Management, 4(1), 65–72.

Benjamin, S., Christopher, R., & Carl, H. (2022). Feature modeling for configurable and adaptable modular buildings. Advanced Engineering Informatics, 51. https://doi.org/10.1016/j.aei.2021.101514

Bianconi, F., Filippucci, M., & Buffi, A. (2019). Automated design and modeling for mass-customized housing. A web-based design space catalog for timber structures. Automation in Construction, 103, 13–25. https://doi.org/10.1016/j.autcon.2019.03.002

Cao, J., Bucher, D. F., Hall, D. M., & Lessing, J. (2021). Cross-phase product configurator for modular buildings using kit-of-parts. Automation in Construction, 123. https://doi.org/10.1016/j.autcon.2020.103437

Cao, J., & Hall, D. (2019). AN OVERVIEW OF CONFIGURATORS FOR INDUSTRIALIZED CONSTRUCTION: TYPOLOGIES, CUSTOMER REQUIREMENTS, AND TECHNICAL APPROACHES. Proceedings of the European Conference on Computing in Construction, 295–303. https://doi.org/10.35490/ec3.2019.145

Forza, C., & Salvador, F. (2006). Product Information Management for Mass Customization: Connecting Customer, Front-office and Back-office for Fast and Efficient Customization. Product Information Management for Mass Customization: Connecting Customer, Front-Office and Back-Office for Fast and Efficient Customization, 1–221. https://doi.org/10.1057/9780230800922/COVER

Goulding, J., & Rahimian, F. (2019). Offsite production and manufacturing for innovative construction: People, process and technology. https://books.google.com/books?hl=nl&lr=&id=1T33DwAAQBAJ&oi=fnd&pg=PP1&ots=m7aev-JhWx&sig=yoPRJN3guT00nJ8msJzyRkwcByw

Hamid, M., Tolba, O., & El Antably, A. (2018). BIM semantics for digital fabrication: A knowledge-based approach. Automation in Construction, 91, 62–82. https://doi.org/10.1016/j.autcon.2018.02.031

Hvam, L., Mortensen, N. H., & Riis, J. (2008). Product customization. Product Customization, 1–283. https://doi.org/10.1007/978-3-540-71449-1

Kotha, S., & Pine, B. J. (1994). Mass Customization: The New Frontier in Business Competition. The Academy of Management Review, 19(3), 588. https://doi.org/10.2307/258941

Lee, C., & Ham, S. (2018). Automated system for form layout to increase the proportion of standard forms and improve work efficiency. Automation in Construction, 87, 273–286. https://doi.org/10.1016/J.AUTCON.2017.12.028

Louth, H. D., Fragachan, C., Bhooshan, V., & Bhooshan, S. (2024). Configurator: A Platform for Multifamily Residential Design and Customisation. In Lecture Notes in Mechanical Engineering: Vol. Part F1562 (pp. 769–805). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-031-36922-3_40

Meyer, M. H., & Lehnerd, A. P. (1997). The power of product platforms. The Free Press, December 1990, 288. https://books.google.com/books/about/The_Power_of_Product_Platforms.html?id=PKJuQjSaHp0C

Myrodia, A., Randrup, T., & Hvam, L. (2018). Configuration Lifecycle Management – An Assessment of the Benefits Based on Maturity (pp. 119–124). University of Hamburg. https://orbit.dtu.dk/en/publications/configuration-lifecycle-management-an-assessment-of-the-benefits-

Peltokorpi, A., Olivieri, H., Granja, A. D., & Seppänen, O. (2018). Categorizing modularization strategies to achieve various objectives of building investments. Construction Management and Economics, 36(1), 32–48. https://doi.org/10.1080/01446193.2017.1353119

Piroozfar, P., Farr, E. R. P., Hvam, L., Robinson, D., & Shafiee, S. (2019). Configuration platform for customisation of design,

References

manufacturing and assembly processes of building façade systems: A building information modelling perspective. Automation in Construction, 106, 102914. https://doi.org/10.1016/j.autcon.2019.102914

Potseluyko, L., Pour Rahimian, F., Dawood, N., Elghaish, F., & Hajirasouli, A. (2022). Game-like interactive environment using BIM-based virtual reality for the timber frame self-build housing sector. Automation in Construction, 142. https://doi.org/10.1016/j.autcon.2022.104496

Said, H. M., Chalasani, T., & Logan, S. (2017). Exterior prefabricated panelized walls platform optimization. Automation in Construction, 76, 1–13. https://doi.org/10.1016/J.AUTCON.2017.01.002

Tetik, M., Peltokorpi, A., Seppänen, O., & Holmström, J. (2019). Direct digital construction: Technology-based operations management practice for continuous improvement of construction industry performance. Automation in Construction, 107, 102910. https://doi.org/10.1016/j.autcon.2019.102910

Warmuth, J., Brütting, J., & Fivet, C. (2020). Computational tool for stock-constrained design of structures.

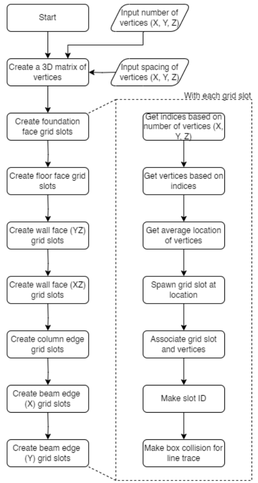
Yuan, Z., Sun, C., & Wang, Y. (2018). Design for Manufacture and Assembly-oriented parametric design of prefabricated buildings. Automation in Construction, 88, 13–22. https://doi.org/10.1016/j.autcon.2017.12.021

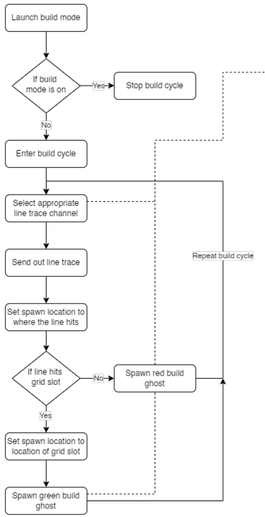
8. Figures

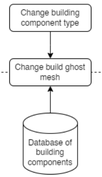
Figures

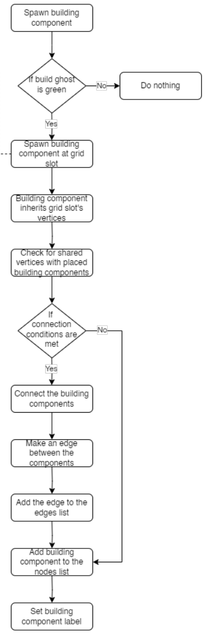
9. Appendix. Complete Software Development Flowchart

AC_BuildComponent

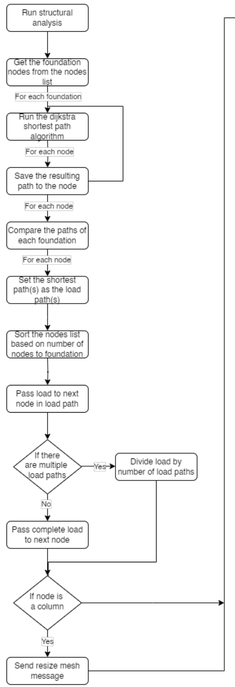




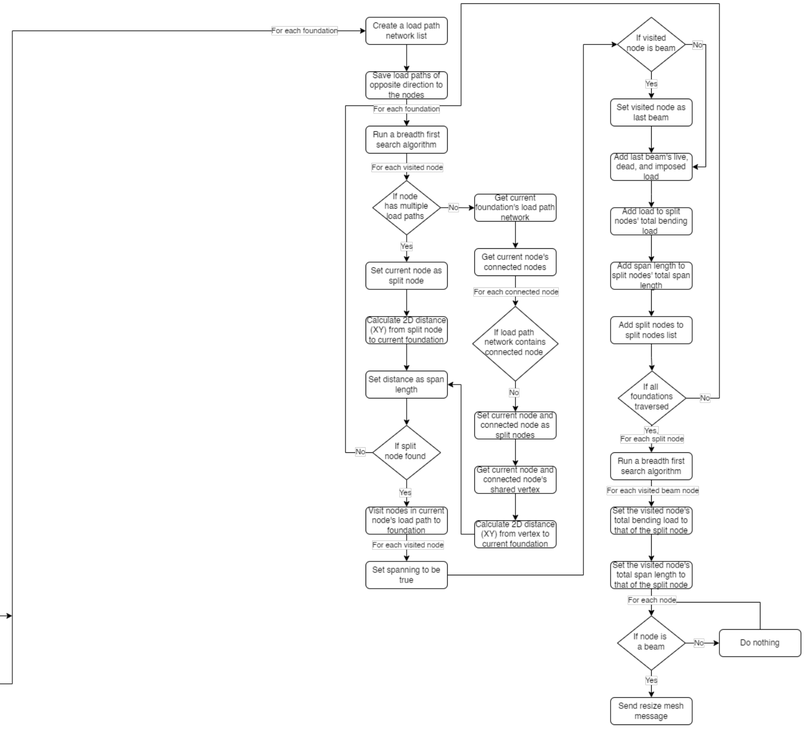




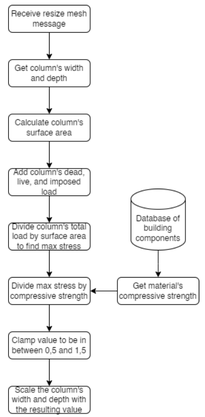
AC_StructuralAnalysis: Load Path



AC_StructuralAnalysis: Bending



BP_Column



BP_Beam