REUSING WASTE WOOD

FOR AN EXTERIOR WALL ELEMENT

TU Delft
BK Bouwkunde

COLOPHON

REPORT P5
Master Thesis

TITLE
Reusing waste wood for an exterior wall element

UNIVERSITY
Technical University of Delft
Faculty of Architecture & the Built Environment
Master Building Technology

TUTORS
Ir. Pieter Stoutjesdijk
Building product innovation
Main mentor

Ir. J.J.J.G. Hoogenboom
Design informatics
Second mentor

BOARD OF EXAMINERS DELEGATE
Mr.dr. H.D. Ploeger

DATE
08-04-2021

STUDENT
Joost van Eijk

STUDENT NUMBER
4222911

ACKNOWLEDGMENTS

Completing this graduation project has been an interesting and intensive process that I could not have completed without the help and support of the people around me. I would like to thank my main mentor Pieter Stoutjesdijk for introducing me to this intriguing topic, his guidance, and for getting me on the right track. I would also like to thank my second mentor Hans Hoogenboom who has helped me in the process of understanding and integrating Python within my thesis. I want to thank all my friends and roommates for their mental support and positive words. Lastly, I would like to thank my parents, Karin & Marco, who have unconditionally supported me throughout my entire study, no matter what interesting sidetrack I decided to take.

Joost van Eijk
Rotterdam, April 2021

TU Delft
BK Bouwkunde

SUMMARY

In 2017, 1610 million kg of waste wood was collected in the Netherlands. The building and demolition industry was, and still is, the most significant contributor. The estimation is made that in 2017 23% of this wood, 370 million kg, consisted of solid waste wood. This solid waste wood has the potential to be reused as a building material without having to shred it. To this day, with the current waste wood processing methods, this high potential waste wood ends being shredded for the engineered board industry or incinerated for bioenergy. The problem with engineered boards is that they cannot be recycled again into an engineered board due to the adhesives. They can only be incinerated for bio energy. If all the 370 million kg of solid B waste wood is downcycled into engineered boards instead of reused, an additional 399.900.000 kg of CO2 is emitted.

Reusing waste wood again as a building material instead of downcycling it into an engineered board is considered circular. Therefore, it is in line with the Dutch government goal to create a circular economy by 2050. However, the current commonly used timber frame construction methods in the Netherlands are not suitable for construction with waste wood. Waste wood has certain characteristics that differ from regular wood for construction on a larger scale. It is possible to create an exterior wall element from waste wood, but this is a time-consuming process. Manual labor is expensive in the Netherlands. Therefore, it will probably not be an economically feasible solution to construct the wall element. A carpenter that needs to modify every piece by hand to get everything in the correct dimensions will probably be more expensive than buying new wood.

Using digital fabrication combined with a parametric model can offer a solution to the challenges of constructing an exterior wall element with waste wood. The characteristics of the waste wood are obtained and digitally stored in a database. This database can communicate with a parametric model and generate all the separate 3dm files required for digital manufacturing.

A design methodology was used in order to identify the design problems, determine the criteria and guide the designer in finding the most suitable design that can work with the challenges of waste wood. The obtained design fulfilled the objective by having a design that can be constructed as much as possible with waste wood, minimize any additional material waste and be inline with the circular economy. This means that the pieces of wood need to be reused as much as possible after the exterior wall element end of life. As a result of following the methodology, a set of design principles were derived:

- Modifications on the pieces of wood should only be on the outside.
- Symmetry in the design should be prevented.
- A broad range of dimensions should be used in the design.

The final design was made parametric so that it could communicate with the database. Using Rhino, Grasshopper and Python, a prototype of the exterior wall element tool was made that bridged the parametric model with a waste wood database and gave the user some insight into the waste wood statistics of the design. Python scripts were used for an efficient selection procedure whereby additional material waste could be minimized. The exterior wall element

The final design combined with the exterior wall element tool can calculate how many wall elements can be constructed from waste wood. The exterior wall element tool with the final design calculates that a wall element with a height of 3 meters, a width of 4,8 meters and a center-to-center distance of 600 mm weighs, on average, 503 kg. For a 100m2 single floor square house with a height of 3 meters, a 120m2 wall is required. 120m2 exterior wall is 4.192 kg. If 10% of the solid B-wood can be reused, 8.826 houses can be constructed every year.

With the exterior wall element tool, it is possible to create a national database with waste wood. The sellers of waste wood can be the demolishing companies but also the waste wood processors. A contractor can use the exterior wall element tool online to create an exterior wall element from waste wood. Here the user, in this case the contractor, gives the desired dimensions of the wall, and the tool generates a design of an exterior wall element. Additional information is given about the cost, the amount of wood required and where the wood is located. The contractor can adjust the parameters of the model and choose, for example, the cheapest option or the option where all the wood is closest to the building side. In the end the tool generates a list with 3dm files for every piece of wood that has to be modified. These 3dm files can be used for digital fabrication.

TABLE OF CONTENTS

1 INTRODUCTION

This chapter will give a brief introduction into the graduation topic

Used wooden beams,
(Gebruikte bouwmaterialen weert)

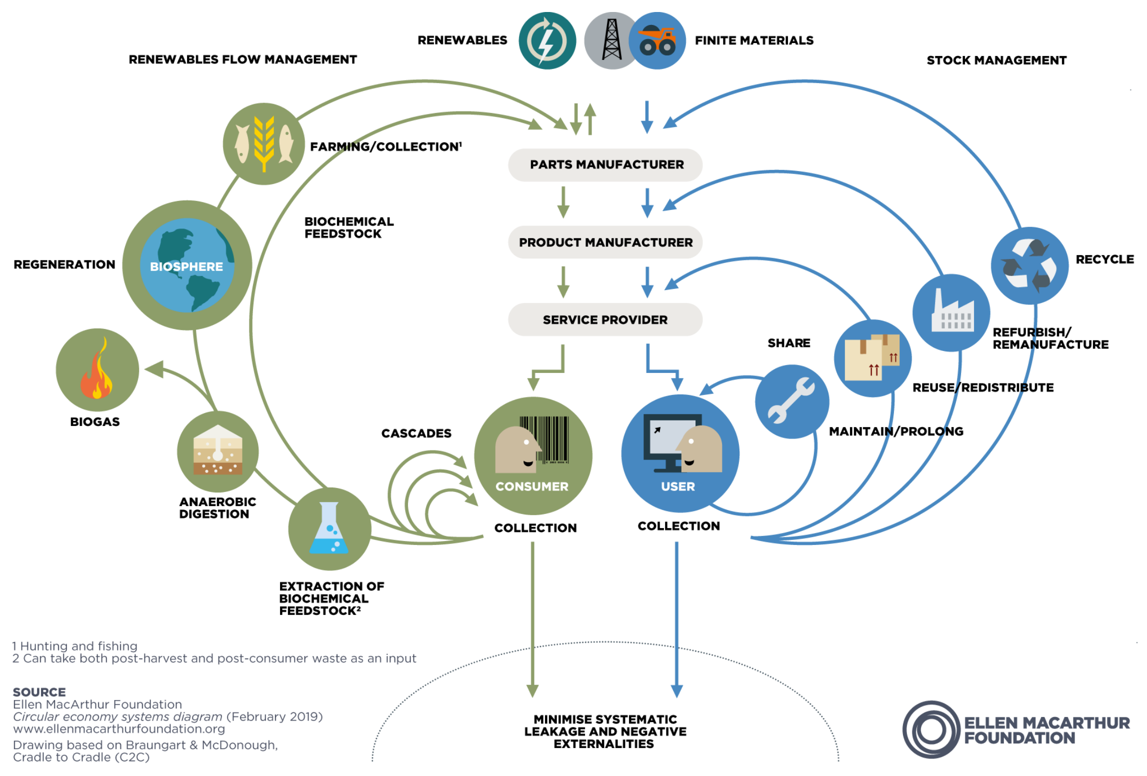
1.1 WASTE WOOD IN THE NETHERLANDS

In 2017, 1610 million kg of waste wood was collected in the Netherlands. The building and demolition industry was, and still is, the most significant contributor. The estimation is made that in 2017 23% of this wood, 370 million kg, consisted of solid waste wood (Bruggen & Zwaag, 2017). This solid waste wood has the potential to be reused as a building material without having to shred it. To this day, with the current waste wood processing methods, this high potential waste wood ends being shredded for the engineered board industry or incinerated for bioenergy.

Meanwhile, the Netherlands is facing two significant challenges. There need to be one million additional houses before 2030 to challenge the increasing shortage (ABF research, 2018). Secondly, the Dutch government stated that the building economy needs to be completely circular by 2050 (Rijksoverheid, 2018). This circular goal is to stop the significant impact the building industry has on the climate because the building and construction industry is responsible for 39% of all carbon emissions in the world (World Green Building Council, 2019). The circular economy is based on three principles (Ellen MacArthur foundation, 2017):

- Design out waste and pollution
- Keep products and materials in use
- Regenerate natural systems

Reusing waste wood again for the same purpose is considered circular. So, reusing a wooden beam in a new building again as a beam is circular. Shredding a wooden beam into wood chips to produce an OSB panel is not considered circular, even though the material is used again as a building material. To manufacture OSB, adhesives are added to glue the shredded wood together. Because of these adhesives, the wood cannot be recycled again and can only be incinerated for bioenergy. A wooden beam turned into an engineered board, such as OSB, can only be recycled once. While reusing it as a beam allows the wood to be reused multiple times. When this is not possible anymore, it can be sawed in smaller pieces for furniture, for example. The wood's value has decreased, but it is still a better purpose than shredding it to manufacture an engineered board. When the furniture is disposed, and the wood has no other use, it can be shredded and turned into an engineered board. After that cycle, it can be incinerated for bioenergy. This process allows the wood to remain part of the circular economy till the end of its life.



SOURCE
Ellen MacArthur Foundation
Circular economy systems diagram (February 2019)
www.ellenmacarthurfoundation.org
Drawing based on Braungart & McDonough, Cradle to Cradle (C2C)

Figure 1.1 Circular economy (Ellen MacArthur foundation)

Due to the required one million additional homes before 2030 and the goal to have a circular economy by 2050, the expectation can be made that the demand for circular building materials will increase in the future. Meanwhile, waste wood gets incinerated and downcycled that has the potential to be reused. Reusing this waste wood as a building material can offer a solution to both these problems.

This thesis aims to research and develop a tool that allows waste wood to be reused as a building material circularly and helps with the increasing housing demand challenge. Given the timeframe for this graduation project and to specify the tool's goal, the choice is made to focus on an exterior wall element instead of a complete house. The tool's output is an exterior wall element that is constructed with the current availability of waste wood. The exterior wall element must be designed so that it can work with the challenges of waste wood.

1.2 CREATING AN EXTERIOR WALL ELEMENT

Manual labor is expensive in the Netherlands. Therefore, it will probably not be economically feasible solution to sort and measure all the waste wood by hand. The same goes for constructing the wall element. A carpenter that needs to modify every piece by hand to get everything in the right dimensions will probably be more expensive than buying new wood.

Design freedom in the dimensions of the wall element will allow for more value. Allowing the user to let the design fulfill their preference is always preferred above a fixed number of design options. However, adjusting everything constantly to the preferences of each user is a costly process. New techniques can offer solutions to these problems. Robotic sorting and machine learning can be used to identify the wood that is suitable for reuse. Scanning can be applied to obtain the dimensions and specific characteristics of the wood.

Digital manufacturing can modify the wood in the right dimensions without expensive manual labor from a carpenter. Parametric modeling can be used to develop a wall element that can generate itself in an efficient way that minimizes additional waste while still dealing with the user's preferences.

Robotic sorting and 3d scanning are techniques that are used today. The same goes for digital manufacturing in the form of CNC milling. In this thesis, the focus lies on the communication between a digital database of waste wood and a parametric model of an exterior wall element.

In the next chapter, the research framework is presented to explain this thesis further.

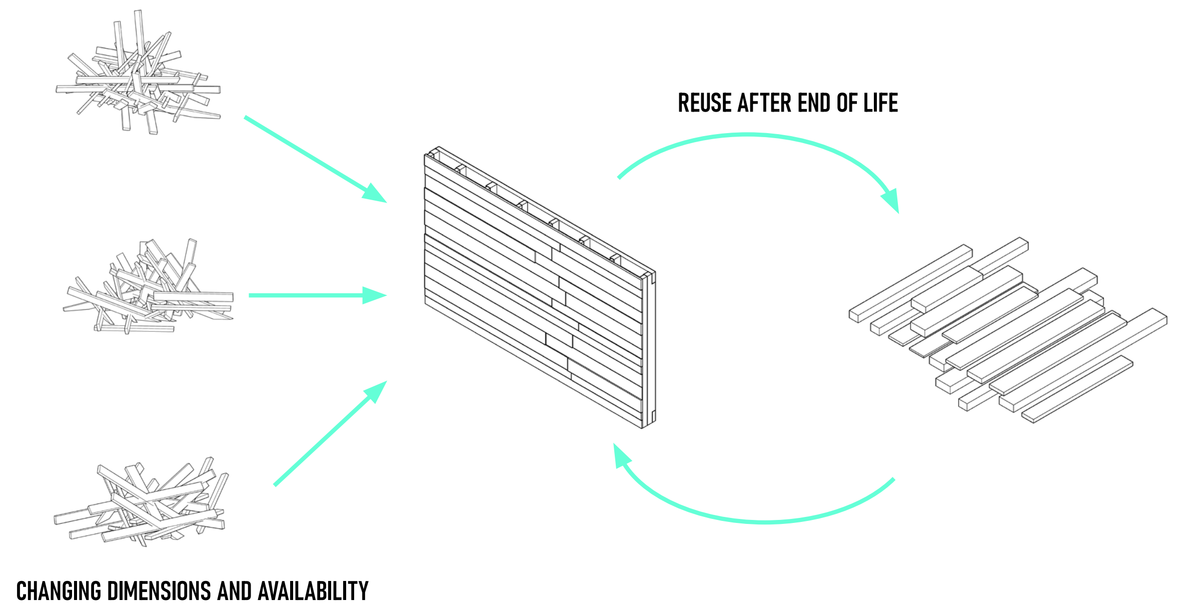
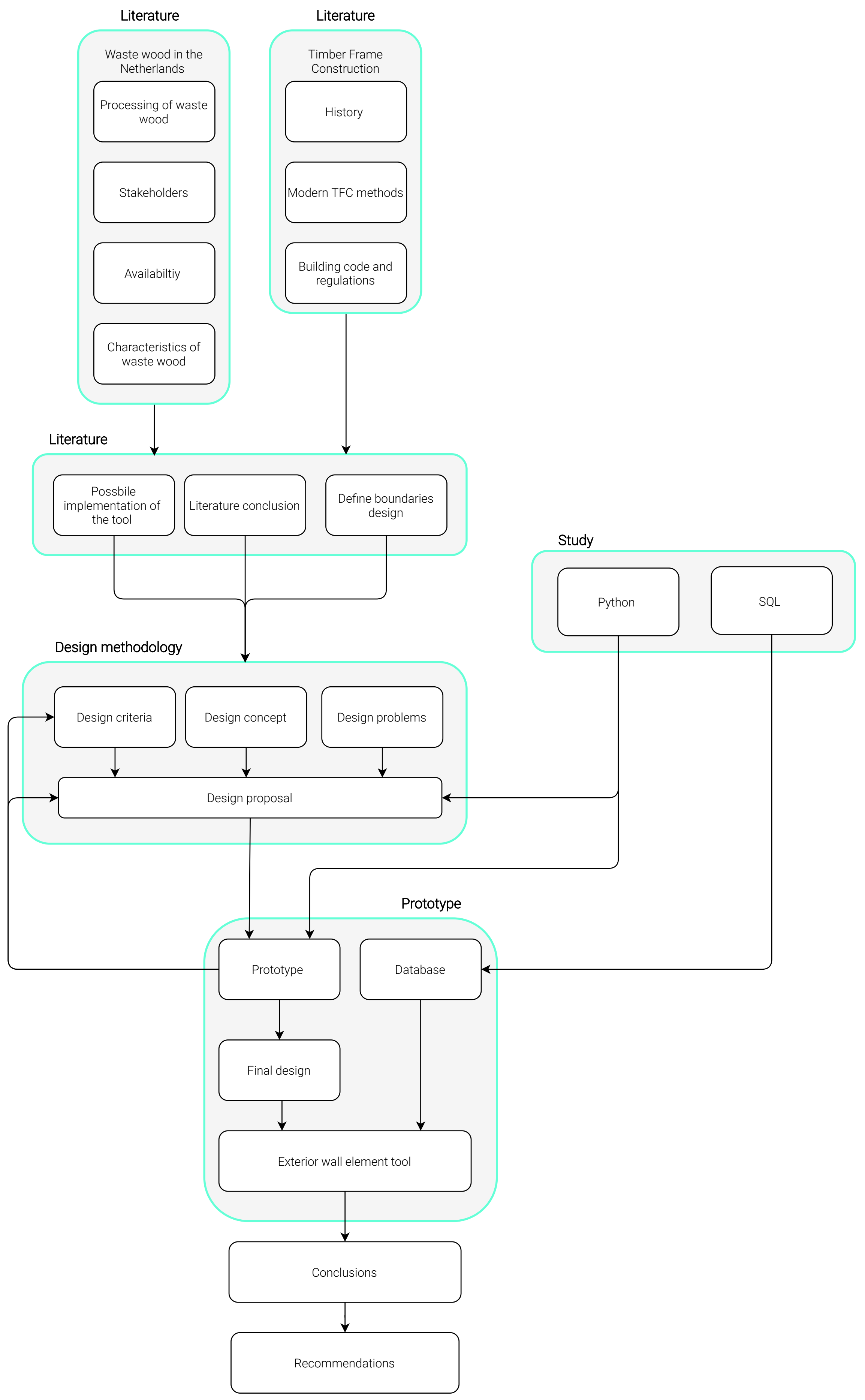


Figure 1.2 Parametric model of an exterior wall element (own ill.)

2 RESEARCH FRAMEWORK

In the chapter the used research framework for this graduation project will be explained. The problem statement, the objectives, the boundary conditions & limitations and research question will be elaborated.



2.1 PROBLEM STATEMENT

Waste wood in the Netherlands gets incinerated for bioenergy or downcycled to engineered boards. This also happens for waste wood that can be reused as a building material. In 2017, there was 370.000.000 kg of waste wood that had the potential to be reused as a building material. Suppose all this waste wood was downcycled into particleboards. This would have resulted in an additional 399.900.000 kg of CO2 emissions (CES EduPack 2019). This number is equal to the yearly carbon footprint of the energy demand for 89.500 household in the Netherlands (Mileucentraal). The calculations can be found in the appendix of this thesis.

Right now, there is no financial reason for companies to reuse waste wood on a large scale (Bruggen & Zwaag, 2017). With the Dutch government's aim to have a completely circular building economy by 2050 and building one million additional houses before 2035, this can change. However, no significant purpose or method is developed yet for reusing waste wood in the construction of buildings on a larger scale.

For waste wood to be reused, it must be sorted, stored, and modified. Doing this with manual labor is expensive in the Netherlands. The additional problem is that the changing sizes of waste wood available for construction result in an intricate design and construction challenge for the architect and contractor.

New technologies can reduce the amount of manual labor and allow for working with the changing dimensions of waste wood. Robotic sorting can replace manual labor. A parametric model of the exterior wall element with a good selection script can handle the waste wood's changing dimensions and reduce additional waste during the exterior wall element construction. Using a digital database with all the sorted and stored waste wood characteristics can function as an input for the parametric model and the selection script.

Research is required about these technologies to develop an exterior wall element tool that can be implemented in the Dutch waste wood market.

2.1.1 SUBPROBLEMS

There are currently two problems that prevent developing an exterior wall element tool and creating an exterior wall element from waste wood with the current timber frame construction methods.

Database
A database with reusable waste wood in the Netherlands does not exist. There is no record that shows what percentage of the solid B-wood is suitable for reuse. Only B & C wood need to be registered by law. The wood that is registered is in number of kilo's, there is no data about the dimensions of the waste wood.

Exterior wall design
The current commonly used timber frame construction methods in the Netherlands are not suitable for construction with waste wood. Waste wood has certain characteristics that differ from regular wood for construction. The developed design principle of the exterior wall element must be compatible with the constantly changing availability of wood with different dimensions.

2.2 OBJECTIVE

The objective of this thesis is to develop an exterior wall element tool that can be implemented within the Dutch waste wood market. The tool consists of a digital database that can communicate with an exterior wall element's parametric design. The design principle of the exterior wall element must be in-line with the circular economy to achieve the government's circular building economy goal by 2050. The design of the exterior wall element is the result of following a design methodology.

2.2.1 SUB OBJECTIVES

Database creation:
A dummy database is created with the characteristics of waste wood. This database should suggest how a future waste wood database can be designed. There is no data available in the Netherlands about the dimensions of reusable waste wood. Market research, a location visit, and interviews are conducted to create a realistic dummy database.

Exterior wall element
To make a parametric design of an exterior wall element that can work with the exterior wall element tool, a suitable design must be created that can work with waste wood. This design is the result of following a design methodology. From this process, some design principles can be derived. The design principles and following the design methodology result in a design that can work with waste wood. The objectives of this design can be specified as follows:

- The design should be constructed as much as possible with waste wood.
- The design should minimize any additional material waste that can occur when building this exterior wall element.
- The design must be compatible with the circular economy. This means that the pieces of wood need to be reused as much as possible after the exterior wall element end of life.

Parametric model
A parametric model is created of the designed wall element. This parametric model allows the design to work with the changing availability of the waste wood. Additionally, the user of the tool can alter the dimensions of the wall to fit their preference. For the model to be more realistic, the user can choose to integrate a door and a window inside the wall element.

The user can be anybody who wants to build an exterior wall element from waste wood. However, in this thesis, a suggestion is made for the implementation of the tool where the user is the waste wood processor and the contractor.

Selection script
The selected waste wood from the database needs to work with the design of the exterior wall element. The selection of the wood is preferably done in the most efficient way. Thereby reducing any additional material waste

2.2.2 END PRODUCTS

- Design and design principle of an exterior wall element that can be constructed with waste wood.
- Dummy database that communicates with the exterior wall element tool.
- Prototype of the exterior wall element tool.

2.3 BOUNDARY CONDITIONS AND LIMITATIONS

It would be interesting to research if a complete building can be constructed with waste wood. However, it would not be possible considering the time available for this thesis. The walls of a building consist of the most surface. Therefore, the decision is made to focus on only that aspect. If an exterior wall element can be constructed from waste wood and meet the building code requirements, it is safe to assume this design can then also be used for the interior walls.

Structural calculations are not conducted for this thesis because of the given time for this graduation project. Within the master Building Technology, it is mandatory for the graduation project to connect two fields within the master. This thesis focuses on the field of design informatics and building product innovation. It would be too time-consuming to include the third field of structural mechanics. For the design to be realistic, existing wooden wall elements are analyzed, and their dimensions are used as a reference. Implementing structural calculation would be an interesting topic for another student to continue this research on.

An exterior wall element in the Netherlands needs to fulfil the requirements of the building code. These requirements are about sound insulation, thermal insulation, airtightness, etc. In this thesis, the choice is made to fulfil these needs with additional already proven materials due to the given time frame. It would be interesting to research if waste wood can meet these requirements without the need for additional materials.

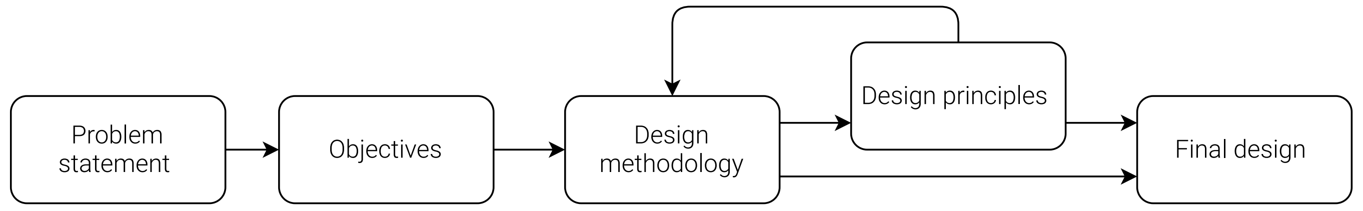


Figure 2.1 Design principles (own ill.)

2.4 RESEARCH QUESTION

The main research question can be divided into multiple sub-questions to answer the main research question better. Some background questions are answered to gain more insight into the current situation and possible potentials.

2.4.1 BACKGROUND QUESTIONS

- How is waste wood being processed in the Netherlands?
- Who are the stakeholders in the waste wood market in the Netherlands?
- What is the history of timber frame construction, and what can be learned from it?
- What are the requirements for building a wooden wall element in the Netherlands?

2.4.2 MAIN RESEARCH QUESTION

How can a database, a parametric model, and scripting be used to develop an exterior wall element from waste wood that minimizes the material loss and takes full benefit of the waste wood dimensions?

2.4.3 SUB QUESTIONS

The main research question can be divided into the following sub-questions:

1. How can a digital database with waste wood properties communicate with a parametric model of an exterior wall element?
2. How can the script minimize the additional wood waste with the selection procedure when selecting the waste wood from the database for the exterior wall element?
3. Which parts of the exterior wall element can be constructed with waste wood?
4. What design principles allow the exterior wall element to be constructed with waste wood while minimizing the additional material waste and allow for reuse after its end of life?
5. How can scripting guide the designer with the design process?

2.5 RESEARCH METHODOLOGY

The research is divided into different phases. There is a literature phase, a study phase, a designing phase, and a tool development phase.

2.5.1 LITERATURE PHASE

The literature phase is used to answer the background questions and create a solid foundation to answer the research question.
The process of waste wood in the Netherlands is researched and analyzed. Interviews are conducted with different stakeholders, and a waste wood processing facility is visited. Modern timber frame construction (TFC) methods and TFC methods from the past are analyzed to study what can be learned from them. The knowledge is used in the development of the exterior wall element that can work with waste wood. Within this thesis, timber frame construction (TFC) can be translated to the Dutch word Houtskeletbouw (HSB). To generate a realistic dummy database, research is done about the characteristics of wood relevant for construction. The building regulations in the Netherlands are analyzed to gain insight into the requirements that the wall element needs to fulfill.

After every chapter, a conclusion is given with relevant information used for design decisions later in this thesis.

2.5.2 STUDY PHASE

The study phase is about learning Python and SQL. These programming languages are required to build a functional database and parametric prototype. At the beginning of this thesis, there was no experience or knowledge in these languages.

2.5.3 DESIGNING PHASE

A design methodology guides the designer in finding the most suitable design solution for the given problems. In the used design methodology, criteria are drafted to test every design solution. Knowledge from the literature phase will help to define the criteria for the design.

A design methodology where the output of scripting is used within the design process does not exist. For this thesis, an existing methodology is modified, and the scripting is implemented to see if it can help the designer find the most suitable solution.

At the end of the design process, the altered methodology is reviewed and critically analyzed to determine the effectiveness and added value compared to the original design methodology.

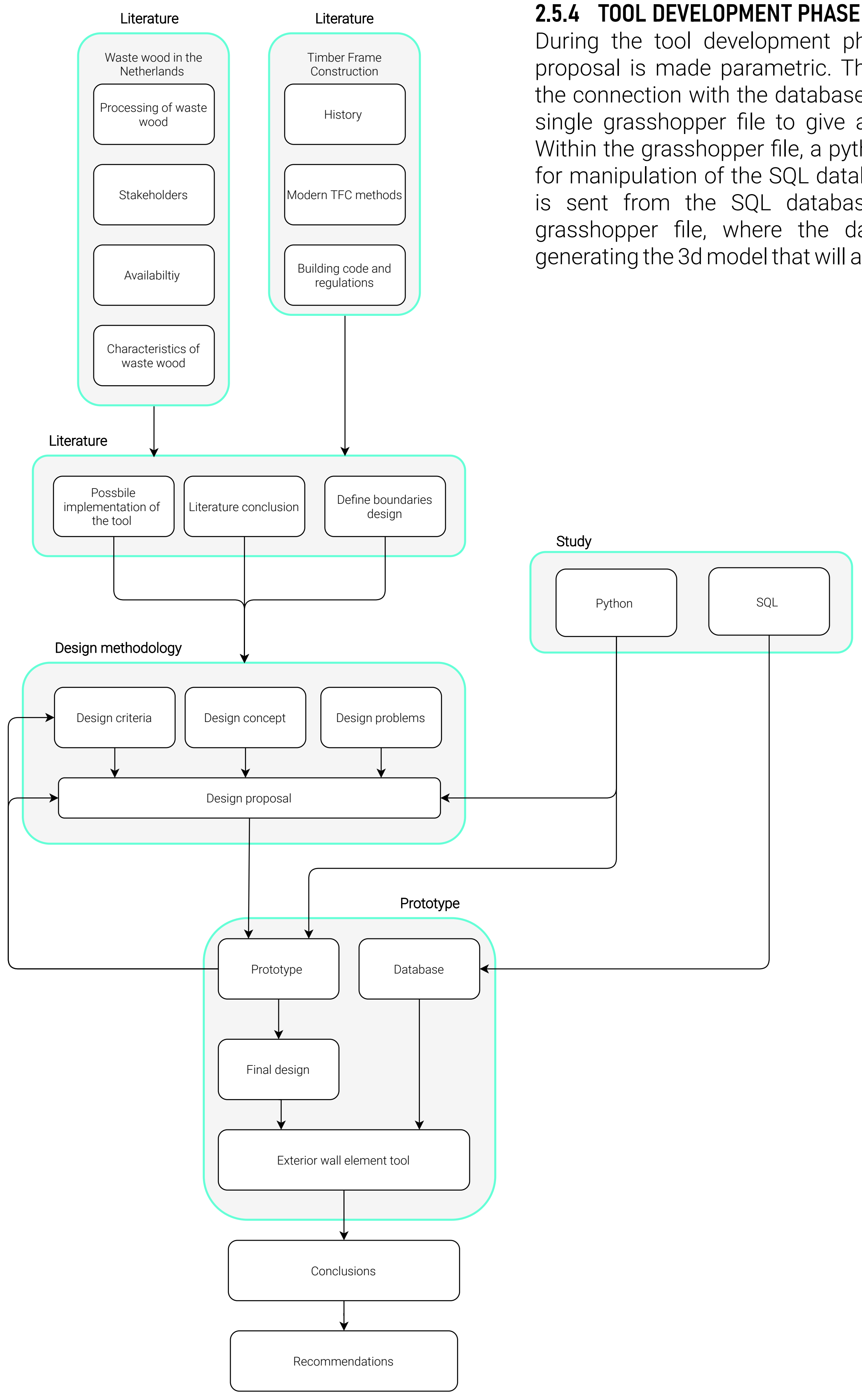


Figure 2.2 Research framework (own ill.)

2.5.4 TOOL DEVELOPMENT PHASE

During the tool development phase, the design proposal is made parametric. The prototype and the connection with the database are created in a single grasshopper file to give a clear overview. Within the grasshopper file, a python script allows for manipulation of the SQL database. The output is sent from the SQL database to the same grasshopper file, where the data is used for generating the 3d model that will act as a prototype.

3 WASTE WOOD

In this chapter the Dutch waste wood market is analyzed to discover what types of waste is suitable for reuse and how it moves trough the market and is processed. By doing this the scope of this graduation project could be further defined. In the end of the chapter a suggestion is made how the exterior wall element tool could be implemented into the Dutch waste wood market.

Sortiva / GP groot waste processing facility, Alkmaar (Bing maps)

3.1 WOOD WASTE INTRODUCTION

To develop a tool that can generate a wall element from waste wood, it is necessary to analyze the Dutch waste wood market. Knowing who the stakeholders are and how waste wood gets processed can help with implementing the tool. Besides the literature, interviews were conducted with Martijn Meuleman from van Werven Recycling and Gerald van Elburg from Nedvang. The processing location from GP Groot / Sortiva was visited in Alkmaar. The combination of literature, interviews, and a location visit resulted in a better understanding of the waste wood market.

In the Netherlands, there are three main components in the waste wood market:

- The demolisher
- The collector
- The processor

In the Netherlands, the most significant contributor in waste wood is the demolishing and building industry (Afman et al, 2014). A demolition company tears down a building and hires a collection company to pick up all the waste. The collection company sorts the different waste streams and brings them to a processing facility where the different types of waste get processed. In the Netherlands, most of the time, two or three components are integrated into one company. This combination results in a demolisher that can also sort and transports waste. Or a processing facility that also collects the waste. Sortiva and Renewi are examples of companies that collect waste and process it.

When waste wood is brought to a processing facility, it is sorted into different waste wood classes. Depending on the assigned class, the waste wood gets shredded for efficient transport and transported to different companies where the waste wood gets recycled into a new product or incinerated for bio energy.

3.2 WOOD CLASSIFICATION

In the Netherlands, waste wood is divided into three categories: A, B & C.

3.2.1 A-WOOD

A-wood consists of untreated, uncoated, and unpainted wood. Examples of A-wood are pallets, fruit crates, and wood waste from milling facilities. Wood from packaging makes up 80% of the A-wood; this includes pallets. Most of the A-wood in the Netherlands ends up on a pile with B-wood. When this happens, the A-wood becomes B-wood. No law demands that A-wood needs to stay separated from B-wood (Bruggen & Zwaag, 2017).

3.2.2 C-WOOD

C-wood is treated wood. This means the wood is treated with preservative chemicals. These chemicals extend the wood's lifecycle and protect wood fibers from structural degradation, decay fungi, termites, marine organisms, and flames. Due to these chemicals, the wood is contaminated and therefore cannot be recycled. The C-wood is chipped and transported to Germany, where they have the proper facilities to incinerate the C-wood and process the hazardous gases released within this process.

3.2.3 B-WOOD

B-wood is all the waste wood that is not classified as A- or C-wood. Processing facilities make a distinction between solid B-wood and non-solid B-wood and sometimes separate these streams. This separation is not required by law, but the solid B-wood can be sold for a higher price due to the wood's higher quality. An example of solid B wood is a painted beam from an old barn, and an example of non-solid B wood is a particleboard. A particle board is labeled as non-solid B wood because of the adhesives. Solid B-wood is the class of wood that has the potential to be reused as a building material.

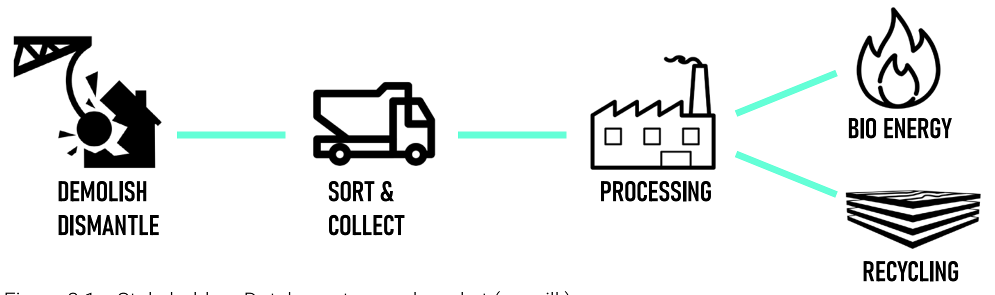


Figure 3.1 Stakeholders Dutch waste wood market (own ill.)

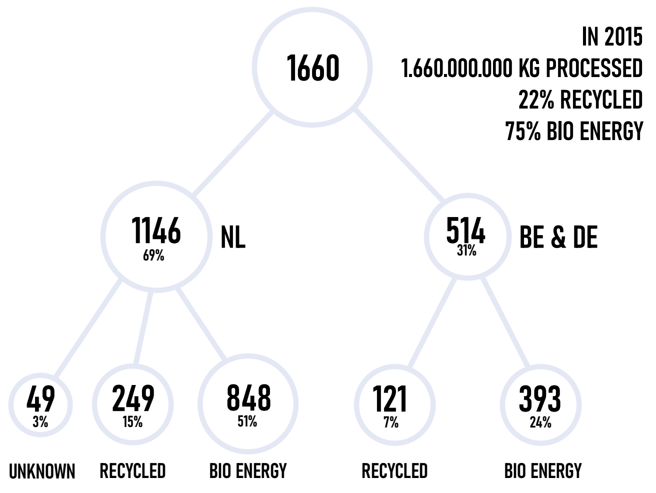
3.3 THE MARKET

Waste wood gets also imported and exported from and to other European countries. This makes the Dutch waste wood market quite complicated. Figure 3.2 shows all the different waste wood streams in 2015.

3.3.1 2015

These numbers are not an estimation but are the registered data by the government. The absence of the A-wood stream is because it is not required by law to register this waste wood stream. If A-wood ends up on a pile with B-wood, the A-wood is categorized as B-wood. In 2015 1.763.000.000 kg waste wood entered the Dutch market. 1.482.000.000 kg (84%) came from the Netherlands, and 281.000.000 kg (16%) was imported. The imported wood came from the UK (52%), Belgium (33%), Germany (8,5%), and Norway (5,5%) (Bruggen & Zwaag, 2017).

The waste wood market is volatile with low-profit margins. Therefore, waste wood from other countries may be more profitable than waste wood from the Netherlands. For example, in England, an additional tax was implemented for the deposition of waste wood. This tax resulted in waste wood being exported to the Netherlands because this was cheaper than disposing it on a landfill in the United Kingdom. This explains the high amount of imported wood from the UK in 2015.



In 2015 1.660.000.000 kg waste wood was processed. The waste wood facilities prepare the waste wood for transportation by shredding the wood in chips. Each incineration and recycling facility has unique preferences in the mixture of waste wood they want to receive. This mixture is a combination of A-wood and solid B-wood.

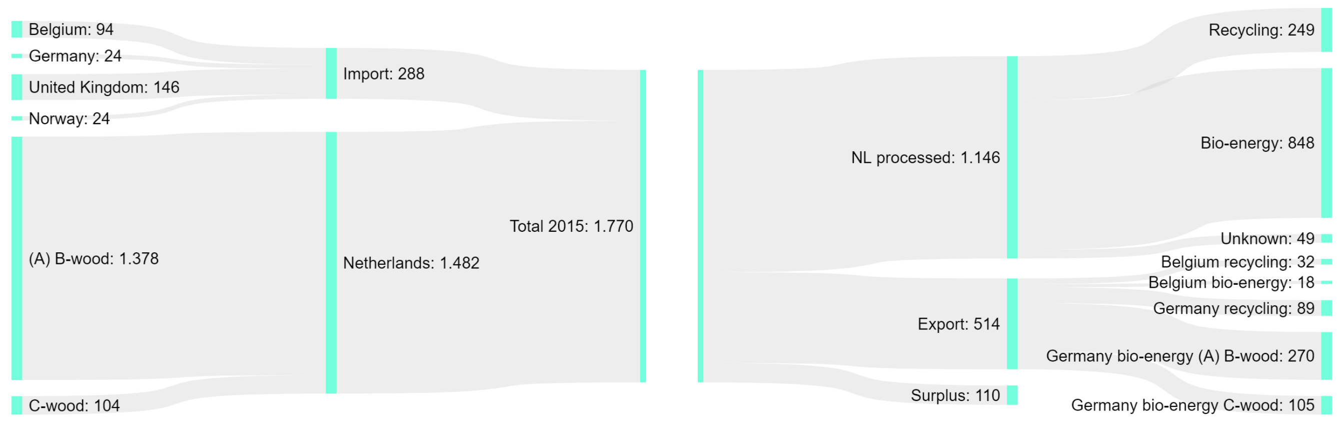


Figure 3.2 Dutch waste wood market in 2015, data by TAUW (own ill.)

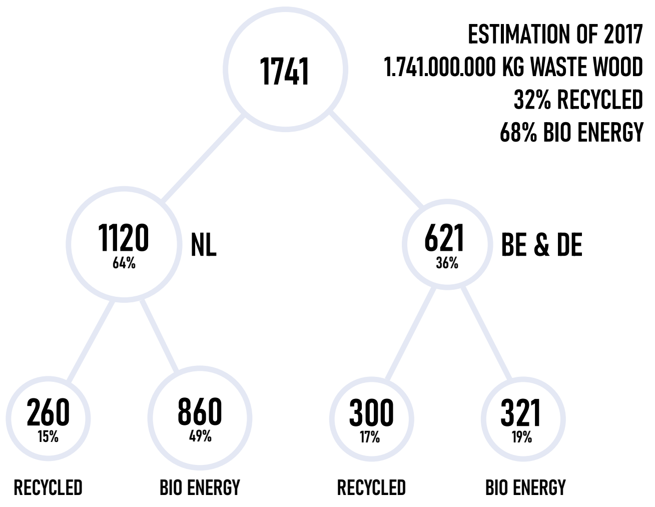
3.3.2 2017

Bruggen & Zwaag estimated the processed wood in 2017. This estimation is based on interviews with stakeholders in the Dutch waste wood market.

In total, there was 1.610.000.000 kg of waste wood collected in the Netherlands.

- 250.000.000 kg (15,5%) as A-wood
- 1.260.000.000 kg (78,3%) as B-wood
- 370.000.000 (23%) as solid B-wood
- 890.000.00 (55,3%) as non-solid B-wood
- 100.000.000 kg (6,2%) as C-wood.

The estimation shows that the amount of waste wood exported to Belgium and Germany for recycling has more than doubled than in 2015. In the estimation of 2017, the mono stream of A-wood is taken into account while it is not shown on the officially registered data from 2015; this explains that specific difference. (Bruggen & Zwaag, 2017). On page 24 & 25 a clear overiew of the estimated waste wood streams can be seen.



3.4 RECYCLING AND INCINERATION

Almost all processed wood that gets recycled in the Netherlands is used to produce pallet blocks.

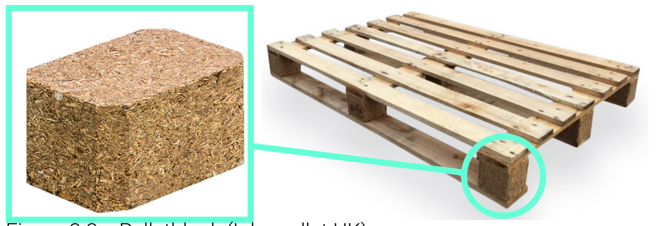


Figure 3.3 Palletblock (Inka pallet UK)

All the processed C-wood is exported to Germany, where it is incinerated. The Netherlands does not have the facilities to incinerate this class of contaminated wood.

The Dutch incineration facilities does not have the capacity to incinerate all the available non-solid B waste wood. Therefore, a significant amount of waste wood gets transported to Belgium and Germany, where the surplus is incinerated. This transportation results in additional CO2 emissions. The Dutch recycling facilities also do not have the capacity to recycle all the waste wood suitable for recycling. Again, a significant amount of waste wood is transported to Belgium and Germany, where the surplus is used for the production of engineered boards. An engineered board is manufactured by compressing layers of wood chips or wood flakes with a binding adhesive. Examples of engineered boards are:

- Oriented Stranded Board (OSB)
- Particleboard
- Medium Density board (MDF)

The downside of creating engineered boards or pallet blocks is that they cannot be recycled again due to the adhesives used to create this product. The production from waste wood to engineered boards and pallet blocks is considered downcycling.

3.5 BOTTLENECKS IN THE WASTE WOOD MARKET

The TAUW report published some interesting bottlenecks about the Dutch waste wood market that is worth mentioning.

3.5.1 A-WOOD INCINERATION SUBSIDIZATION

80.000.000 kg of A-wood in the Netherlands gets incinerated for bioenergy instead of being downcycled to an engineered board. The reason for this is the subsidization from the government for bioenergy installations. With this subsidization, the bioenergy installations can offer more money for the A-wood than production companies of engineered boards are willing to offer Bruggen & Zwaag, 2017). Stopping this subsidization will result in more A-wood being processed to engineered board and less A-wood being incinerated. This will help the circular economy because downcycling is at least better than incineration.

3.5.2 DOWNCYCLING CAPACITY

There is not enough capacity in the recycling factories to process all the waste wood in the north-west European waste wood market suitable for recycling. Governments should stimulate new and innovative ways of recycling to increase the recycling capacity.

3.5.3 SEPARATE SOLID B-WOOD FROM B-WOOD

It is not required by law to separate the solid B-wood from the B-wood stream. Therefore, it happens that processing facilities which not separate sell B-wood to incineration facilities that have solid B-wood in them, and consequently, it would be better to transform this into Engineered boards.

3.6 REAL-LIFE SCENARIO

To obtain more knowledge about the Dutch wood waste industry, an interview with Martijn Moleman from van Werven was conducted. The interview took place on the 7th of May 2020. Van Werven is a company that is active in demolishing, collecting, and processing. They offer the possibility to demolish circular. This circular demolishing means they carefully dismantle a building and reuse products where possible. They only do this if the client pays for it or if they have enough time to find a new destination before the dismantling begins.

During the interview, Martijn said that it regularly happens that reusable wood ends up being downcycled to engineered board or incinerated due to time pressure. For example, on the 30th of April 2020, they started the demolishment of a swimming pool in Dronten. This swimming pool had sixteen 20-meter-long beams/rafters that are suitable for reuse. However, they will not be reused because van Werven got the job two days before the demolition, which is not unheard of within the demolition industry. The beams got saw into pieces to make transportation easier and ended up on a pile with other B-wood.

Here the tool developed exterior wall tool can offer a solution for these rush jobs.

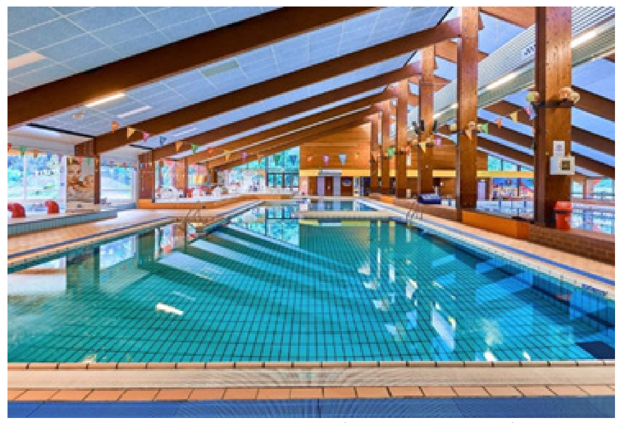


Figure 3.4 Swimmingpool dronten (omroep Flevoland)

3.7 LOCATION VISIT

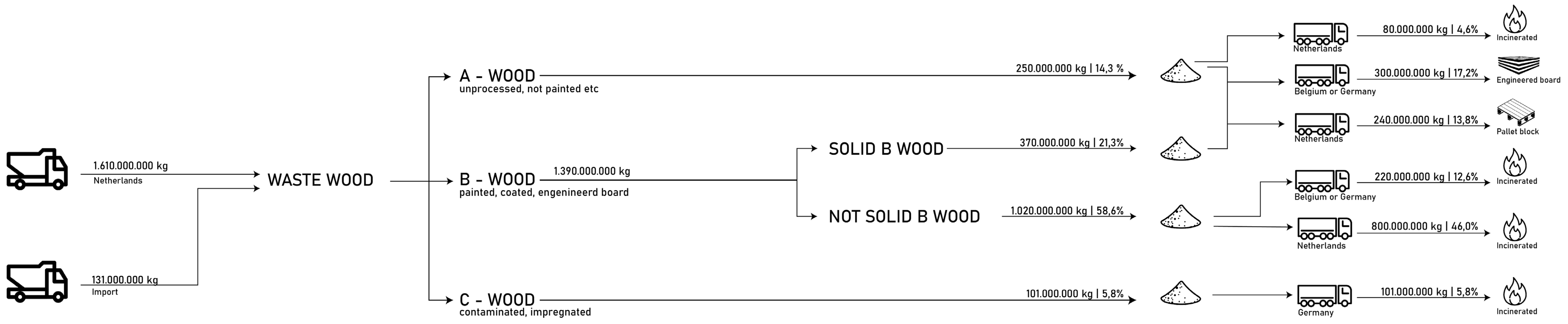
On August 26th, 2020, a visit was made to the processing facility of GP Groot / Sortiva in Alkmaar. Here Martine de Wit gave a tour and talked about a new circular pilot of GP Groot. With this pilot, old window frames were dismantled from houses in Amsterdam and modified to be reused for a business lobby's interior. The modification took place between all the waste wood on location. This pilot was more something of a prestige project, a way for the company where the interior is used to say they participate in a circular project. The window frames were modified by hand, and therefore it is not a solution for a larger scale. Here the developed tool could help this process.

3.8 CONCLUSION OF THE WASTE WOOD MARKET

What conclusion can be taken from this analysis that helps with the implementation of the developed tool?

Solid B-wood has the potential to be reused as a building material. Estimation by TAUW stated that in 2017 there was 370.000.000 kg of solid B wood processed. What percentage of this solid B-wood can be reused as a building material is unknown. However, the conducted interviews and the visit to GP Groot / Sortiva proved that there is waste wood suitable for reuse.

A law forcing the sorting and registration of solid B-wood and non-solid B-wood would give more insight into the exact data of waste wood. It would prevent solid B-wood from being incinerated for bioenergy. Processing solid B-wood into engineered boards is still a better solution than incineration.



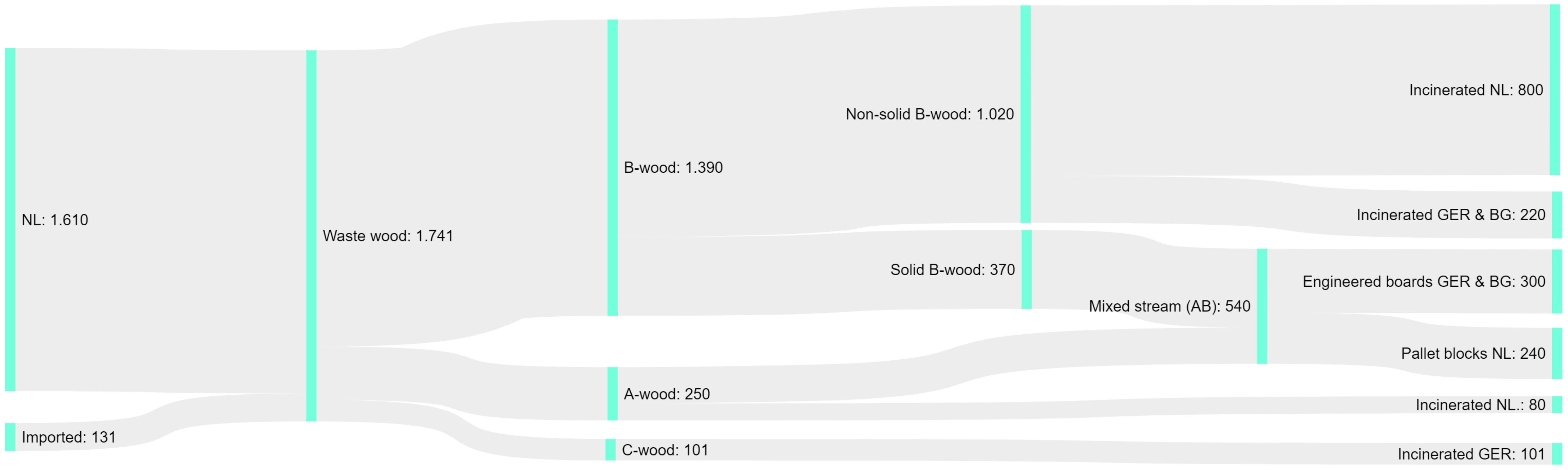


Figure 3.5 Estimation of the processed waste wood in 2017 in million kg, data by TAUW (own ill)

The recycling and incineration capacity of the Netherlands is not enough to process all the waste wood. Therefore, a significant percentage gets exported to Belgium and Germany. During the export, there is strong competition with other countries. It would be better if the Netherlands would not be that dependent on the international market to buy all the surplus of waste wood.

3.9 PROCESSING REUSABLE WASTE WOOD

In this section, a suggestion for a method is made how the reusable waste wood can get processed and work together with the exterior wall element tool.

The first part of the process starts with sorting the reusable solid B wood from the other waste wood. There are multiple ways to filter the reusable wood from the waste stream. The simplest option is to do it by hand. However, sorting, checking, measuring, weighing, and putting all the input in the database is manual, a very time-consuming process that is not feasible in the Netherlands due to high labor costs.

Another solution can be found in robotic sorting, photogrammetry, and machine learning.
Machine learning is an application of artificial intelligence that provides systems, the ability to learn and improve from experience without being explicitly programmed automatically. For example, showing pictures of cracks in wood can teach the machine to recognize these cracks by itself. This technique is already used in domestic waste sorting. Here the robot can detect the different types of waste and sort them in the designated recycling bin (Recycling today, 2019).

Figure 3.6 Robotic sorting of waste wood (recycling today)

Using photogrammetry with a grid-like backdrop or with scanning, the dimensions of the waste wood can be obtained. Machine learning, combined with these methods, can detect the structural integrity of the wood and if the wood is painted. There are more suitable options available for this step in the process. However, due to the given time frame, this is not a part of this thesis.
The obtained characteristics of a piece of reusable waste wood are sent to an SQL database. The physical piece of wood is stored until it is selected for the exterior wall element. It is crucial that the piece of wood is tagged with an identification number that correlates with an identification number in the database. Otherwise, it is difficult to find the correct piece of wood when assembling the exterior wall element.

The database communicates with the exterior wall element tool. When an exterior wall element is needed, the tool searches in the database for the most suitable pieces of wood and creates for every piece of wood selected a separate 3d file that can be used for digital fabrication. The tool will show a list of all the identification numbers of the pieces of wood required so that the correct pieces of wood can be selected from the storage for digital fabrication.

After modifying all the selected pieces of waste wood, the exterior wall element can be assembled. The scheme below shows an overview of this process.

Within this thesis, the focus lies on the development of the exterior wall element tool, developing a realistic design for the exterior wall element, and the communication between the database and the parametric model (figure 3.7).

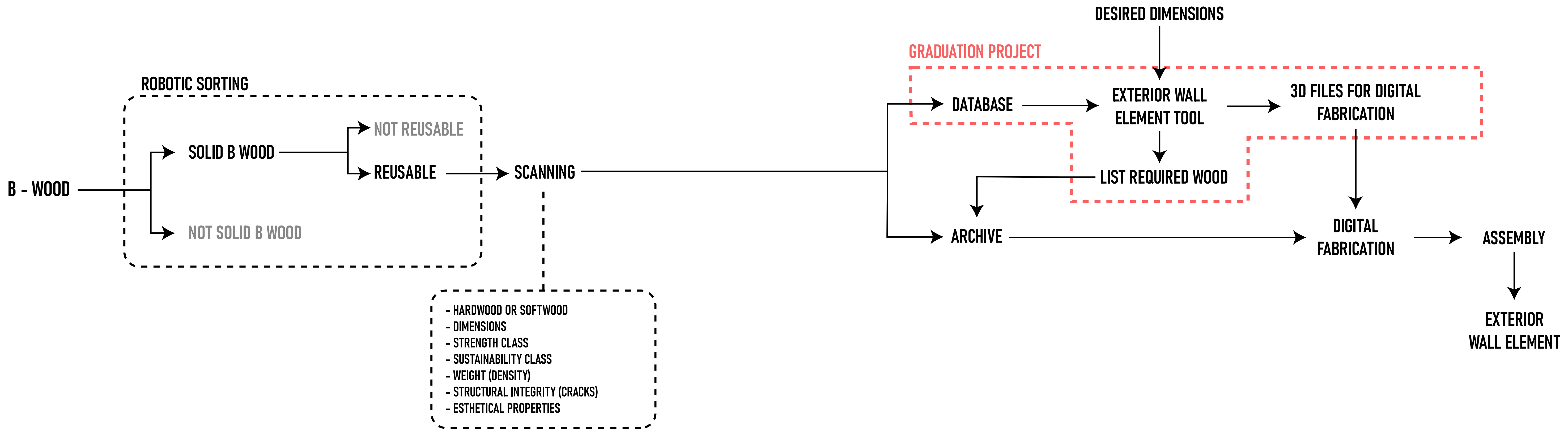


Figure 3.7 Scope of research (own ill)

3.10 IMPLEMENTATION PROPOSAL

With this chapter's knowledge, the conducted interviews with Martijn Meuleman from van Werven Recycling and Gerald van Elburg from Nedvang and the processing location visit in Alkmaar; a thought experiment can be conducted about a possible implementation of the exterior wall element tool.

With the increasing demand for circular building materials due to the Dutch government's new regulations and the need for one million additional houses, demolition companies started to dismantle instead of demolishing. The dismantled products are being sold in the local shop of the demolisher or online at their website. With the exterior wall element tool, it is also possible to register the wood they dismantled in the database. This way, a national database is realized with sellers of waste wood in the Netherlands. These sellers can be the demolishing companies but also the waste wood processors. A contractor can use the exterior wall element tool online to create a wall element from waste wood. Here the user, in this case the contractor, gives the desired dimensions of the wall, and the tool generates a design of an exterior wall element. Additional information is given about the cost, the amount of wood required and where the wood is located. The contractor can adjust the parameters of the model and choose, for example, the cheapest option or the option where all the wood is closest to the building side. In the end the tool generates a list with 3dm files for every piece of wood that has to be modified. These 3dm files can be used for digital fabrication. It is not difficult to implement more building elements in the tool. This can create a positive spiral where more people use the exterior wall element because more building elements are implemented. An additional benefit is the additional data that is obtained about waste wood that is suitable for reuse.

A different possible implementation can be directly at the waste wood processor. With the increasing demand for circular building materials, the processor decides to focus on reusing waste wood instead of shredding the wood and selling it to recycling or bioenergy facilities. With the reusable wood, the processor wants to produce wall elements. Next to the waste wood facility, a workspace and storage are designed. The reusable waste wood is temporarily placed in the storage where the wood is labelled. The characteristics and the label number are stored in a digital database. The workspace allows for assembly of the exterior wall element and robotic manufacturing such as CNC milling. This way, the waste wood processor can sell the complete wall element or just the waste wood they have registered to the database. Most of the time, a waste processor also collects waste. Therefore, the logistics and equipment needed for transporting the wall elements are already available. The same goes for the space required to store the reusable waste wood.

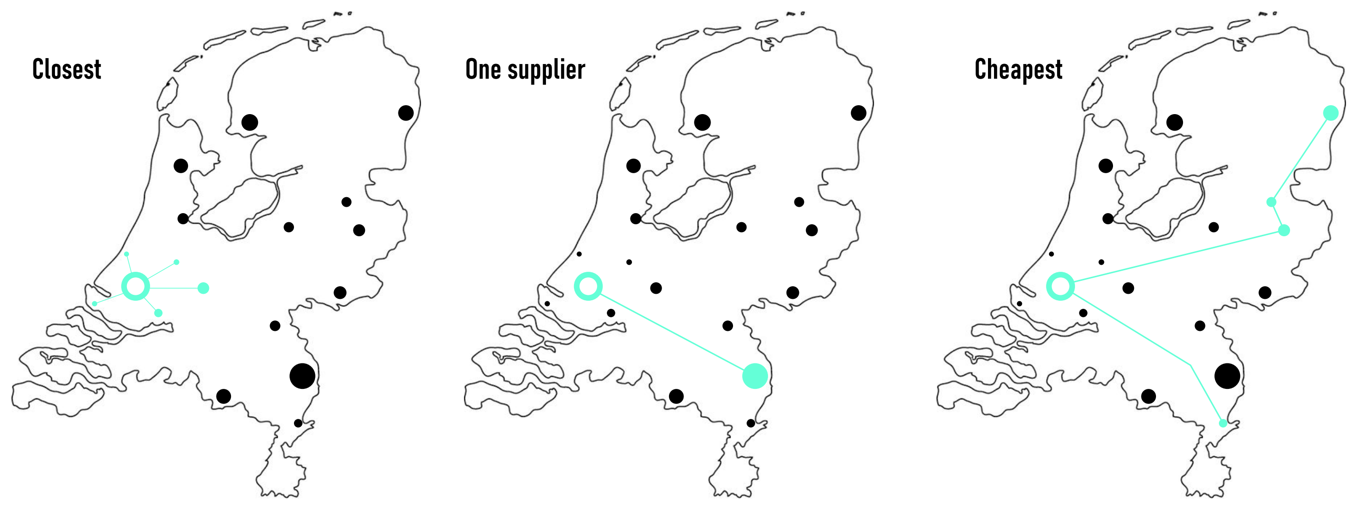
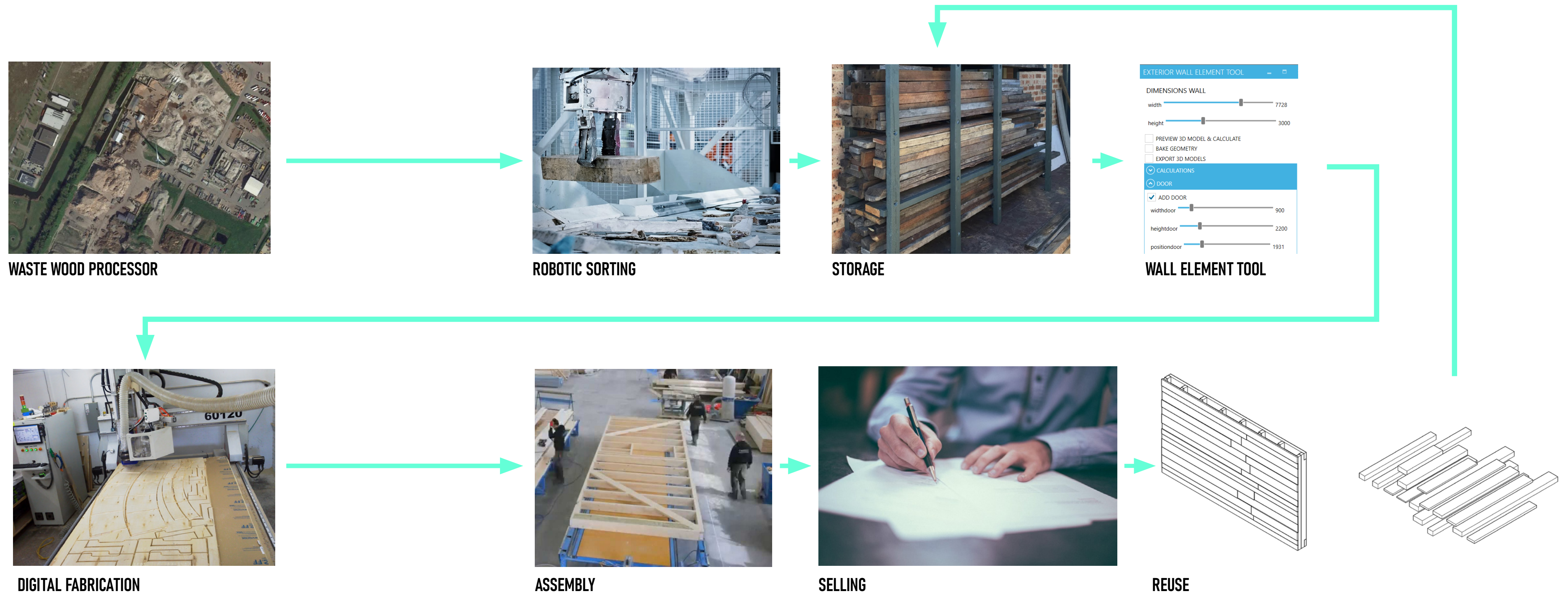


Figure 3.8 National database with waste wood (own ill)



EXTERIOR WALL ELEMENT TOOL

DIMENSIONS WALL

width 7728

height 3000

PREVIEW 3D MODEL & CALCULATE

BAKE GEOMETRY

EXPORT 3D MODELS

CALCULATIONS

DOOR

✓ ADD DOOR

widthdoor 900

heightdoor 2200

positiondoor 1931

Figure 3.9 Implementation suggestion at the waste wood processor

4 DIGITAL DATABASE

The previous chapter analyzed the Dutch waste wood market and showed that there is no database or data available about the characteristics of waste wood in the Netherlands. This chapter will focus on the database and suggest how this database can be constructed.

Used beams (Archdaily)

4.1 WASTE WOOD CHARACTERISTICS

In the previous chapter, a possible implementation of the database combined with the exterior wall element tool in the Dutch waste wood market was suggested. This chapter will focus on the database and suggest how this database can be constructed. A database with waste wood does not already exist in the Netherlands. There is also no official registered data in the Netherlands available about the sizes of the waste wood suitable for reuse. To better understand the possible sizes of suitable waste wood, market research is performed by looking at online sellers of reusable wood.

The database should consist of two tables. One table should give an overview of the characteristics of the waste wood that are relevant for the exterior wall element tool. The Python script can use these characteristics to select the most suitable pieces of wood. These characteristics can be the dimensions but also if a piece of wood is cracked for example. Cracked cannot have a load bearing function, while it still can be applied in the cladding. The second table should contain information about the seller of the waste wood, such as the name of the company, city, address, etc. These two tables allow the user to filter the wood that matches their preferences, such as wood from one seller or inside a certain radius of the construction site.

During the literature research, the following characteristics of waste wood were found essential for the construction of an exterior wall element. Therefore, these characteristics should be included in the database.

- Dimensions
- Weight
- Hardwood or softwood
- Sustainability class
- Strength class
- Structural integrity
- Esthetical properties
- Moisture level

4.1.1 DIMENSIONS

The dimensions are one of the most essential characteristics to add to the database. To prevent additional waste, the script selects wood that matches the dimensions of the required wood.

4.1.2 WEIGHT

combining the weight and the dimensions can determine the density of the wood. The density can indicate the type of wood. Adding the wood's weight to the tool can give an insight into the total weight of the construction.

4.1.3 HARDWOOD OR SOFTWOOD

Wood can be divided into two categories, Hardwood (loofhout) and softwood (naaldhout). Hardwood is wood from dicot trees and is deciduous. Deciduous means that the tree will lose its leaves during the fall. An exception is hardwood in the tropics and subtropics, where it is evergreen. Softwood is wood from gymnosperm trees such as pines and spruces. 77% of the logs processed in Europe in 2019 were softwood (United Nations, 2020). The term hardwood and softwood can be confusing because it says nothing about the hardness or the wood's softness. However, on average, softwood is less dense than hardwood.

4.1.4 SUSTAINABILITY CLASSES

The European code EN 350-2 defines wood in different sustainability classes. These classes represent the natural durability of wood. The most known sustainability class is against fungus/mold.

There are methods to increases the wood's sustainability class. When wood is thermally modified, its sustainability class is increased in an environmentally friendly method. It can also be done by injecting chemicals, though this is not environmentally friendly and the wood cannot be recycled or reused again after its end of life. Wood with a sustainability class of I – III is suitable for outdoor functions.

4.1.5 STRENGTH CLASS

The EU code EN 338 defines the different strengths within timber. Unlike the sustainability class, the strength class is not defined based on the type of tree. There is also a distinction between hardwood and softwood. To perform relevant calculations about the structural capacity of the waste wood, it is important to know the strength class of the wood.

Timber strength grading in Europe is based on three key grade determining properties: strength, stiffness, and density. The grading is done in two

ways; there is visual strength grading and machine strength grading. With visual strength grading, the wood is graded based on the grader's experience to assess each piece of wood according to rules that define the size type and the number of strengths reducing characteristics allowed in each grade. These strength-reducing characteristics include knots, wane, and the slope of the grain. Machine grading is performed by a machine based on the relationship between strength and stiffness (Woodcampus, 2017).

4.1.6 STRUCTURAL INTEGRITY

The structural integrity determines if the waste wood is still suitable for a load bearing or structural function. Wood with cracks can still be used for a different purpose, such as the exterior cladding.

4.1.7 ESTHETICAL PROPERTIES

Esthetical properties are relevant if the wood is visible. If one side of the wood suitable for cladding is painted, it is preferable to face this piece in a direction so that it is not visible.

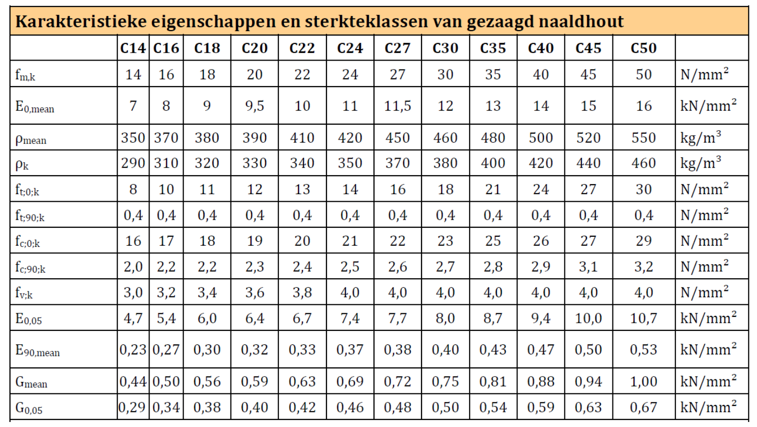
4.1.8 MOISTURE LEVEL

If a piece of wood exceeds a certain moisture level, the weight and dimensions can change during the drying process. In the Netherlands, wood for construction should have a moisture level of around 14-18%. The moisture level can be easily measured with a small device. The device to measure this moisture level is simply pressed against the wood (Houtinfo, 2013).

Sustainability classes for wood (NEN-EN 350-2 Norm)

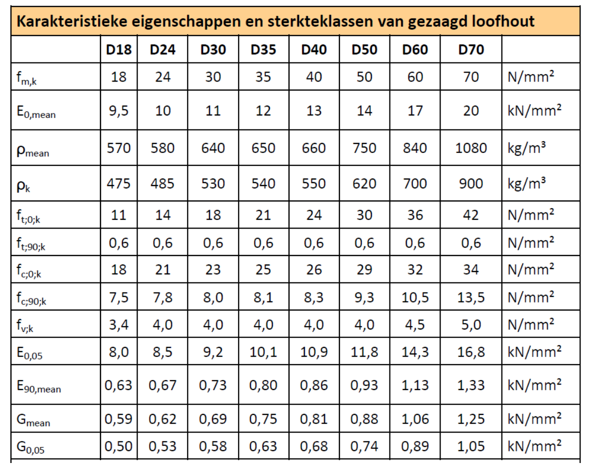
Sustainability classes		Lifespan (years), conditions		Wood types
		Protected wood in contact with damp soil	Unprotected wood outdoors	
1	Very sustainable	Lower than 25	50	Iroko, red cedar, teak, rosewood
2	Sustainable	15 - 25	40 - 50	Chestnut, larch, meranti, mahogany, robinia
3	Moderately sustainable	10 - 15	25 - 40	Oak, walnut, American red fir, oregon pine
4	Not really sustainable	5 - 10	15 - 25	Balsa, birch, elm, linden, parana pine, fir
5	Not sustainable	Lower than 5	6 - 12	Ash, plane, poplar, willo

Figure 4.1 Sustainability classes wood (Design for Environmental Sustainability: Life Cycle Design of Products (p132)



Karakteristieke eigenschappen en sterkteklassen van gezaagd naaldhout

	C14	C16	C18	C20	C22	C24	C27	C30	C35	C40	C45	C50	
$f_{m,k}$	14	16	18	20	22	24	27	30	35	40	45	50	N/mm²
$E_{0,mean}$	7	8	9	9,5	10	11	11,5	12	13	14	15	16	kN/mm²
ρ_{mean}	350	370	380	390	410	420	450	460	480	500	520	550	kg/m³
ρ_k	290	310	320	330	340	350	370	380	400	420	440	460	kg/m³
$f_{t;0;k}$	8	10	11	12	13	14	16	18	21	24	27	30	N/mm²
$f_{t;90;k}$	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	N/mm²
$f_{c;0;k}$	16	17	18	19	20	21	22	23	25	26	27	29	N/mm²
$f_{c;90;k}$	2,0	2,2	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3,1	3,2	N/mm²
$f_{v;k}$	3,0	3,2	3,4	3,6	3,8	4,0	4,0	4,0	4,0	4,0	4,0	4,0	N/mm²
$E_{0,05}$	4,7	5,4	6,0	6,4	6,7	7,4	7,7	8,0	8,7	9,4	10,0	10,7	kN/mm²
$E_{90,mean}$	0,23	0,27	0,30	0,32	0,33	0,37	0,38	0,40	0,43	0,47	0,50	0,53	kN/mm²
G_{mean}	0,44	0,50	0,56	0,59	0,63	0,69	0,72	0,75	0,81	0,88	0,94	1,00	kN/mm²
$G_{0,05}$	0,29	0,34	0,38	0,40	0,42	0,46	0,48	0,50	0,54	0,59	0,63	0,67	kN/mm²



Karakteristieke eigenschappen en sterkteklassen van gezaagd loofhout

	D18	D24	D30	D35	D40	D50	D60	D70	
$f_{m,k}$	18	24	30	35	40	50	60	70	N/mm²
$E_{0,mean}$	9,5	10	11	12	13	14	17	20	kN/mm²
ρ_{mean}	570	580	640	650	660	750	840	1080	kg/m³
ρ_k	475	485	530	540	550	620	700	900	kg/m³
$f_{t;0;k}$	11	14	18	21	24	30	36	42	N/mm²
$f_{t;90;k}$	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	N/mm²
$f_{c;0;k}$	18	21	23	25	26	29	32	34	N/mm²
$f_{c;90;k}$	7,5	7,8	8,0	8,1	8,3	9,3	10,5	13,5	N/mm²
$f_{v;k}$	3,4	4,0	4,0	4,0	4,0	4,0	4,5	5,0	N/mm²
$E_{0,05}$	8,0	8,5	9,2	10,1	10,9	11,8	14,3	16,8	kN/mm²
$E_{90,mean}$	0,63	0,67	0,73	0,80	0,86	0,93	1,13	1,33	kN/mm²
G_{mean}	0,59	0,62	0,69	0,75	0,81	0,88	1,06	1,25	kN/mm²
$G_{0,05}$	0,50	0,53	0,58	0,63	0,68	0,74	0,89	1,05	kN/mm²

Figure 4.2 Strength class for softwood (left) & hardwood (right), (houtinfo)

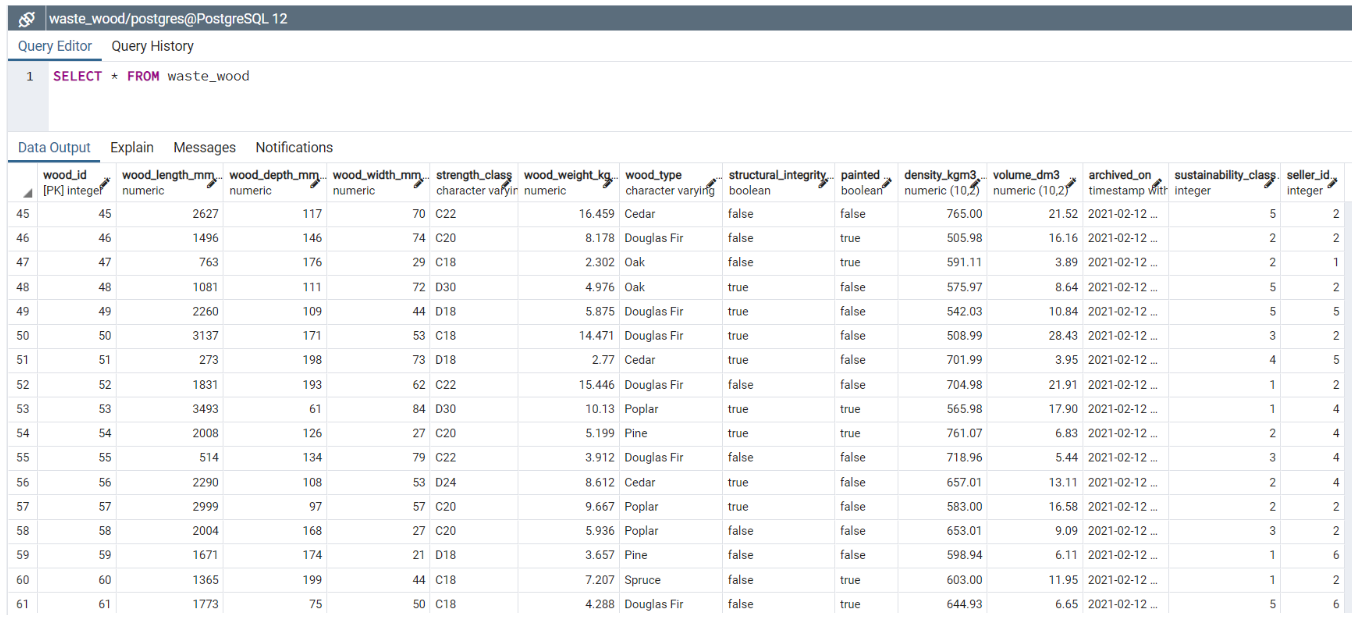
4.2 REUSABLE WASTE WOOD

The number of initiatives to reuse waste wood grew within the last decade. Demolishers started to see the value of the products that can be dismantled from a building. Some demolishers have a physical shop on their terrain where they sell the dismantled products such as windows, window frames, doors, etc. There also emerged online initiatives. Websites as gebruiktebouwmaterialen.com and bloemgebruiktebouwmaterialen.nl are offering a larger number of products that are dismantled from buildings. Demolishers can bring their dismantled products to these companies and the website will resell these products to for example, furniture makers or contractors. Marktplaats.nl also offers a large amount of waste wood that is being sold in smaller quantities by varies parties. To create a representative dataset for the database these websites are analyzed to obtain the dimensions of waste wood that can be used for the dataset.

During an email conversation with an employee from Bloem Gebruikte Bouwmaterialen some relevant knowledge was obtained. Between January 2020 and September 2020, around 4000m of wood had arrived with varying lengths between 300 and 510 cm. The cross-section of this wood was in the following dimensions (mm): 50x150, 65x165, 70x195, 75x210 and 70x220. When new wood arrives, it is not measured. It is only sorted on cross-section. Therefore, it is unclear to know the exact lengths of all the reusable waste wood they have in stock. Implementing the developed tool would require for the wood to be measured. Having a better understanding of the stock can also result in more sales because contractors can see which dimensions are available.
The mail conversation and the online research conclude that the waste wood can have a length of 5 meters. There is no restriction on the minimum size. Bloem Gebruikte Bouwmaterialen does not sell anything below 3 meters because it is not profitable. Implementing the developed tool can make this wood under 3 meters profitable in the future.

4.3 DUMMY DATABASE

For this thesis, a suggestion is made of how such a waste wood database can look like. The database is created so that different sellers of reusable wood can store their wood in the database.
The decision was made to create this database in SQL instead of Excel. Excel would be suitable for this prototype, but when this database is used with all the reusable waste wood for the dataset, the dataset's size will be too large, and Excel will not handle it properly. The SQL database is created with PostgreSQL. There was no specific reason for this other than the SQL course that was followed used PostgreSQL. Other SQL languages would also be suitable for creating this database. A python script is used to generate a fake dataset. The dataset is only used to set an example. Therefore, the wood type, strength class and density do not correlate.


waste_wood/postgres@PostgreSQL 12

Query Editor · Query History

```
SELECT * FROM waste_wood
```

Data Output · Explain · Messages · Notifications

	wood_id [PK] integer	wood_length_mm numeric	wood_depth_mm numeric	wood_width_mm numeric	strength_class character varyin	wood_weight_kg numeric	wood_type character varying	structural_integrity boolean	painted boolean	density_kgm3 numeric (10,2)	volume_dm3 numeric (10,2)	archived_on timestamp with	sustainability_class integer	seller_id integer
45	45	2627	117	70	C22	16.459	Cedar	false	false	765.00	21.52	2021-02-12 ...	5	2
46	46	1496	146	74	C20	8.178	Douglas Fir	false	true	505.98	16.16	2021-02-12 ...	2	2
47	47	763	176	29	C18	2.302	Oak	false	true	591.11	3.89	2021-02-12 ...	2	1
48	48	1081	111	72	D30	4.976	Oak	true	false	575.97	8.64	2021-02-12 ...	5	2
49	49	2260	109	44	D18	5.875	Douglas Fir	true	false	542.03	10.84	2021-02-12 ...	5	5
50	50	3137	171	53	C18	14.471	Douglas Fir	true	false	508.99	28.43	2021-02-12 ...	3	2
51	51	273	198	73	D18	2.77	Cedar	true	false	701.99	3.95	2021-02-12 ...	4	5
52	52	1831	193	62	C22	15.446	Douglas Fir	false	false	704.98	21.91	2021-02-12 ...	1	2
53	53	3493	61	84	D30	10.13	Poplar	true	true	565.98	17.90	2021-02-12 ...	1	4
54	54	2008	126	27	C20	5.199	Pine	true	true	761.07	6.83	2021-02-12 ...	2	4
55	55	514	134	79	C22	3.912	Douglas Fir	false	false	718.96	5.44	2021-02-12 ...	3	4
56	56	2290	108	53	D24	8.612	Cedar	true	false	657.01	13.11	2021-02-12 ...	2	4
57	57	2999	97	57	C20	9.667	Poplar	true	false	583.00	16.58	2021-02-12 ...	2	2
58	58	2004	168	27	C20	5.936	Poplar	false	false	653.01	9.09	2021-02-12 ...	3	2
59	59	1671	174	21	D18	3.657	Pine	false	false	598.94	6.11	2021-02-12 ...	1	6
60	60	1365	199	44	C18	7.207	Spruce	false	true	603.00	11.95	2021-02-12 ...	1	2
61	61	1773	75	50	C18	4.288	Douglas Fir	false	true	644.93	6.65	2021-02-12 ...	5	6

Figure 4.3 Screenshot from waste wood database in pgAdmin 4

The database consists of two tables. One table with all the waste wood characteristics and one table with the information from the different sellers. Both tables are relational tables, this means search queries can be performed over both tables. For example, it is possible to select all the pieces of wood that are longer than 3 meters (waste wood characteristics) and only sold in Alkmaar (seller information). For both tables to be relational with each other, they both require a primary key. The primary key is a column where the data inside that column is unique for every row. For the waste wood characteristics table this is the wood_id, and for the seller information table the seller_id. Linking the two primary keys makes the relation tables. In the waste wood characteristics table, the seller_id is then defined as the foreign key.

The waste wood characteristics table consist of the following columns:

- wood_id, primary key of the table
- wood_length_mm, length of the piece of wood in mm.
- wood_depth_mm, depth of the piece of wood in mm.
- wood_width_mm, width of the piece of wood in mm.
- strength_class, strength class of the wood with C for softwood and D for hardwood.
- wood_weight_kg, weight of the piece of wood in kg.
- wood_type, type of the piece of wood.
- structural_integrity, if the piece of wood can be used for structural purposes.
- painted, if the piece of wood is painted.
- density_kgm3, generated column with the density in kg/m3.
- volume_dm3, generated column with volume in dm3.
- Archived_on, timestamp on which day and time the piece of wood is archived.
- Sustainability_class, sustainability class ranging from 1 to 5, with 1 as most sustainable.
- Seller_id, shows which piece of wood is from which seller.

The wood ID is a unique number for every stored piece of wood. This id number must also be labelled on the archived wood. This ensure that the wood selected for manufacturing correlate with the wood from the exterior wall element tool.

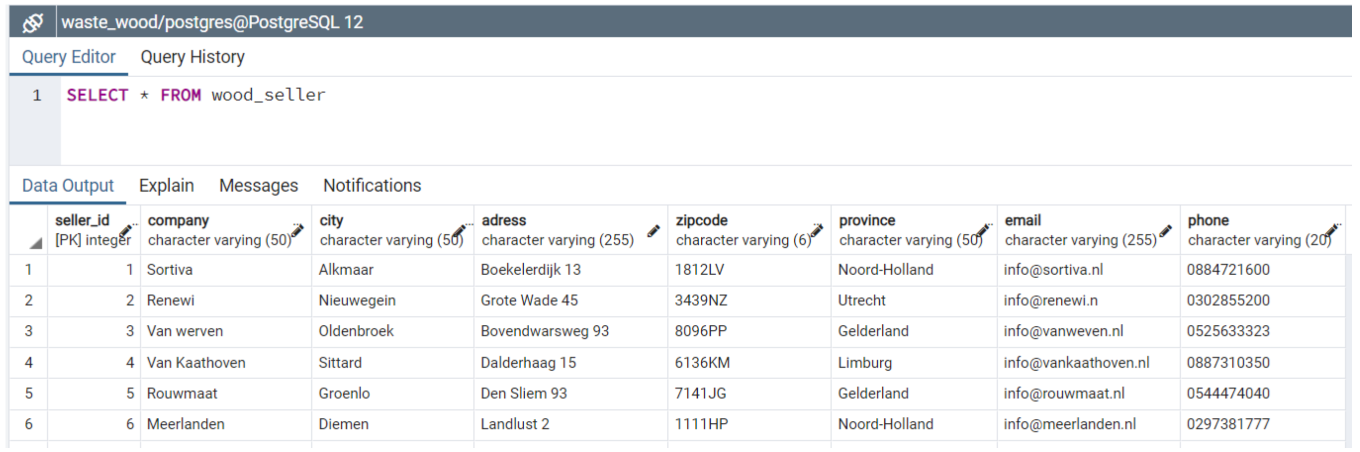
Length, depth, width, and weight are stored in a numeric datatype. The data from these characteristics are later used to create generated columns. The generated columns are created with an expression that require a decimal. Density and volume are the generated columns. A generated column is a special column that is computed from other columns. An example of a generated collumn can be seen in **figure 4.4**.

Structural integrity and painted are stored with a Boolean datatype, so either true or false. If the structural integrity is true, the piece of wood does not have any cracks and can be used for structural purposes. False means it cannot be used for structural purposes, but it is stills suitable for the cladding or substructure.

Seller_id is a foreign key correlating to the wood_ seller table. Here a dataset is entered from six waste wood processor that can sell reusable waste wood. Each piece of wood in the database is assigned to one of these waste wood processors.

```
(wood_weight_kg / (((wood_length_mm / (1000)::numeric) *
(wood_depth_mm / (1000)::numeric)) * (wood_width_mm / (1000)::numeric)))
```

Figure 4.4 Expression for a generated column that determines the density



waste_wood/postgres@PostgreSQL 12

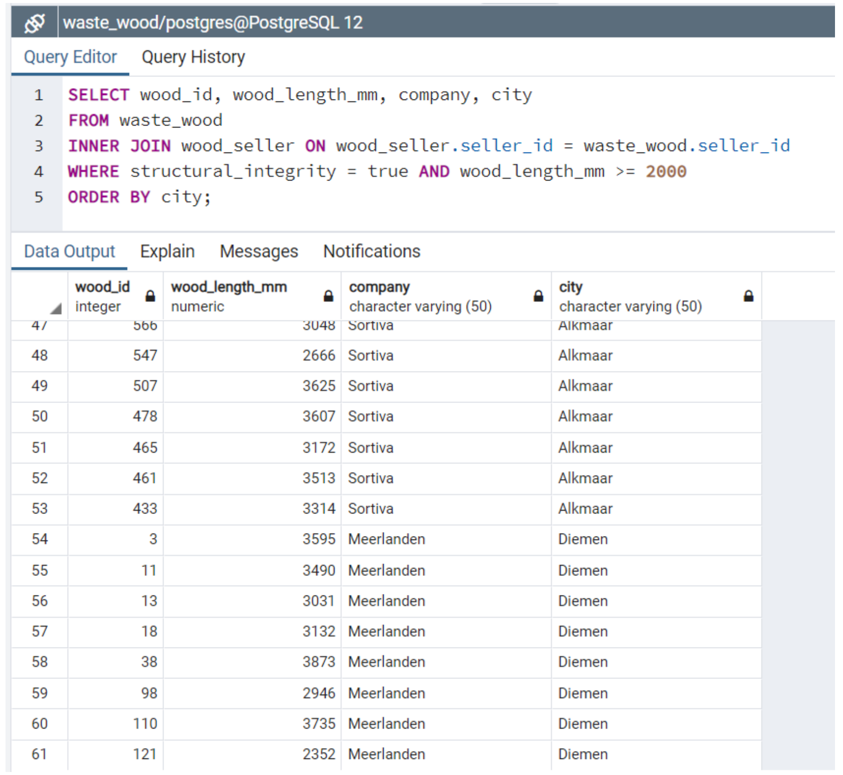
Query Editor | Query History

```sql
SELECT * FROM wood_seller
```

Data Output | Explain | Messages | Notifications

	seller_id [PK] integer	company character varying (50)	city character varying (50)	adress character varying (255)	zipcode character varying (6)	province character varying (50)	email character varying (255)	phone character varying (20)
1	1	Sortiva	Alkmaar	Boekelerdijk 13	1812LV	Noord-Holland	info@sortiva.nl	0884721600
2	2	Renewi	Nieuwegein	Grote Wade 45	3439NZ	Utrecht	info@renewi.n	0302855200
3	3	Van werven	Oldenbroek	Bovendwarsweg 93	8096PP	Gelderland	info@vanweven.nl	0525633323
4	4	Van Kaathoven	Sittard	Dalderhaag 15	6136KM	Limburg	info@vankaathoven.nl	0887310350
5	5	Rouwmaat	Groenlo	Den Sliem 93	7141JG	Gelderland	info@rouwmaat.nl	0544474040
6	6	Meerlanden	Diemen	Landlust 2	1111HP	Noord-Holland	info@meerlanden.nl	0297381777

With a search query, the wood can be found suitable for specific needs. In the images below, two examples are given.



waste_wood/postgres@PostgreSQL 12

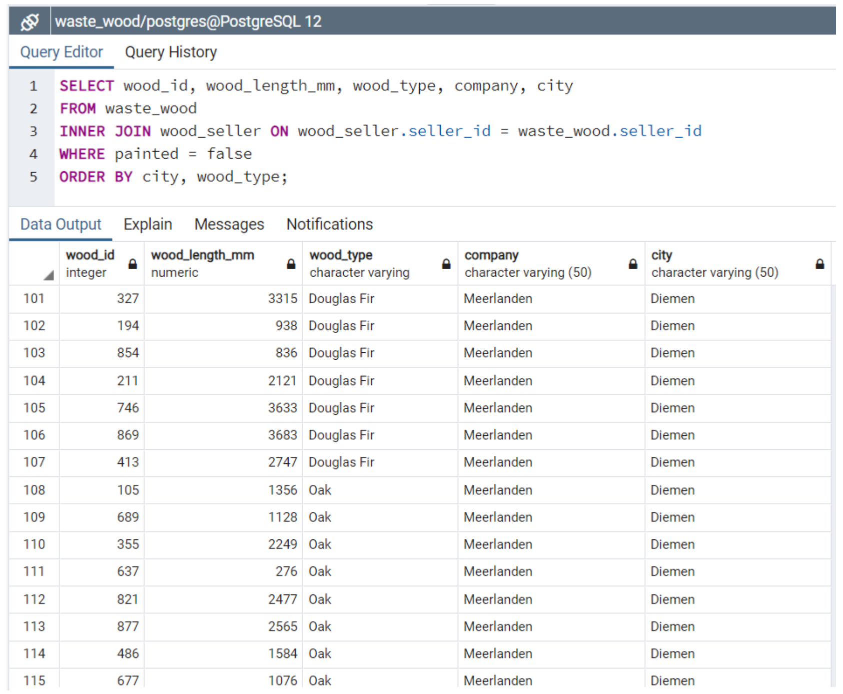
Query Editor | Query History

```sql
SELECT wood_id, wood_length_mm, company, city
FROM waste_wood
INNER JOIN wood_seller ON wood_seller.seller_id = waste_wood.seller_id
WHERE structural_integrity = true AND wood_length_mm >= 2000
ORDER BY city;
```

Data Output | Explain | Messages | Notifications

	wood_id integer	wood_length_mm numeric	company character varying (50)	city character varying (50)
47	566	3048	Sortiva	Alkmaar
48	547	2666	Sortiva	Alkmaar
49	507	3625	Sortiva	Alkmaar
50	478	3607	Sortiva	Alkmaar
51	465	3172	Sortiva	Alkmaar
52	461	3513	Sortiva	Alkmaar
53	433	3314	Sortiva	Alkmaar
54	3	3595	Meerlanden	Diemen
55	11	3490	Meerlanden	Diemen
56	13	3031	Meerlanden	Diemen
57	18	3132	Meerlanden	Diemen
58	38	3873	Meerlanden	Diemen
59	98	2946	Meerlanden	Diemen
60	110	3735	Meerlanden	Diemen
61	121	2352	Meerlanden	Diemen

This search query selects wood that is longer than two meters and is suitable for structural purposes. The wood_waste table and the wood_seller table are shown together with the inner join function. Inner join is possible because of the assigned primary and foreign key. This view allows the user to see where the wood is located.



waste_wood/postgres@PostgreSQL 12

Query Editor | Query History

```sql
SELECT wood_id, wood_length_mm, wood_type, company, city
FROM waste_wood
INNER JOIN wood_seller ON wood_seller.seller_id = waste_wood.seller_id
WHERE painted = false
ORDER BY city, wood_type;
```

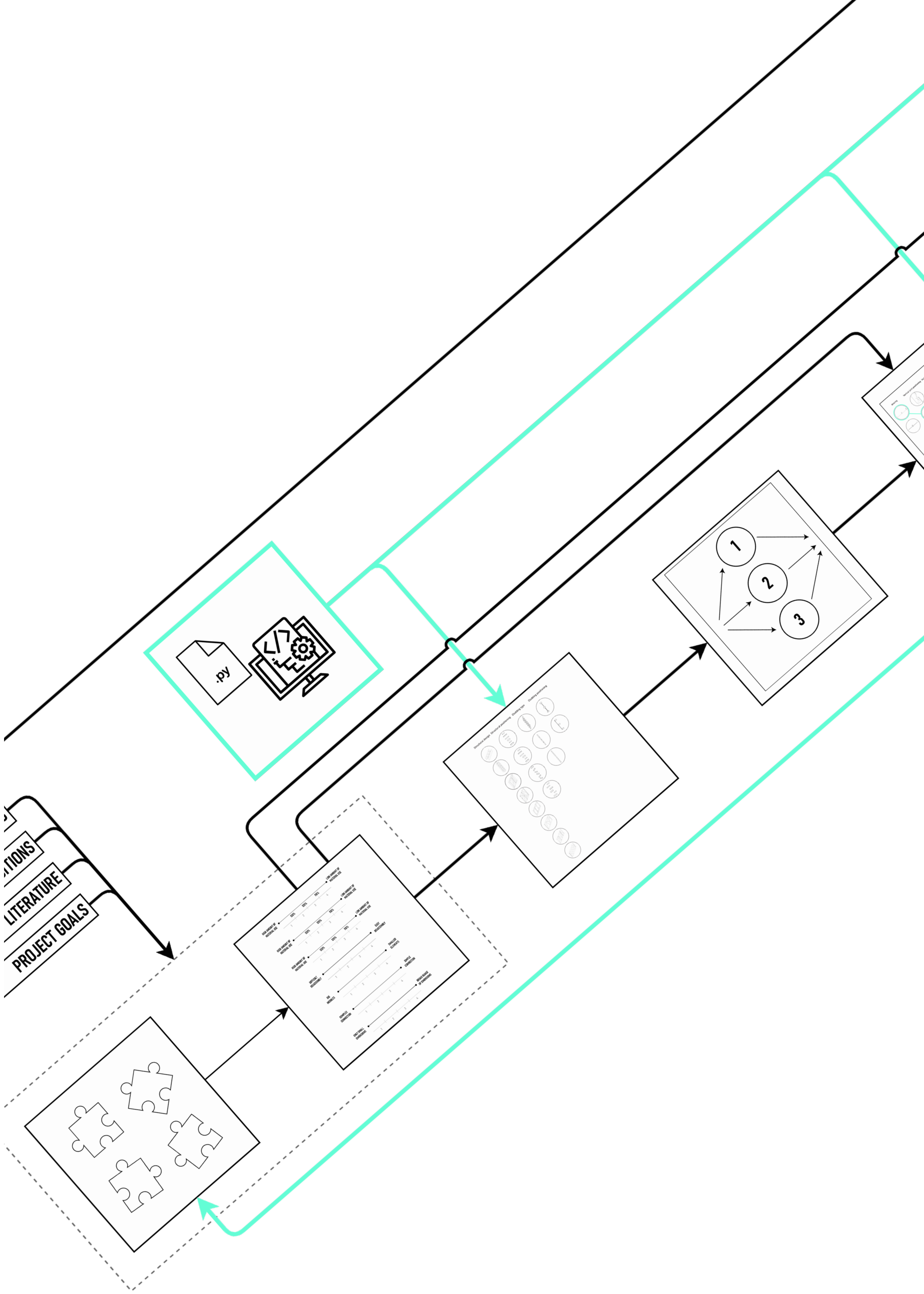
Data Output | Explain | Messages | Notifications

	wood_id integer	wood_length_mm numeric	wood_type character varying	company character varying (50)	city character varying (50)
101	327	3315	Douglas Fir	Meerlanden	Diemen
102	194	938	Douglas Fir	Meerlanden	Diemen
103	854	836	Douglas Fir	Meerlanden	Diemen
104	211	2121	Douglas Fir	Meerlanden	Diemen
105	746	3633	Douglas Fir	Meerlanden	Diemen
106	869	3683	Douglas Fir	Meerlanden	Diemen
107	413	2747	Douglas Fir	Meerlanden	Diemen
108	105	1356	Oak	Meerlanden	Diemen
109	689	1128	Oak	Meerlanden	Diemen
110	355	2249	Oak	Meerlanden	Diemen
111	637	276	Oak	Meerlanden	Diemen
112	821	2477	Oak	Meerlanden	Diemen
113	877	2565	Oak	Meerlanden	Diemen
114	486	1584	Oak	Meerlanden	Diemen
115	677	1076	Oak	Meerlanden	Diemen

This search query selects wood that is not painted

5 DESIGN METHODOLOGY

To design an exterior wall element that can be made parametric and work with the exterior wall element tool, a design methodology was followed. The design methodology was used to identify the design problems, determine the criteria, and guide the designer in finding the most suitable design that can work with the challenges of waste wood. In this chapter the chosen design methodology is elaborated and altered to see if scripting can improve this design methodology.



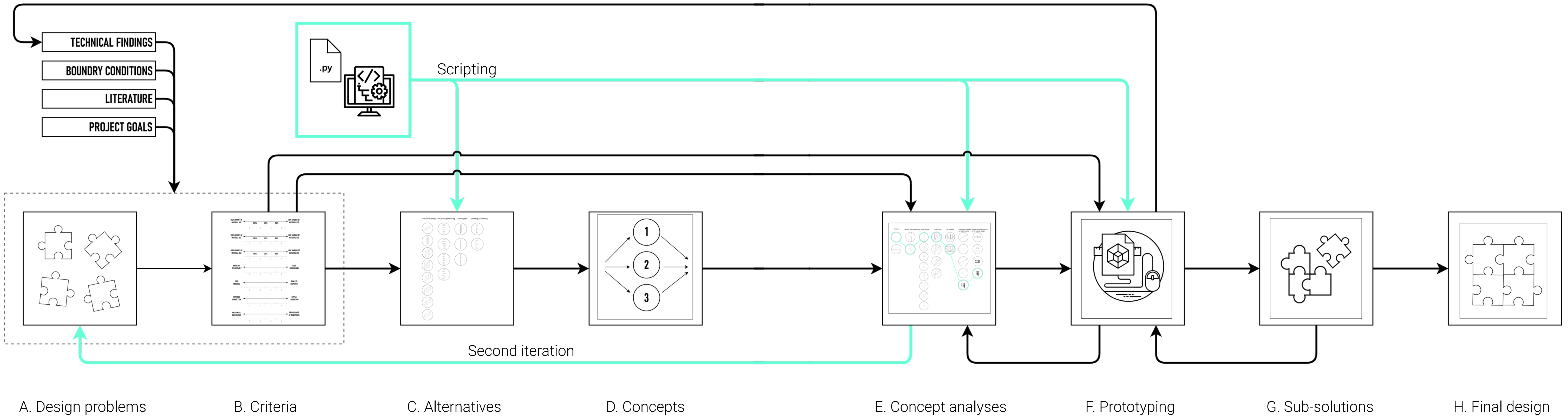


Figure 5.1 Altered design methodology

5.1 DESIGN METHODOLOGY

A design method will allow the designer to leave the mind free to produce ideas, solutions and, hunches at any time without being inhibited by practical limitations and without confusing the processes of analysis. During this process, the method will provide a system of notation that records every item of design information outside the memory, keeps design requirements and solutions utterly separate from each other, and provides a systematic means of relating solutions to requirements with the least possible compromise (Cross, 1984)

Designing can be a subjective activity where a certain creative feeling can determine, for the designer, if a design decision is good or not good. When using a design methodology, the design process and this 'feeling' can become more substantiated.

Designing something where flexibility is required due to the constantly changing availability of materials can increase the design's complexity. A design methodology can guide the designer in finding and choosing the most suitable design for the giving problem.

In this part of the research, an existing design methodology is selected and modified, to discover if it can be applied for a flexible design whereby the available materials constantly change. Writing 'quick' scripts can give the designer some useful insight and quantify the result. The output of the scripts can be compared with an existing standard system.

To see the effectiveness of the altered methodology, the methodology is applied with scripting and without scripting. This way, a comparison can be made about the efficacy of making the different parametric scripts.

The design methodology developed by Jeroen van Veen & Nick van der Knaap is used and modified to see if it can be applied to the problem of this thesis. Their design methodology is a result from a graduation thesis at the TU Delft. Van Veen and van der Knaap used literature from Eekels (Productontwerpen, structuur en methoden, 2003), Rutten & Zeiler (Geintergreerd ontwerpen van gebouw en installaties, 2005) and Gijsbers (Aanpasbaarheid van de draagstructuur, 2011) to create their methodology.

5.2 USED METHODOLOGY

In this thesis, the design method developed by Jeroen van Veen & Nick van der Knaap is used. However, the scripting aspect is implemented within the methodology to see if it can help the designer choose the most suitable option.
The methodology of van der Knaap & van Veen is divided into eight different stages:

A. Design Problems
B. Criteria
C. Alternative selection
D. Concept comparison

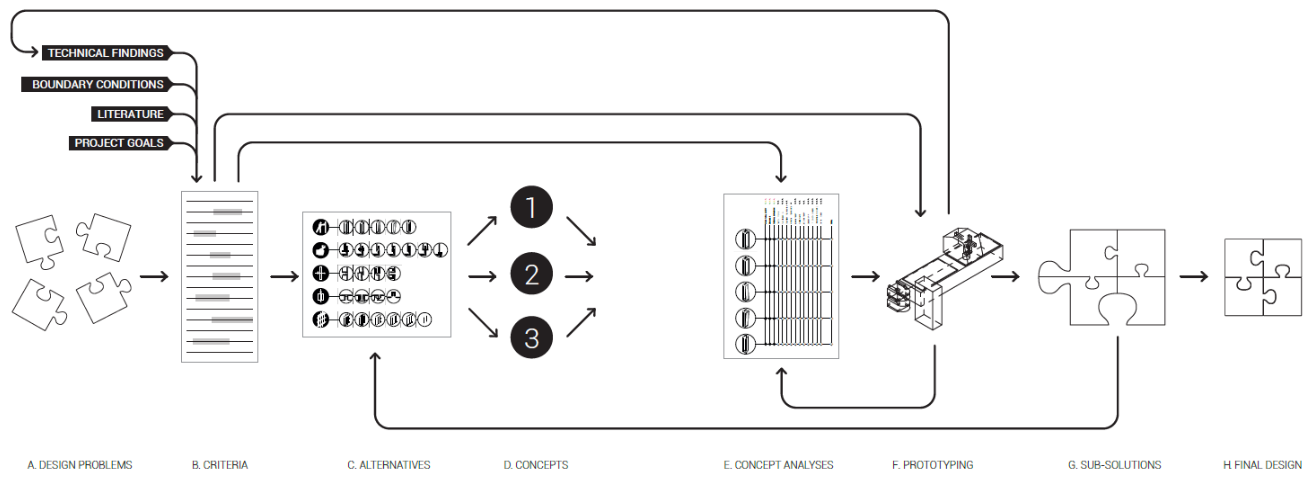


Figure 5.1 Design methodology by van Veen & van der Knaap (van Veen)

E. Concept proposal
F. Prototyping and testing
G. Sub-solutions
H. Final design

This methodological approach provides designers a framework to develop their products based on a specific set of criteria. The method can be used to generate design options, draft recommendations, and justify the final design on a quantitative basis. (van der Knaap & van Veen, 2016)

5.2.1 DESIGN PROBLEMS

The methodology starts by identifying all design problems that come with the specific design task. In this case, it is about designing an exterior wall element that can work with the varying dimensions of waste wood.

5.2.2 CRITERIA

The selected criteria are a summary of the different design objectives and the literature study. The criteria are the backbone for this methodology. Therefore, determining the criteria is an important task; they represent the essence of the design. The criteria should represent what the user desires and what the designer sees as essential.

For this methodology, The Pugh concept selection method (Pugh, 1981) is chosen to act as a guide in finding the most suitable design. This decision-matrix method is invented by Stuart Pugh, hence the name Pugh concept selection method. The decision-matrix works as follows; for each design problem different solutions are tested against a beforehand set up of weighted criteria. These criteria are specified to define the essential and the fundamentals of the design. The Pugh concept selection method works with weighted criteria. With the weighted criteria, each design solution's final scores will generate an outcome based on the importance of the criteria. To determine the weight of each criteria, the criteria are compared with each other in a matrix. If the selected criterion is more important than the compared criterion, the selected criterion will receive a 1. Vice versa, if the selected criterion is less important than the compared criterion, the selected criterion will receive a 0. A ranked list can be made by adding the scores for each criterion. Each criterion can have a weight of 1,2 or 3. The score required for a certain weight depends on the number of criteria that are used. It can be possible that certain criteria are not applicable for a specific design problem. If this happens, these criteria will be removed from the list of criteria from that specific design problem.

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	**Total**	**Weight**
Criteria 1	■	1	0	1	1	**3**	**3x**
Criteria 2	0	■	0	0	1	**2**	**2x**
Criteria 3	1	1	■	1	1	**4**	**3x**
Criteria 4	0	1	0	■	1	**2**	**2x**
Criteria 5	0	0	0	0	■	**0**	**1x**

Figure 5.2 Method to determine the weight of each criteria (own ill)

5.2.3 ALTERNATIVE SELECTION

All individual design problems with their concept designs are reviewed according to the set criteria. This process is done for every design problem. After reviewing all the design problems with their concept designs the most favorable concept is selected. The most favorable concept can be found by scoring the concepts to the criteria and multiplying this by the applied weight. The sum of these scores gives a theoretical most suitable concept solution (van der Knaap & van Veen, 2016). (van der Knaap & van Veen, 2016). Here a difference is made with the existing methodology. With the existing methodology, the number of points the designer can give for each criterion ranges between 1 up to and including 4. These points are distributed on a subjective scale. For example: complex (1) – simple (4), difficult (1) – easy (4), long (1) – quick (4), a lot of material (1) – almost no material (4). The difference between 2 or 3 points is subjective, and there is not a quantified threshold. Therefore, the points assigned by the designer are based on a feeling. This feeling is backed up with experience and research so that the designer can substantiate it. However, it remains a feeling.

By implementing scripting in this process, an alternative from the original methodology is made. Here scripting is used to quantify the criteria's thresholds and, therefore, create an objective scale. Each design solution is feed through the script, and the outputs can be placed on the scale. For the criterion of material use, the scale can be quantified as follow: 1 (> Z%) – 2 (Y-Z%) – 3 (X-Y%) – 4 (<X%). The percentage is the output compared with a reference solution. For this criterion, every output was compared with the material use of a standard TFC element.

The script parameters can be easily adjusted and therefore show the designer how these parameters relate to each other and, their impact is on the outcome of the criterion. An additional benefit of having scripts is that the designs can be easily altered during a brainstorm and generate direct feedback.

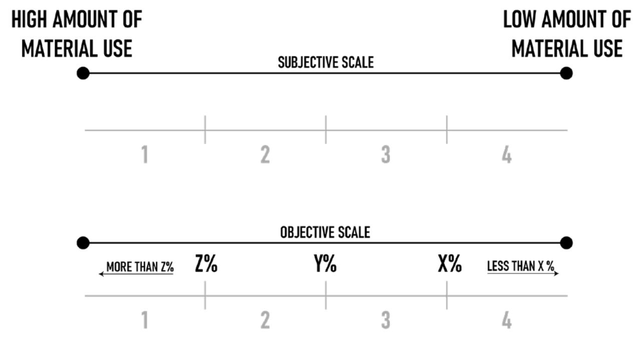
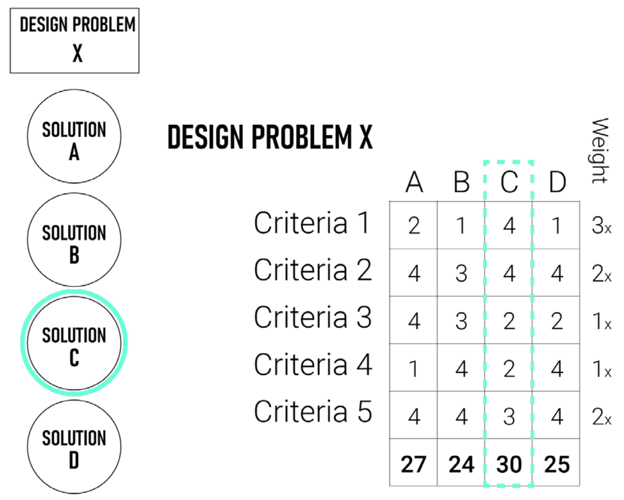


Figure 5.3 Criterion for total material use (own ill)



	A	B	C	D	Weight
Criteria 1	2	1	4	1	3x
Criteria 2	4	3	4	4	2x
Criteria 3	4	3	2	2	1x
Criteria 4	1	4	2	4	1x
Criteria 5	4	4	3	4	2x
	27	**24**	**30**	**25**	

Figure 5.4 Determing the highest scoring design solution (own ill)

5.2.4 CONCEPT COMPARISON

In the previous step, the design solutions of a specific design problem are compared with each other. Here the solutions to different design problems are compared. It is not a given fact that the overall best design combines the highest scoring solutions to each design problem. It can happen that a lower scoring solution works better with a solution from another design problem. It can also happen that some solutions have nearly identical scores. In the end, choosing the most suitable option is something that the designer does. Therefore, it is recommended to do this in a brainstorm session so that multiple people can give their input and come to a common decision.

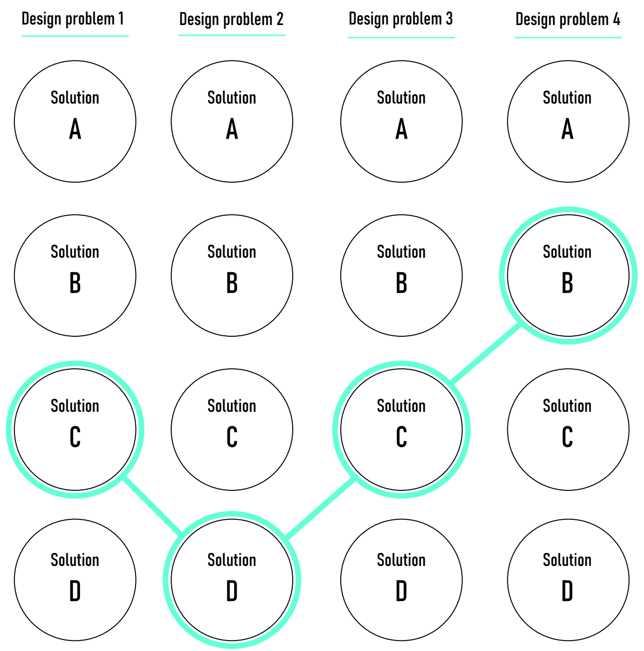


Figure 5.5 Concept comparison (own ill)

5.2.5 CONCEPT PROPOSAL

After elaborating and prototyping multiple concepts created with the previously mentioned methods, different concepts can be compared. This will result in one concept proposal which seems the most suitable for the specific design task. This concept proposal will be made parametric for this thesis.

5.2.6 PROTOTYPING AND TESTING

This is a chapter where the altered methodology deviates from the existing methodology. Here the prototype takes shape in the form of a digital playground. The result of an adjustment can be directly visible due to the data output generated by the model.

5.2.7 SUB SOLUTIONS

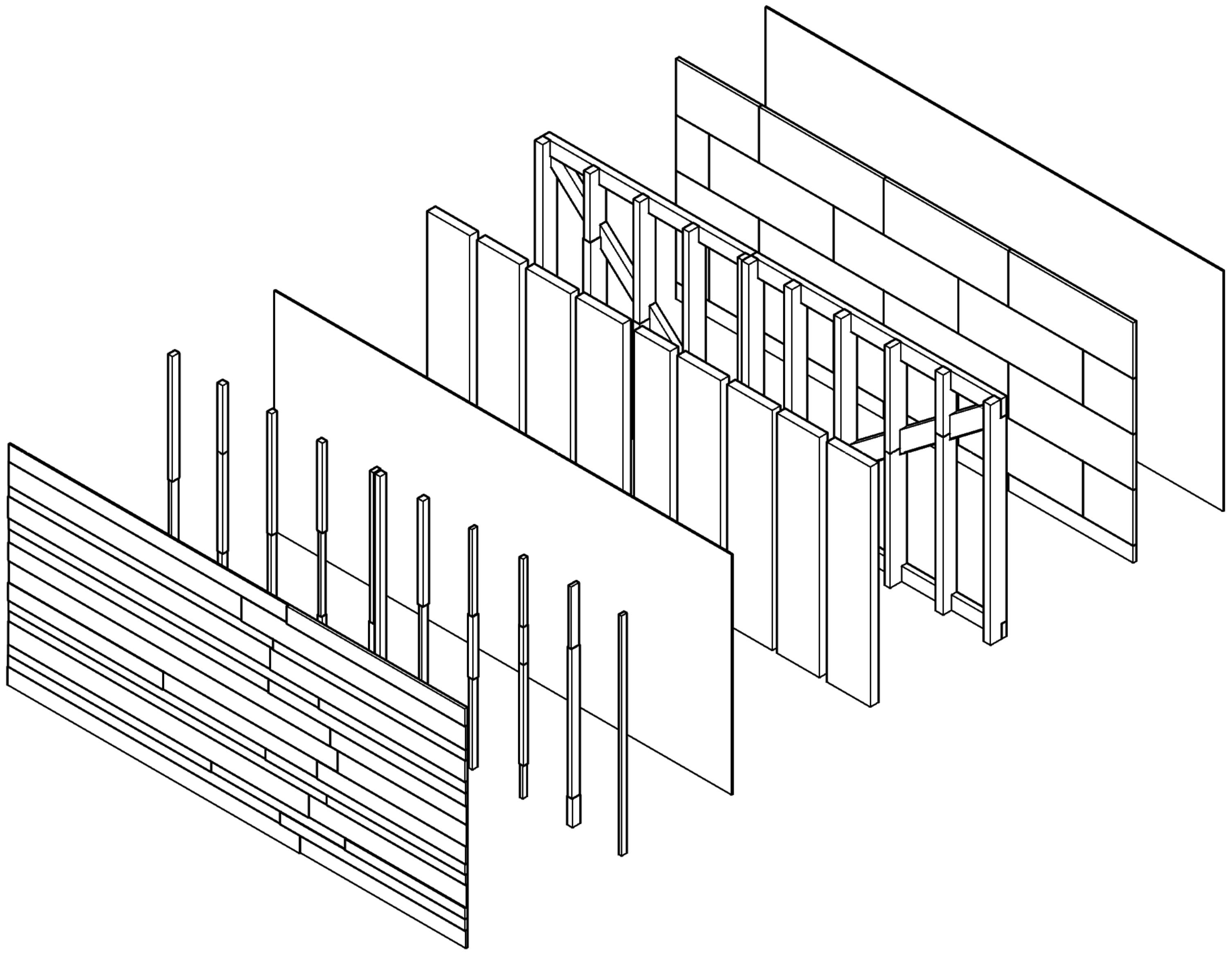
During the build of the prototype, new design problems can emerge. The parametric model gives a better 3d view of the design and can provide a direct output based on the database. Little adjustments can be made to see if the design can be optimized.

5.2.8 FINAL DESIGN

After all the adjustments, a final design can be shown that is specific to this problem. However, it can also result in a set of compromises the user can choose in.

6 CONCEPT DEVELOPMENT

In this chapter the altered design methodology from the previous chapter is applied. In the end of this chapter a reflection is written about the effectiveness of the alterations.



6.1 CRITERIA

The criteria are a direct extension of the objectives in chapter two and the literature study. The objectives for the exterior wall element are:

- The design should be constructed as much as possible with waste wood.
- The design should minimize any additional material waste that can occur when building this exterior wall element.
- The design must be compatible with the circular economy. This means that the pieces of wood need to be reused as much as possible after the exterior wall element end of life.

The study about existing TFC construction methods, which can be red in the appendix, concluded that the middle and exterior part of the exterior wall element could be constructed with waste wood. The selected criteria need to make sure this can indeed happen. Based on the objectives and the literature, the criteria can be summarized into three categories, a criterion can overlap with multiple categories:

1. Criteria that allow the exterior wall element to be constructed with waste wood and thereby minimizing additional waste.
2. Criteria that allow the exterior wall element to be reusable as much as possible after its end of life and have an environmentally positive impact.
3. Criteria that allow for a realistic and straightforward construction process.

The criteria that can determine which design solution minimizes the additional waste are:

- Expected material loss
- Full length usage
- Flexibility of pieces

The criteria were derived by implementing scripting to some of the design solutions. During this process, it became clear what types of design decisions resulted in less waste. By considering the wood that cannot be reused again after the exterior wall element end of life, the expected material loss criterion also overlaps with category two.

Within this research, the concept is all about reusing waste wood. Reusing waste wood is derived from a broader perspective to create an environmentally friendly solution instead of the alternative where waste wood suitable for reuse as a building material gets downcycled or incinerated. Therefore, the selected criteria should have an environmentally positive impact besides utilizing building with waste wood. The criteria that can determine which design solution have an environmentally positive impact and allow for as much as reuse after the wall elements end of life are:

- Total material use
- Disassembly
- Loading efficiency.
- Expected material loss

Having less material result in a more environmentally friendly design. Disassembly allows for the exterior wall element to be reused again after its end of life and efficient loading for less carbon emissions during transportation.

For the exterior wall element tool to have a better change to be implemented in the Dutch waste wood market and for the exterior wall element to be constructed with waste wood, the design should be simple and realistic. The assembly should be possible with a couple of people. The criteria that allow for a simple and realistic construction process are:

- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Complexity of connection

All the criteria are elaborated in more detail in the next paragraph.

6.1.1 DESIGN PRINCIPLES

The selected criteria are not only a result of the literature study but also by following the design methodology and discover by research through design what design interventions can help achieve the design objectives determined in chapter 2. Some of these criteria are derived from certain design principles that were discovered during the design methodology. The design methodology was repeated with the new criteria and design principles integrated. The design principles that were derived are as follows:

Modifications on the pieces of wood within the chosen design should only be on the outside. This allows for the wall element to maximize the reusable pieces of wood after its first use as a wall element. The modified parts can be removed, and the remaining piece of wood can be reused again in a new exterior wall element.

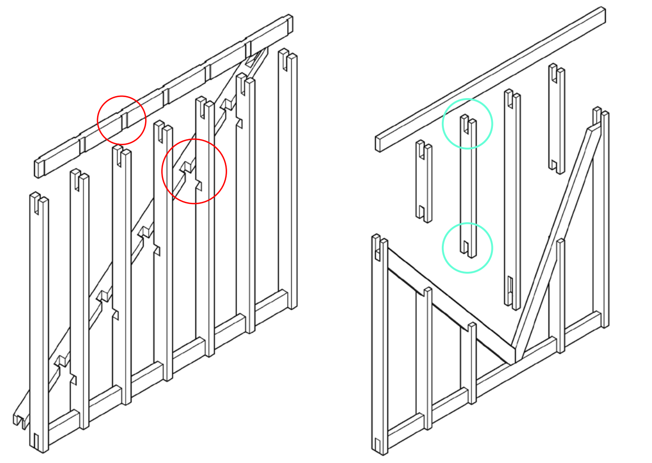


Figure 6.1 Design principle 1, modifications should be on the outside of the piece of wood (own ill)

Symmetry of the design should be prevented. A symmetric design can result in additional waste because the same dimensions of the wood have to be selected twice. The chance increases that the wood is not available in that exact dimensions twice, and therefore, a larger piece of wood has to be cut to fulfill the required dimensions.

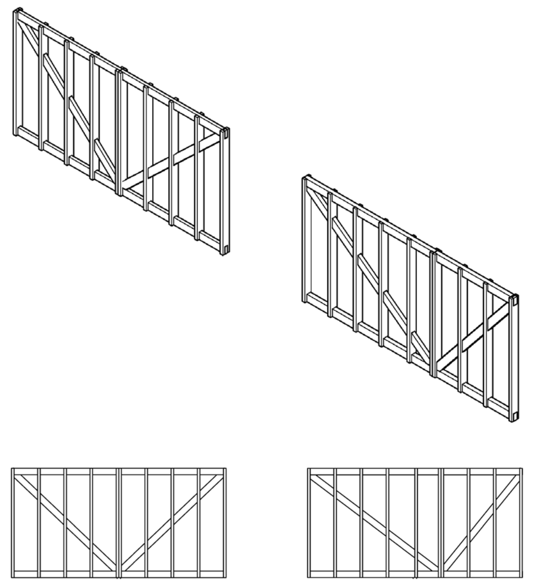


Figure 6.2 Design principle 2, symmetric design (left) should be prevented (own ill)

A broad range of dimensions should be used in the design. This way, the chance increases in selecting the pieces of waste wood that fits closest to the required dimensions. Having only small pieces required for the design can result in splitting larger pieces of waste wood into smaller pieces. This will result in a decrease in value. Having a design with only large pieces of wood will result in not having a function for the small pieces of wood.

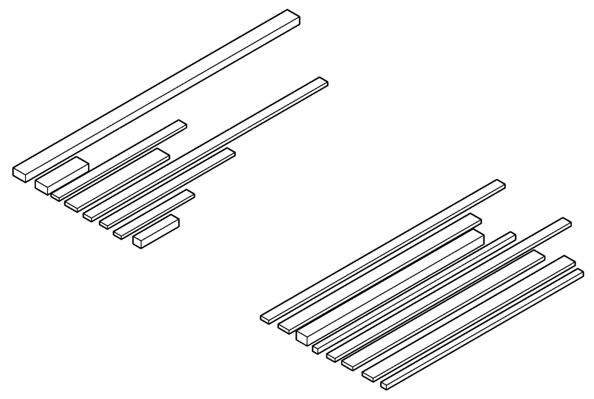


Figure 6.3 Design principle 3, broad range of required dimensions is preferred (left option) (own ill)

QUANTIFY THE CRITERIA

In this thesis making scripts that can give a quantified output allows for rating the criteria on certain thresholds. The use of a parametric script can provide a better understanding when all the possible solutions are compared with each other. The data from the script will be used in determining the score. The criteria that are suitable for scripting are:

- Expected material loss
- Total material use
- Machine time
- Flexibility of pieces

The following chapter will explain how these criteria can be quantified. The exact scale is not given because that can change with each given design problem.

6.1.2 CRITERIA EXPLAINED

Expected material loss:

Expected material loss is all the waste that is created during construction of the wall element. For example, if the stud's required length is 2 meters, but there is only a 2.15 meter piece of waste wood available, 0.15 meter will be considered as waste. If this specific wood can find a function somewhere else in the design, it will not be defined as waste. Wood that cannot be reused after the end of life of the exterior wall element is also considered as waste.

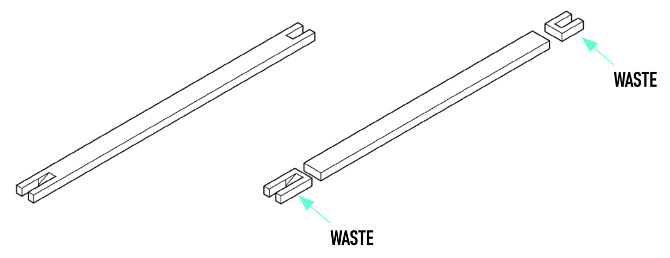
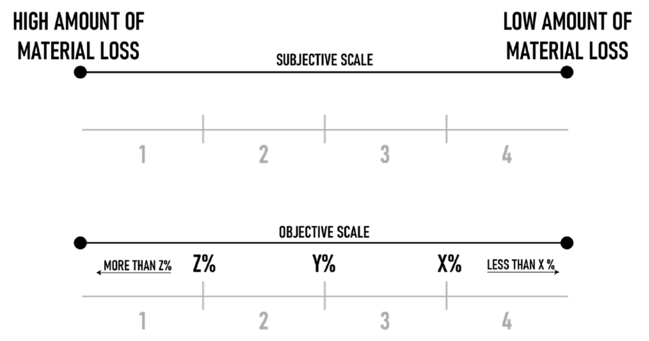


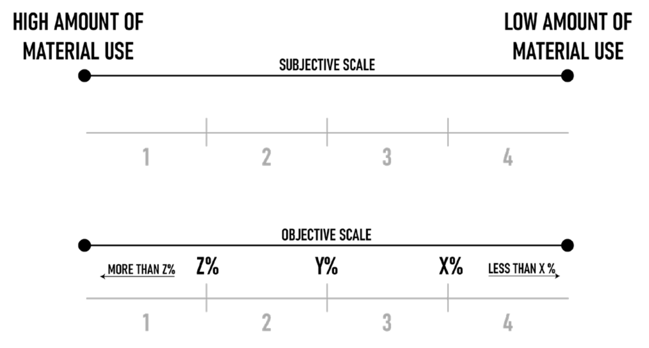
Figure 6.4 End of life material loss (own ill)



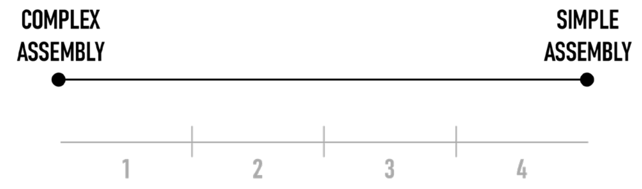
Total material use:
It is important to not only look at the total material loss but also the total material use. Using less material is overall a better design in an environmental point of view.

However, a design can minimize the additional waste but still requires more meters in total. Because end of life is considered in the expected material loss, it can be justified to have a design that has a higher total material use but less waste in the end.

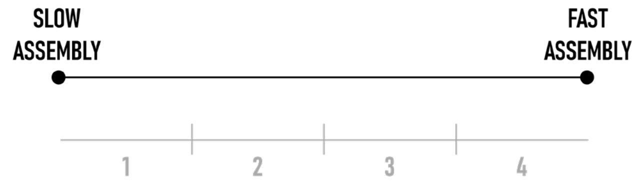
To quantify the scale, the design is compared with a standard TFC wall element.



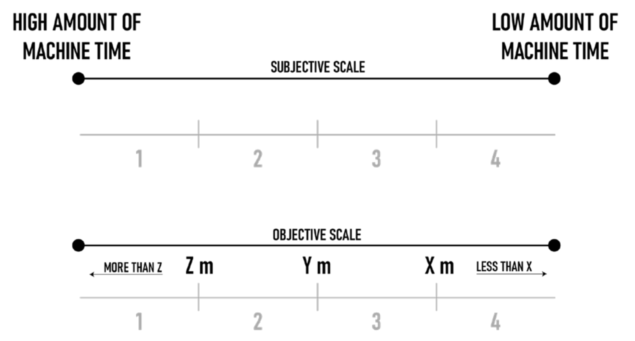
Assembly complexity:
To end up with a realistic design, assembly complexity is considered. The assumption is made that humans, with the help of tools, are assembling the construction. In the future, this process can be replaced by machines, but that is outside the scope of this research.



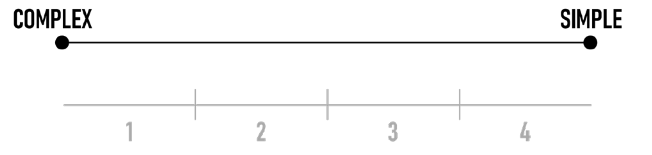
Assembly time:
It is safe to say that assembly complexity and assembly time correlate. Increased complexity will result in more assembly time. However, a simple design can still take more time to assemble than a complex design due to the number of actions / pieces that need to be assembled. Therefore, it is essential to separate the two criteria.



Machine time:
The pieces of waste wood need to be modified so they get the right dimensions and allow for certain connections. The number of cuts and the complexity of the cut will result in a machine time. To objectify the scale a calculation about the estimated machine time can be made by total meters of cuts that need to be performed.



Machine complexity:
Machine complexity depends on the type of cut. For a groove in a piece of wood a simple saw blade can be efficient, while a complex Japanese inspired joint requires a multiple axis CNC milling machine.



Flexibility of pieces:
One of the biggest challenges of building with waste wood is the diversity in the material's dimensions. A design that allows for multiple combinations between the pieces of wood is considered flexible. A repetitive design will require the pieces to have the same dimensions. Every time this piece is selected, the chance to find another piece with that exact dimensions become smaller. To obtain the right dimensions a larger piece needs to be selected and shortened. This will result in additional waste. A design with repetitive dimensions will score low on this criterion. To objectify this criterion, a percentage can be calculated. The number of different lengths divided by the total number of pieces * 100 gives a percentage. This percentage can be used to compare the different designs solutions.

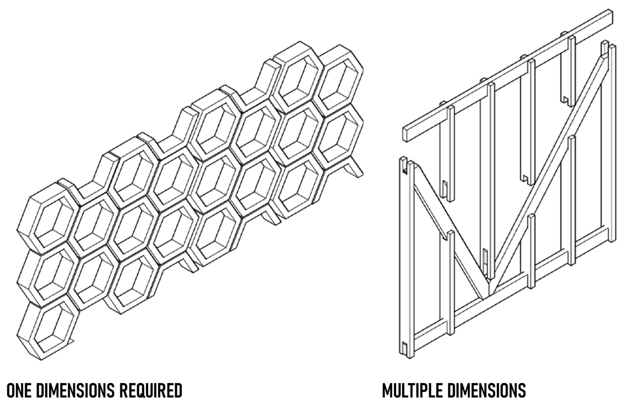
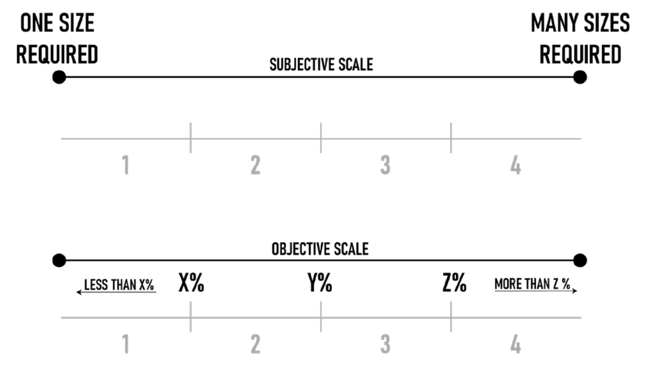
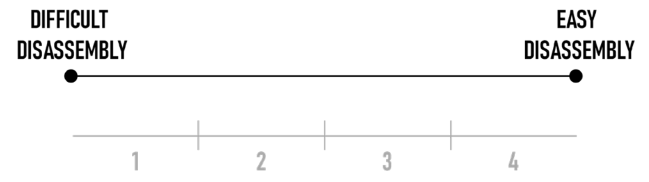


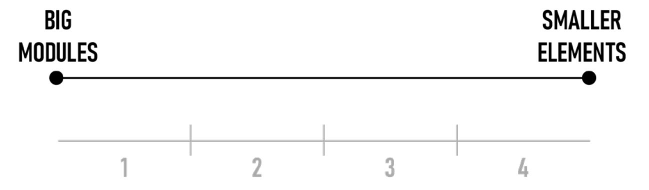
Figure 6.5 Flexibility of pieces (own ill)



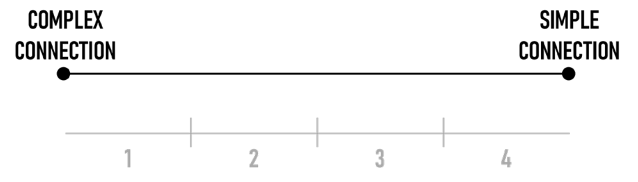
Disassembly:
Disassembly allows for the pieces of waste wood to be reused on a new wall element. Some connections allow for easier disassembly then others.



Loading efficiency:
Can the wall be transported in segments and stacked inside a truck, or does it need to be transported in one piece. A higher loading efficiency will result in less CO2 emission and will reduce the transportation cost.



Complexity of connection:
Complex connections can have a negative impact on the assembly and disassembly. A simple connection is preferred over a complex one.



Full length usage
When a design only requires small pieces, eventually, a long piece needs to be cut in multiple pieces to fulfill this demand. This will result in a loss of value of the piece of wood and after the disassembly of the wall element the piece of wood cannot be reused in its original length. It is preferred to use the pieces of wood in their original length. A design scores higher if the length of dimensions required vary.

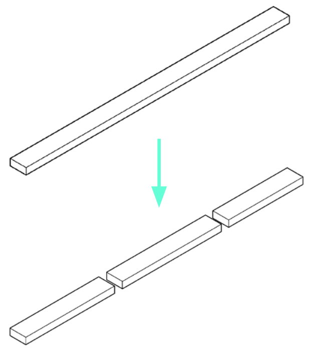
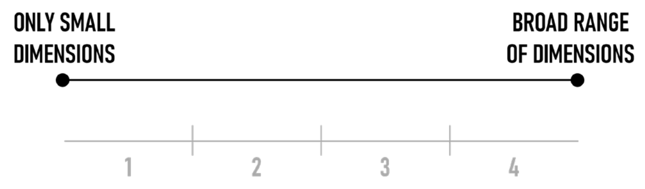


Figure 6.6 Splitting wood in multiple pieces result in a loss of value (own ill)



6.2 CRITERIA MATRIX

The following table gives an overview of every criteria compared with each other. The decision which criteria is more important is a subjective decision made by the designer(s).

The eleven criteria that where selected can have a weight 1, 2 or 3. Criteria with a total score 0-3 will have a weight of 1. Criteria with a score 4-6 will have a weight of 2. Criteria with a score 7-10 will have a weight of 3.

Product methodologies give to possibility to make a substantiated decision and can support the designer in the decision-making process. The first step is to have a clear overview of all the design problems that exist when creating an exterior wall element from waste wood.

The following list identifies the problems of designing an exterior wall with waste wood. Some problems do not have a direct relation to designing with waste wood, such as insulation or soundproofing. For these problems, existing solutions are chosen that already proven to be a sustainable solution.

6.3 ASSERTING CRITERIA TO DESIGN PROBLEMS

Structural infill:
This aspect determines the structural infill of the wall element and will carry the load of its own weight, the substructure, and the cladding. Different systems are described and researched. The following criteria are applied:

- Expected material loss
- Total material use
- Assembly complexity
- Assembly time
- Machine time
- Flexibility of pieces
- Loading efficiency
- Complexity of connection
- Full length usage

Structural positioning:
How is the structural infill (studs) positioned relative to each other, working with waste wood gives the challenge of working with varying dimensions of each stud.

Cladding:
The cladding is responsible for protecting the inner structure against the elements. The choice can be made for an open facade with an additional foil that will act as water barrier or a closed facade whereby the cladding is the water barrier.

- Total material use
- Assembly complexity
- Assembly time
- Disassembly

Cladding position:
Different options are researched and described regarding the position of the cladding relative to the substructure.

- Expected material loss
- Total material use
- Flexibility of pieces
- Full length usage

	Expected material loss	Total material use	Assembly complexity	Assembly time	Machine time	Machine Complexity	Flexibility of pieces	Disassembly	Loading efficiency	Complexity of connection	Full length usage	**Total**	**Weight**
Expected material loss	■	1	1	1	1	1	0	1	1	1	1	**9**	**3x**
Total material use	0	■	1	1	1	1	0	1	1	1	1	**8**	**3x**
Assembly complexity	0	0	■	1	1	1	0	0	1	0	0	**4**	**2x**
Assembly time	0	0	0	■	1	0	0	0	1	0	0	**2**	**1x**
Machine time	0	0	0	0	■	1	0	0	1	0	0	**2**	**1x**
Machine Complexity	0	0	0	1	0	■	0	0	1	0	0	**2**	**1x**
Flexibility of pieces	1	1	1	1	1	1	■	1	1	1	1	**10**	**3x**
Disassembly	0	0	1	1	1	1	0	■	1	1	0	**6**	**2x**
Loading efficiency	0	0	0	0	0	0	0	0	■	0	0	**0**	**1x**
Complexity of connection	0	0	1	1	1	1	0	0	1	■	0	**5**	**2x**
Full length usage	0	0	1	1	1	1	0	1	1	1	■	**7**	**3x**

Figure 6.7 Decision matrix to determine weight of each criteria (own ill)

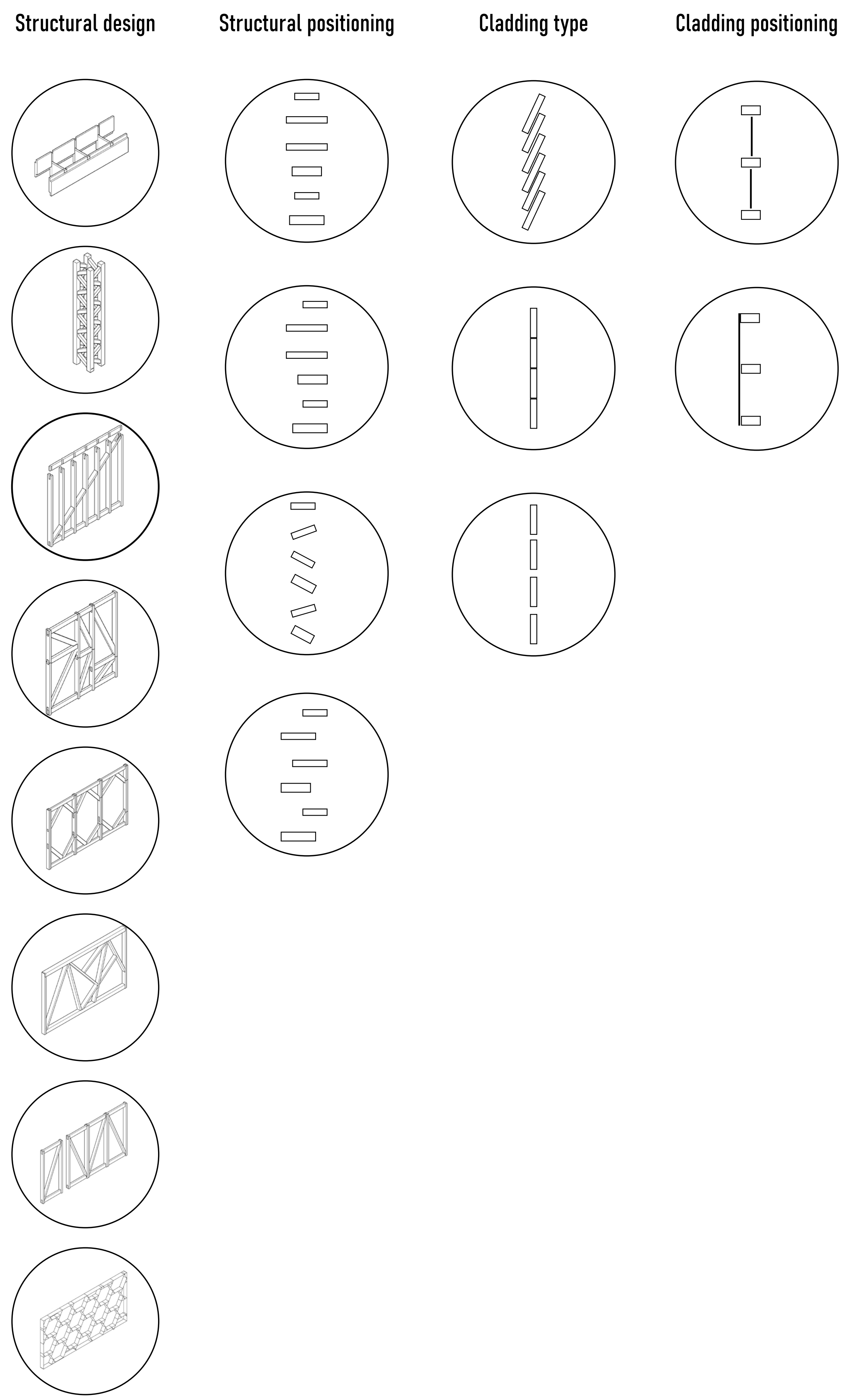


Figure 6.8 First iteration of design problems with each possible solution (own ill)

6.4 DESIGN PROBLEMS

6.4.1 STRUCTURAL INFILL:

The solutions to this design problem determine the structural infill of the wall element. The structural infill will carry the load of its own weight, the substructure, and the cladding. Existing designs and designs derived from traditional building methods are compared and researched. The criteria total material use has high importance due to its weight. To generate a better estimation about the total material use, the designs are made parametric in a script. This will give a fascinating insight into how a parametric script can support the design methodology and test if the score for each design based on logical thinking is the same as the score based on the script's output.

To keep the calculations and estimations simple, only the total length required is compared. Here the width and depth are neglected.

The outputs of the scripts are compared with the length required of a basic timber framing wall. The wall has a width of 4.8 meters and a height of 3 meters. The center-to-center distance is 600 millimeters. With these calculations, the number of studs can be a number with a decimal. This happens when the width of the wall is not a multiply of 600mm. Usually, when this happens, an additional stud is placed. However, for comparing the different outputs this principle is ignored. Otherwise, some designs would have an extra negative result in comparison with the reference wall element.

Score overview with Total material use based on the outcome of the script with the following scale. 1 (> 130%) – 2 (120-130%) – 3 (110%-120%) - 4 (< 110%). The percentages are based on a comparison of the total meters required for a standard TFC element.

Score overview with Total material use based on estimation. The score generated by the parametric model did vary with the estimations by the designer, especially on the more complicated designs.

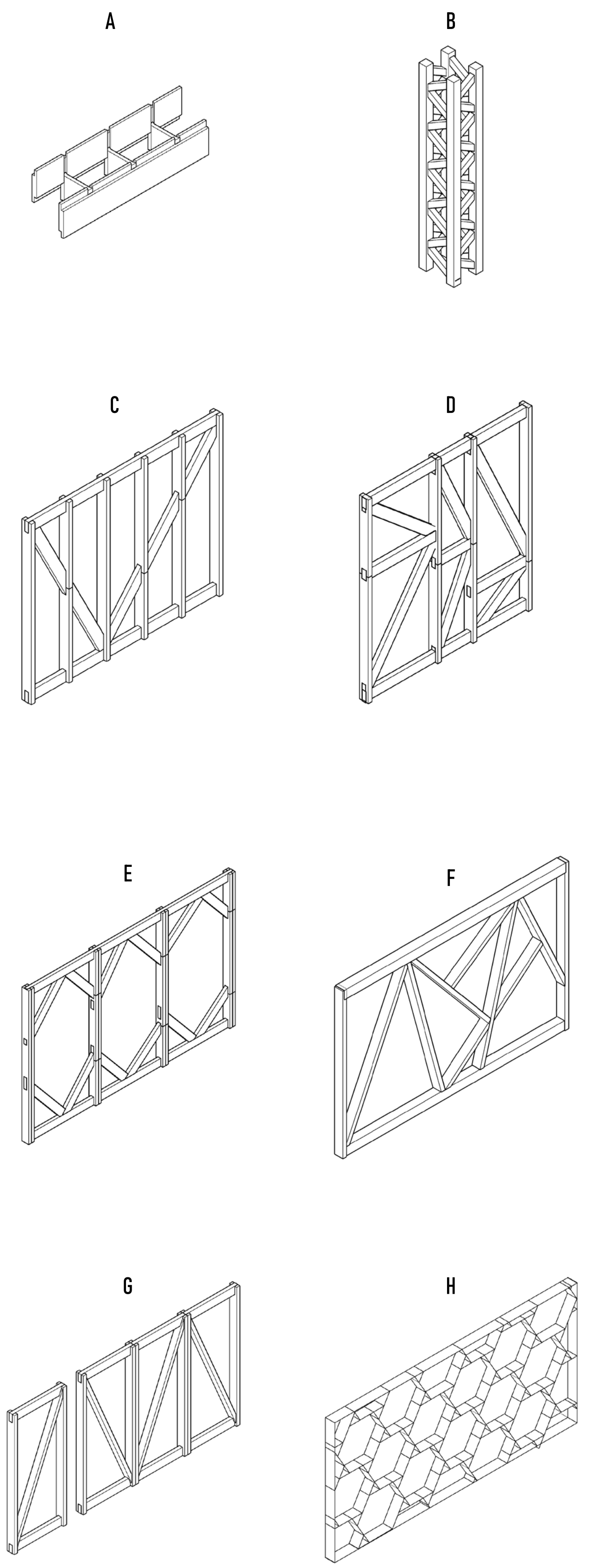


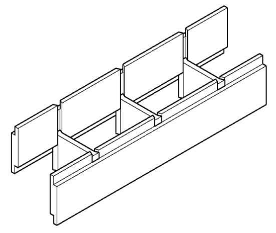
Figure 6.9 Structural design problem (own ill)

STRUCTURAL INFILL

	A	B	C	D	E	F	G	H	Weight
Expected material loss	3	4	3	3	3	4	3	1	3x
Total material use	1	3	4	1	4	4	4	4	3x
Assembly complexity	4	4	3	2	3	1	4	1	2x
Assembly time	1	1	3	2	3	1	3	1	1x
Machine time	1	1	4	3	4	3	4	1	1x
Flexibility of pieces	4	4	4	4	4	4	4	1	3x
Loading efficiency	4	4	1	1	1	1	3	1	1x
Complexity of connection	4	4	3	1	2	1	4	1	2x
Full length usage	1	4	4	4	4	4	4	1	3x
	49	**67**	**65**	**48**	**63**	**57**	**68**	**19**	

Figure 6.10 Determing the scores for the structural solutions (own ill)

Brikawood (A):



Brikawood is an existing system where the construction consists of a brick-like design. The benefit of Brikawood, besides being an existing system and therefore already proven itself, is that the use of nails, screws, or adhesives are not required (Brikawood, 2012).

In the parametric script, the brick's height and the distance between the connection pieces can be adjusted. Adjusting these parameters will result in more understanding of the model and how the total length can be minimized.

The brick has a standard length of 500 mm, with two connector pieces joining the brick together. The height is approximately 300mm. Using this as an input for the script, while creating a wall of 4.8 meters wide and 3 meters high, will require roughly 154 meters of wood. This high result comes from the requirement of two pieces of wood and two connector pieces for every 500 mm.

Even increasing the center-to-center distance between the connection pieces will result in a higher total meters of wood required than the standard TFC.

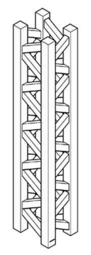
If the Brikawood would also act as an exterior cladding, the high amount would be justified and maybe even be more beneficial than other designs. For the design to function as exterior cladding and structural infill, it needs to be completely watertight. However, working with waste wood brings some downsides, such as the quality of the wood. Waste wood has been used, it can contain small holes from nails or screws, and therefore not all waste wood can guarantee to be completely watertight. Eliminating all this wood for reuse would shrink the available wood drastically and is therefore not preferred.

For the brick-like construction, every piece of wood used needs to be modified, resulting in a high machine time.

```
def brickawood(height_wall, width_wall, ctc_connection, height_brick):
    horizontal_length = height_wall / height_brick * width_wall * 2
    vertical_length = width_wall / ctc_connection * height_wall
    total_length = horizontal_length + vertical_length
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')


brickawood(3000, 4800, 250, 300)

```

```
total length = 153.6 meter
```

Truss (B):

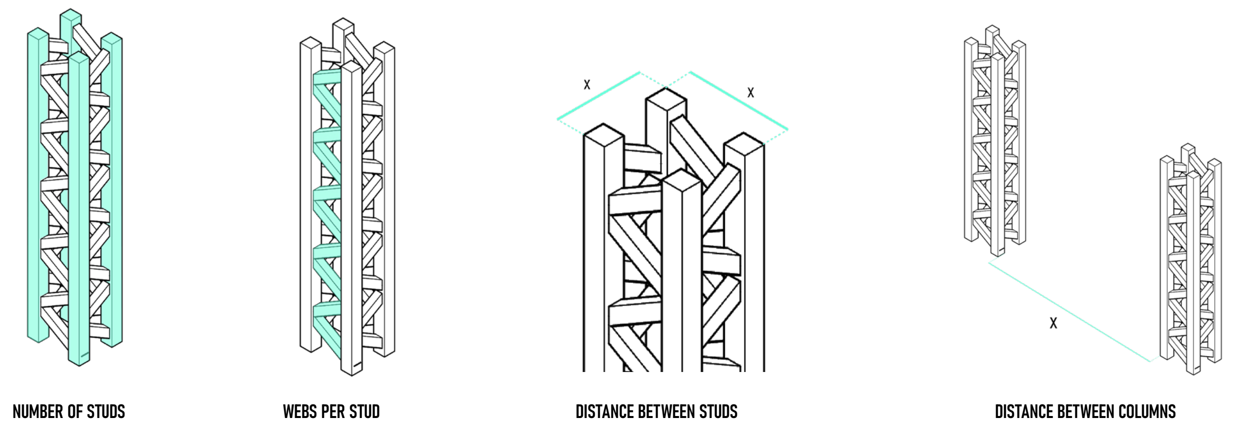


Design B is a wooden truss that acts as a column. The column consists of multiple studs with slanted web pieces between them. The following elements are parametric within the script:

- The number of web pieces for every meter height between two studs.
- The distance between the studs.
- The number of studs per truss
- The distance between the columns.

For this input, a truss consists of 4 studs, with 3 webs per meter height within 2 studs, a distance between the studs of 200mm and a distance of 4 meters between the trusses will result in 68 meters of wood required.

Even decreasing the number of studs, the distance between the studs and the number of webs will still result in a higher required length than the TFC.

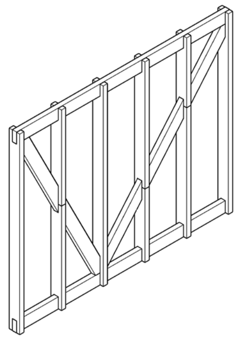


```
def truss(height_wall, width_wall, web_connection_per_meter, distance_between_trusses,
          studs_per_truss, distance_between_studs):

    number_trusses = width_wall / distance_between_trusses + 1
    vertical_length = number_trusses * studs_per_truss * height_wall
    height_single_web = 1000 / web_connection_per_meter
    length_single_web = (height_single_web**2 + distance_between_studs**2)**0.5

    total_length_webs = height_wall / height_single_web * length_single_web * studs_per_truss * number_trusses

    total_length = total_length_webs + vertical_length
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')

truss(3000, 4800, 3, 3000, 4, 200)
```

```
total length = 67.59 meter
```

Figure 6.11 Script with parameters for the truss design solution (own ill)

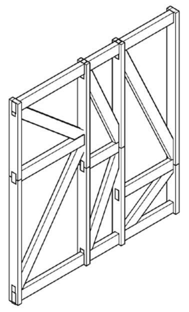
Traditional framing with diagonal bracing (C):



Design C is closest to the current TFC method with diagonal bracing. Therefore, the meters of wood required for this type of construction is the same as the standard reference.

The modifications in the studs allow this construction design to work with waste wood. The studs can consist of two pieces that are joint at the location where it intersects with the bracing. This results in lower material loss due to the changing required length of the pieces.

Segments (D):



Design D consists of different rectangles with diagonal bracing. The method of two braces between two studs is called dog leg bracing. However, this bracing type will result in a higher amount of wood required because of the increased bracing amount.

The increased number of bracing results in an increased total material use. Also, with this design, it can occur that five pieces of wood intersect, and this will require a more complicated connection.

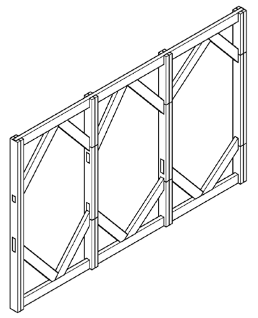
```python
def hsb(height_wall, width_wall, ctc):

    number_studs = width_wall / ctc + 1
    horizontal_length = 2*width_wall

    if width_wall <= 3000:
        bracing = (height_wall**2+width_wall**2)**0.5

    else:
        bracing = ((height_wall**2+(width_wall/2)**2)**0.5)+((height_wall**2+(width_wall/2)**2)**0.5)

    total_length = number_studs * height_wall + horizontal_length + bracing
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')

hsb(3000, 4800, 600)


```

```
total length = 44.28 meter
```

```python
def segments(width_wall, height_wall, ctc):

    number_studs = width_wall / ctc + 1
    vertical_length = number_studs * height_wall
    horizontal_length = 2*width_wall
    bracing  = (((0.5*height_wall)**2 + ctc**2)**0.5) * (width_wall / ctc) * 2

    total_length  = vertical_length + horizontal_length + bracing
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')

segments(4800, 3000, 600)


```

```
total length = 62.45 meter
```

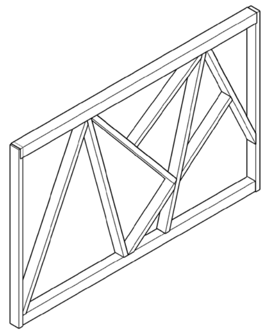
Post and beam (E)



Design E is a derivative from a traditional post and beam construction method. This design combines two studs that allow for an increased center-to-center distance compared to the TFC. Knee bracing is added for stability and stiffness. The length of the knee bracing does not have to be consistent and can vary based on the wood available. The location where the knee bracing connects with the stud can also be the point where the stud is divided into multiple pieces. This will also allow for more flexibility in the required length of wood.

For the script, a fixed height is applied where the bracing intersects with the studs. Being able to change this parameter can give some insight into the total meters of wood that are required. An input of 0.5 means that the knee braces intersect in the middle of the studs.

Diagonal design (F)



Design F is designed to use the full length of the available wood, resulting in minimizing the waste. Putting this principle in a parametric script would consume too much time for this research. From a material point of view, vertical load transfer is always preferred over diagonal load transfer because diagonal load transfer will require more material to carry the same amount of load. However, this design does not require any additional bracing. The diagonal pieces already create the needed stiffness. With the TFC reference, 13% of the total length consists of bracing. The lack of bracing required can compensate for the additional material necessary for this design to carry the load.

A downside of working with only slanted pieces will result in a more complex connection when the two pieces need to be joined to create a longer piece. Slanted pieces require a connection that can take compression and tension. The different angles between the slanted pieces can also increase the complexity of the connection.

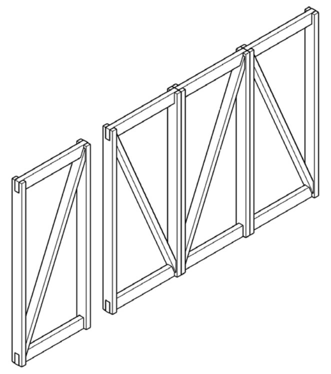
```python
def post_beam (width_wall, height_wall, ctc, relative_height_bracing):

    amount_studs = width_wall / ctc + 1
    vertical_length = amount_studs * 2 * height_wall
    horizontal_length = 2 * width_wall
    bracing = (((relative_height_bracing * height_wall)**2 + (0.5*ctc)**2)**0.5) * 2 * (width_wall/ctc)
    total_length = vertical_length + horizontal_length + bracing
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')

post_beam(4800, 3000, 1200, 0.5)

```

```
total length = 52.52 meter
```

```python
post_beam(4800, 3000, 1200, 0.25)

```

```
total length = 47.28 meter
```

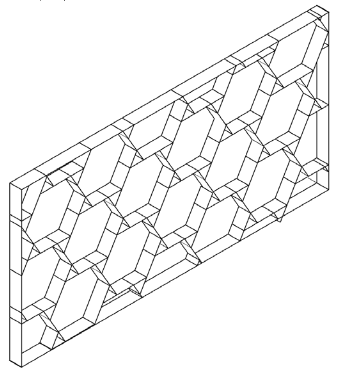
Modular design (G):



Design G consists of a box module that can be attached to each other to create the wall element's structural infill. This design, like design E, has double studs. These double studs are created when the modules are connected with each other. These double studs will allow for an increased center to center distance compared to a traditional timber frame.

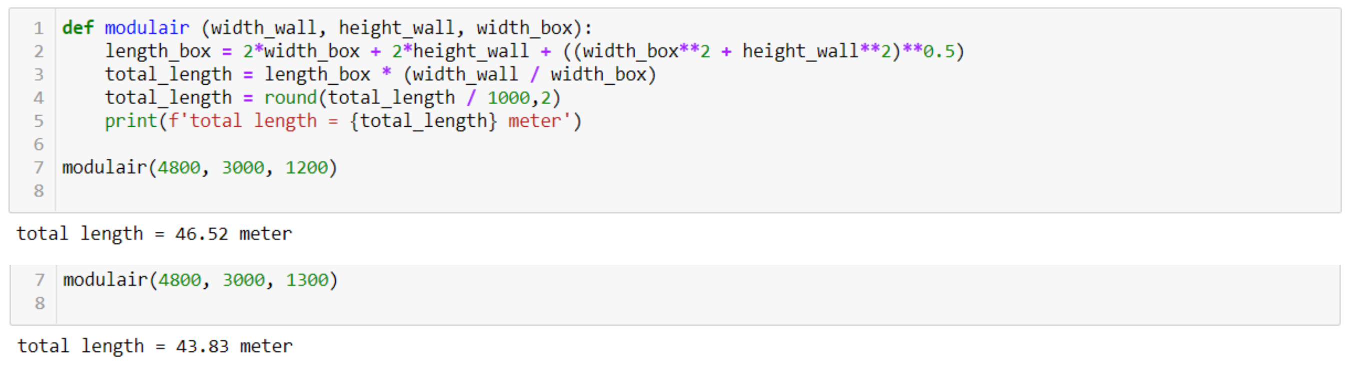
Increasing the width of the box by 100 mm results in a lower amount of meter wood required than the traditional timber frame.

Honeycomb (H):



Design H consists of a honeycomb structure. This type of structure would not allow for a simple python script. Instead of Python, grasshopper was used. The design is slightly simplified, but the number of meters required will give a good estimation. In the parametric model, the dimensions of the triangle and the hexagon can be adjusted to see the effect relative to each other on the total meters of wood required.

A hexagon where the length of a side is 650mm and a triangle where the side is 50 mm will result in 45.1 meters of wood.

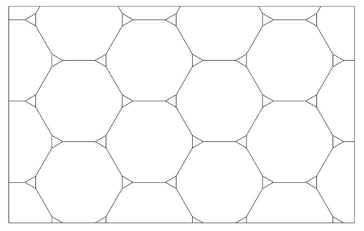


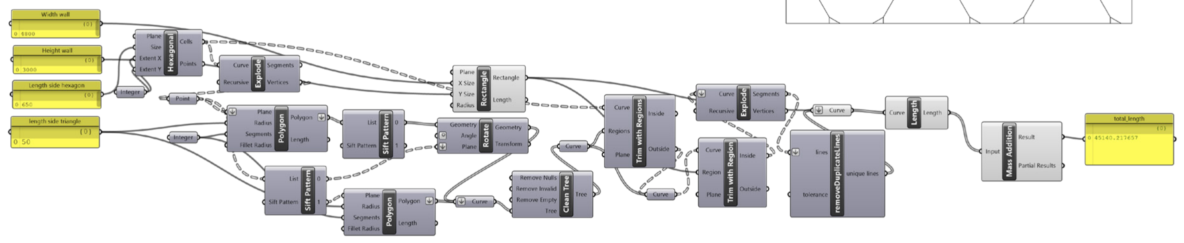
```
def modulair (width_wall, height_wall, width_box):
    length_box = 2*width_box + 2*height_wall + ((width_box**2 + height_wall**2)**0.5)
    total_length = length_box * (width_wall / width_box)
    total_length = round(total_length / 1000,2)
    print(f'total length = {total_length} meter')

modulair(4800, 3000, 1200)

```

```
total length = 46.52 meter
```

```
modulair(4800, 3000, 1300)

```

```
total length = 43.83 meter
```





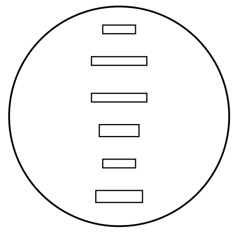
A hexagon where the side is 500mm and where the side of the triangle is 100 mm will result in 59 meters of wood.

A downside is that the pieces of wood are all the same length making it less suitable for a wall element from waste wood. Also, the load carrying benefit of having a honeycomb type of structure is mostly in a horizontal direction.

6.4.2 STRUCTURAL POSITIONING

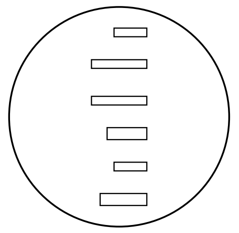
The waste wood will have different dimensions, resulting in an irregular alignment of the studs. Multiple options are described for the alignment. The decision for the alignment has no direct relation to the criteria. Therefore, the methodology cannot be used appropriately to gain a score for each option. However, the decisions for the alignment does have an impact on later stages of the design.

Centre align (A):



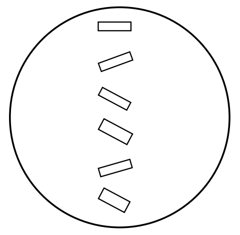
With a center alignment, the studs will be aligned through the middle. This will result in a difference between the depts of the studs towards the exterior and the interior.

Alternating (B)



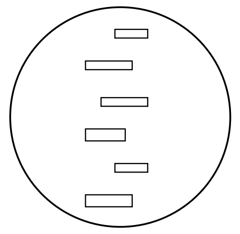
An alternating positioning results in an alignment towards the exterior and the interior. However, the distance between the studs, which can be used to attach the substructure, is doubled.

Rotating (C)



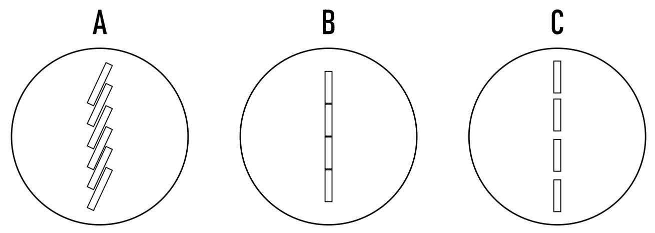
Rotating the studs can result in an interior and exterior alignment. However, the rotated surfaces increase the complexity of assembling the substructure to the studs.

Exterior or interior align (D)



Alignment towards one side has the benefit that the difference between the depts only occurs on one side. With the assumption that the wall horizontally assembled, one-sided alignment will complement this building method.

6.4.3 CLADDING

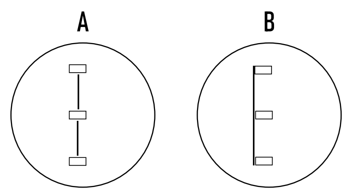


CLADDING

	A	B	C	Weight
Expected material loss	2	4	4	3x
Assembly complexity	4	4	4	2x
Assembly time	2	4	4	1x
Disassembly	2	4	4	2x
	20	**24**	**68**	

The cladding protects the structural infill against the elements. The cladding can be placed in a way that it creates a closed facade or an open facade. When an open facade is chosen, the water-barrier is done by a foil placed behind the cladding with enough room to create a ventilated cavity. With a closed facade, the cladding itself will act as a water barrier. Given that the cladding will consist of waste wood, it is more likely that the pieces of wood can be crooked, vary in dimensions, and have tiny holes from previous nails and screws. Therefore, it is a better choice to ensure the water barrier is secured and performed by a foil.

6.4.4 CLADDING POSITION



CLADDING POSITION

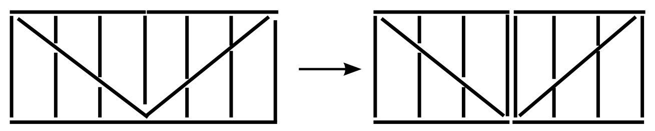
	A	B	Weight
Expected material loss	1	4	3x
Total material use	4	3	3x
Flexibility of pieces	1	4	3x
Full length usage	1	4	3x
	21	33	

The cladding can be placed in front of the substructure or between the substructure. Placing the cladding between the substructure will generate more waste and unnecessary cuts, because this distance determines the required length of the cladding. Placing the cladding in front of the substructure will allow for using the full length of the wood.

6.5 CONCEPT COMPARISON:

Circled in a solid line is the solution with the highest score. Solutions that have a similar score have a dashed circle.

The decision for the cladding type and positioning is evident. However, there are multiple solutions for the structural infill with a similar score, with the box module scoring slightly higher. Choosing an open facade will require a ventilated cavity and an additional substructure where the exterior cladding can be attached on. For this requirement, an increased center to center distance results in less space for the substructure and cladding to be attached onto the structural infill. If we take this into account, the truss would not be suitable due to the considerable distance between the studs. A combination between design C & G can be made, where each element only consists of one diagonal but has the center-to-center distance from design C.



The wall will be assembled horizontally, having an alignment to one side allows the studs to lay flat during assembly. Additional substructure is required for creating a cavity. This substructure can be used to flatten the difference in depth from the structural infill.

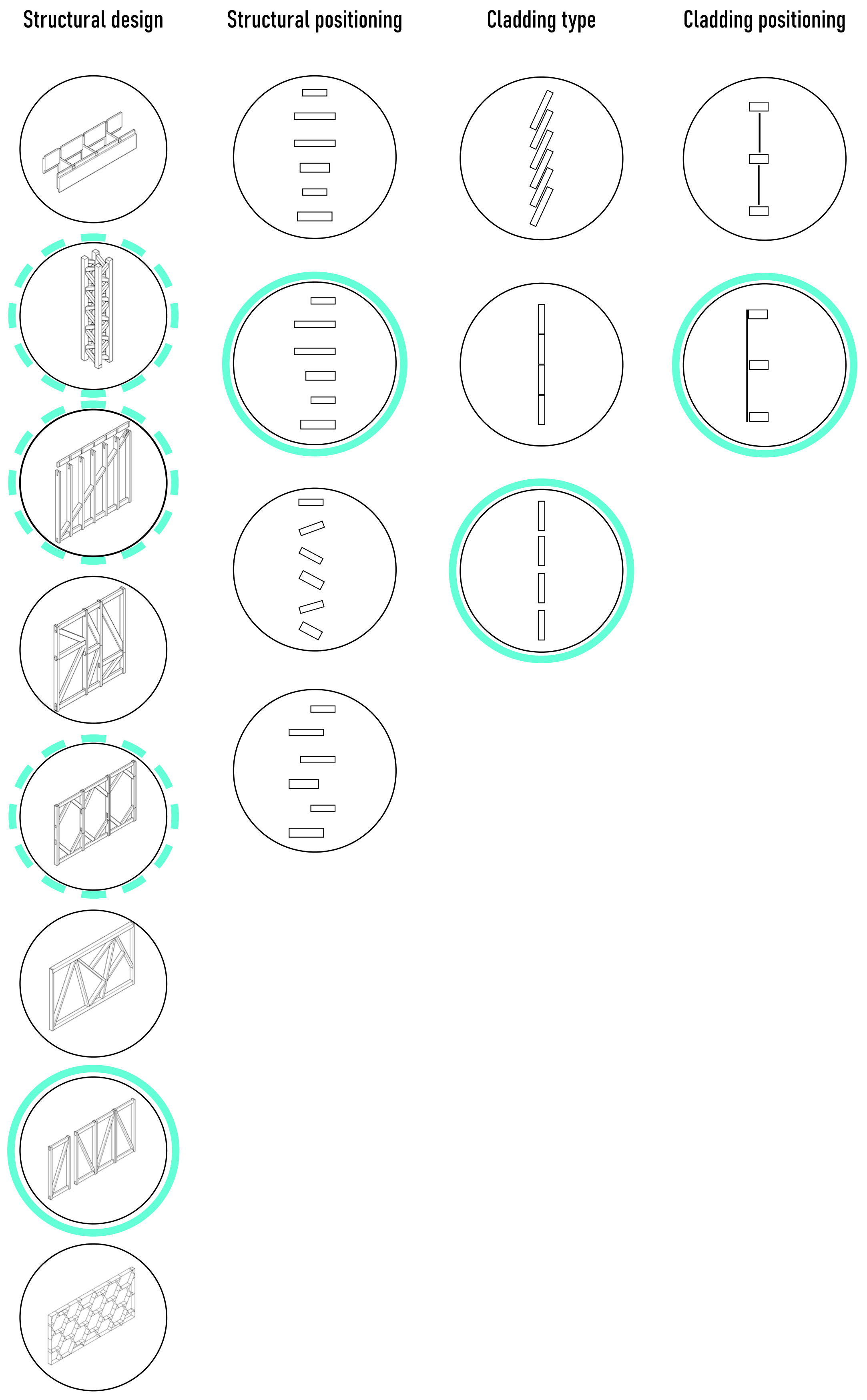


Figure 6.12 Highest scoring design solutions (own ill)

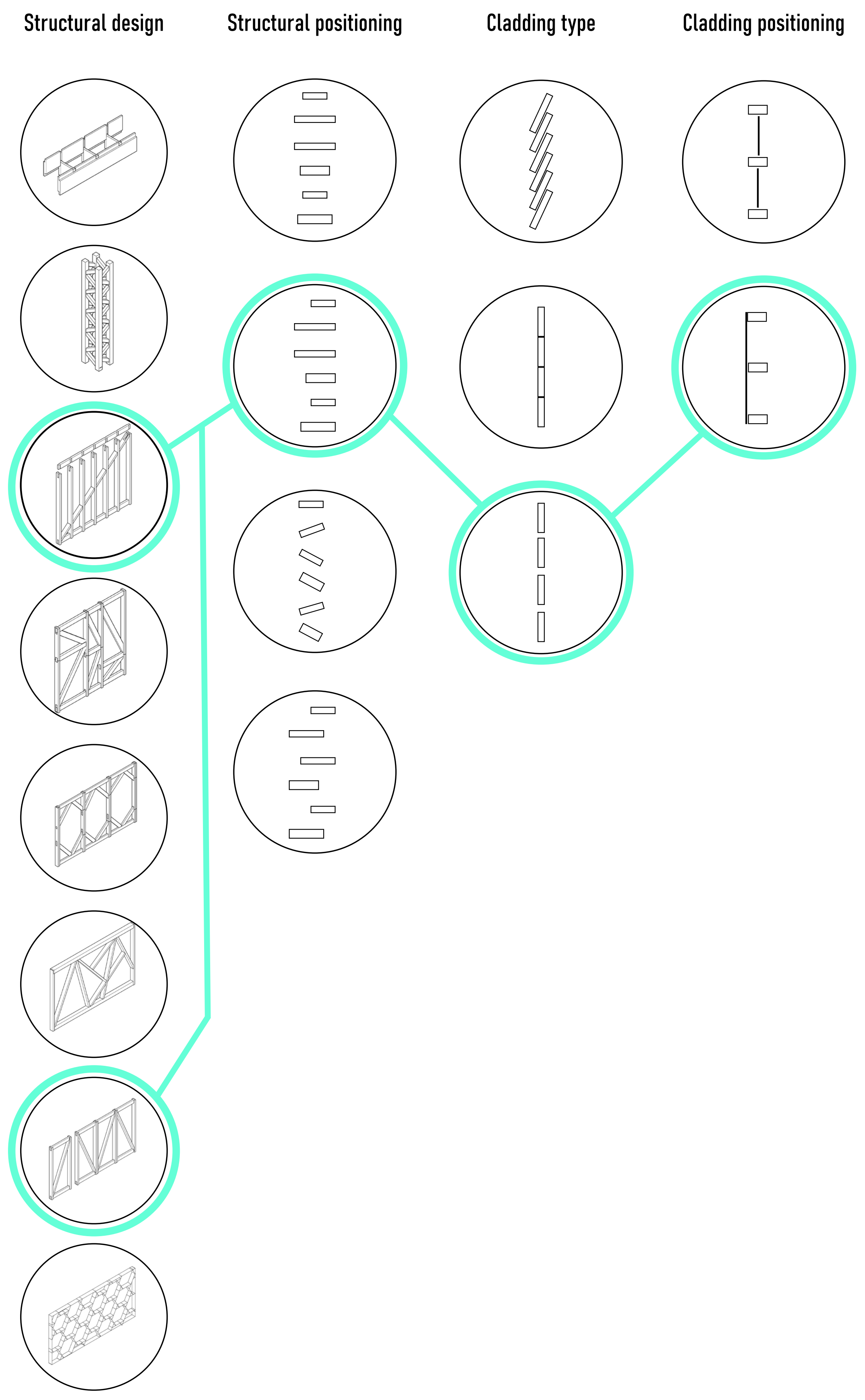


Figure 6.13 Most suitable design solutions first iteration (own ill)

6.6 NEW DESIGN PROBLEMS

The selected structural infill, structural positioning, cladding type, and cladding positioning resulted in more detailed design problems.

Bracing
To make the construction stiff, bracing is required. Different ways of placing the bracing are described and researched.

- Expected material loss
- Total material use
- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Flexibility of pieces
- Disassembly
- Loading efficiency
- Complexity of connection
- Full length usage

Vertical joint positioning
It will not always be the case that the stud will consist of one single piece of waste wood. Therefore, a vertical joint needs to be applied. Besides looking at how to join the two pieces, it is necessary to research the positioning relative to each other.

- Expected material loss
- Total material use
- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Disassembly

Vertical joints
Multiple types of joints are researched and described, varying from simple to more complex vertical joints.

- Expected material loss
- Assembly complexity
- Machine time
- Machine complexity
- Complexity of connection

Corner joint
The stud, the bracing, and the horizontal beam will connect/intersect with each other in some corners. Multiple options for the positioning of the connection are researched and described.

- Expected material loss
- Assembly complexity
- Machine time
- Machine complexity
- Complexity of connection

Center to center distance
Having a varying center to center distance can have a positive impact on the total material waste.

Cladding connection
There are different ways to connect the cladding to the substructure. The solutions will have an impact on both the cladding and the substructure.

- Expected material loss
- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Flexibility of pieces
- Disassembly

Connection substructure to structural framing
The substructure needs to be attached to the structural infill (studs). Multiple connections are described and researched.

- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Disassembly
- Complexity of connection

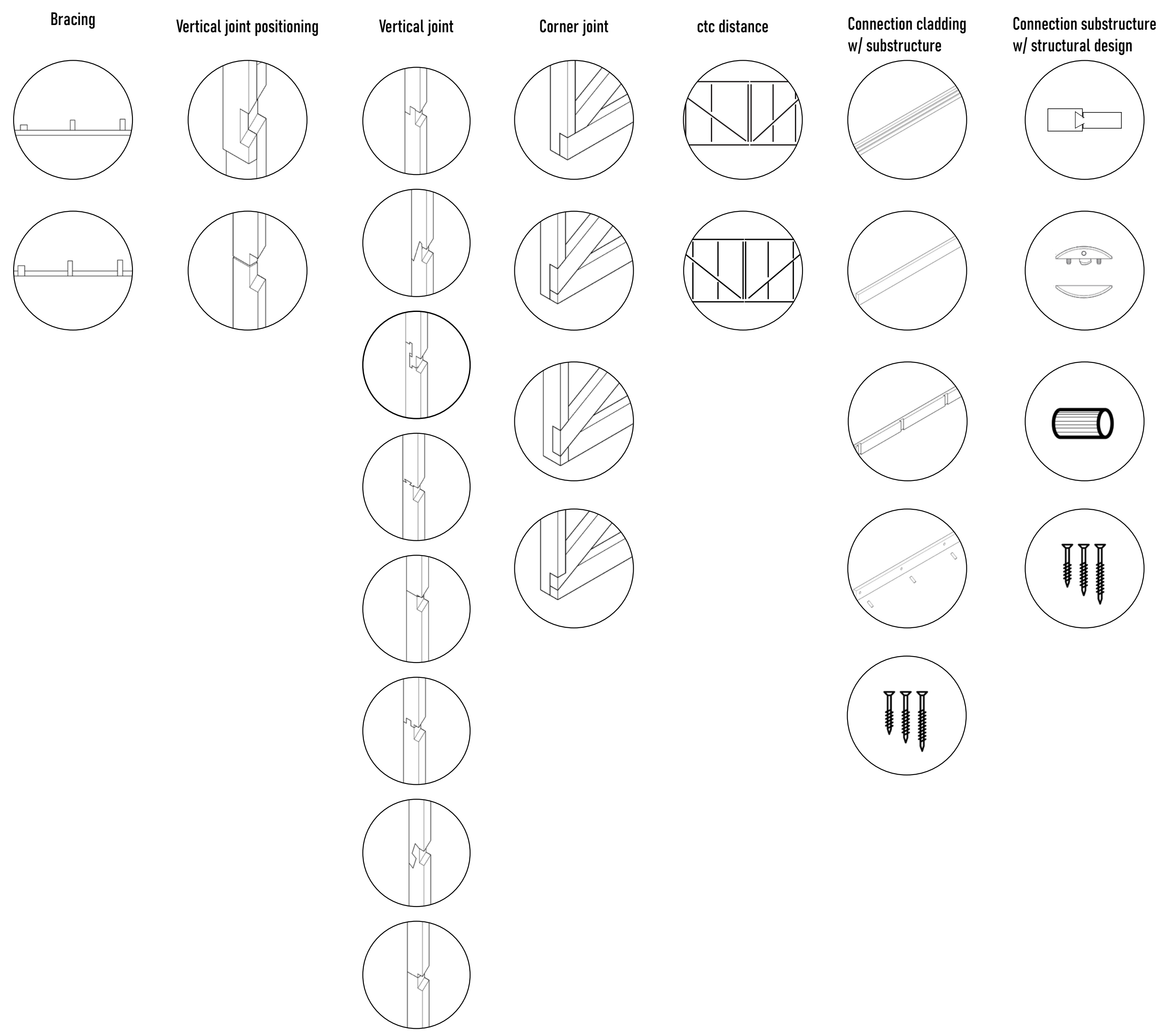
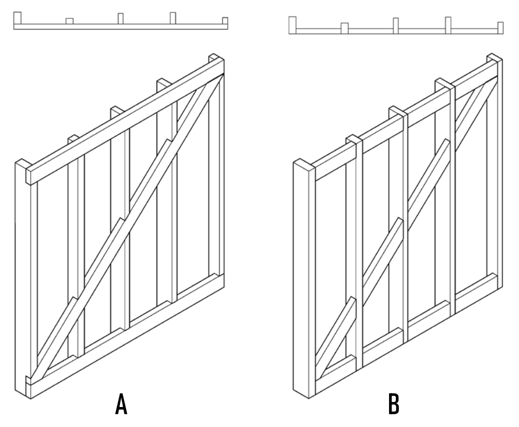

Bracing
Vertical joint positioning
Vertical joint
Corner joint
ctc distance
Connection cladding w/ substructure
Connection substructure w/ structural design

Figure 6.14 Second iteration of design problems with each possible solution (own ill)

6.6.1 BRACING:

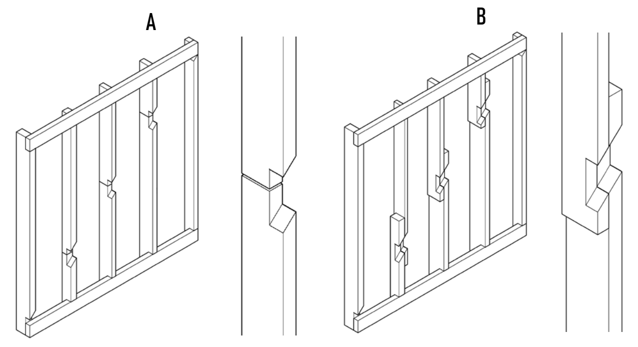


BRACING

	A	B	Weight
Expected material loss	4	2	3x
Total material use	3	3	3x
Assembly complexity	4	3	2x
Assembly time	4	1	1x
Machine time	4	1	1x
Disassembly	4	1	2x
Full length usage	4	2	3x
	57	**31**	

Two options are examined for the bracing: continuous (A) and fragmented bracing (B). Continues bracing consists of one single piece of wood while with fragmented bracing, the bracing is being interrupted by the studs, resulting in a higher number of connections. An additional downside of fragmented bracing is that the pieces for the bracing are the same length. Using pieces of the same length increases the chance of ending up with additional waste when selecting the wood from the database.

6.6.2 VERTICAL JOINT POSITION:



VERTICAL JOINT POSITION

	A	B	Weight
Expected material loss	4	2	3x
Total material use	4	2	3x
Assembly complexity	2	2	2x
Assembly time	3	3	1x
Machine time	4	2	1x
Machine complexity	4	4	1x
Disassembly	3	2	2x
	45	**29**	

For the vertical joint position, the decision can be made between a butt joint (A) and a side joint (B). With a side joint, the load must be transferred through a connection that will cause rotation. This will result in an unnecessary engineering problem that does not occur with a butt joint. A butt joint needs to hold the two pieces above each other so that the load can be transferred due to compression.

6.6.3 VERTICAL JOINT:

Different types of existing wood joints are described and researched. The joints are derived from standard wood joints that are used in furniture building construction. When the joint can be made without rotating the wood, the machine time and machine complexity are significantly reduced.

VERTICAL JOINT

	A	B	C	D	E	F	G	H	Weight
Expected material loss	4	4	3	4	3	2	4	4	3x
Assembly complexity	4	3	2	3	4	1	4	4	2x
Machine time	4	4	3	4	3	1	4	3	1x
Machine Complexity	4	3	4	3	4	1	3	3	1x
Complexity of connection	4	2	2	2	4	1	4	4	2x
	36	**29**	**24**	**29**	**32**	**12**	**35**	**34**	

Joint A:
This is a simple joint that can be manufactured from one axis but would rely on additional connectors to secure the joint in place firmly.

Joint B:
Joint B is a dovetail joint; this joint can also be manufactured from one axis but is more likely to fail due to buckling.

Joint C:
The benefit of joint C is the ability to take tensile forces. However, the number of cuts and machine time is increased.

Joint D:
The possibility of joint D is limited by the width of the stud. A smaller width will result in a more fragile joint.

Joint E:
Joint E Is designed in such a way that it can take compression and remain locked in place. However, the material loss is increased when the joint's height is higher than the height of the bracing intersecting the joint.

Joint F:
Joint F can take compression and tension, but the high complexity will significantly increase machine time.

Joint G:
The two slanted surfaces, in combination with the diagonal bracing, will result in a self-locking connection. However, this will require multiple axis milling if the piece of wood is being processed flat.

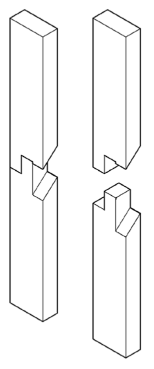
Joint H:
This joint will have the same problem as joint D, only in the depth direction.

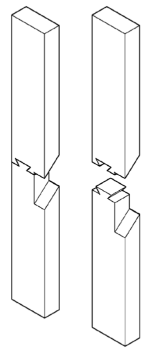
6.6.4 CORNER JOINT:

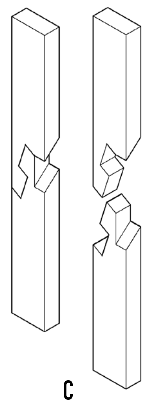
CORNER JOINT

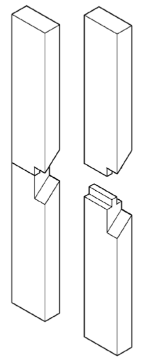
	A	B	C	D	Weight
Expected material loss	4	1	2	1	3x
Assembly complexity	4	4	4	4	2x
Machine time	4	3	2	2	1x
Machine Complexity	4	4	2	4	1x
Complexity of connection	4	4	4	4	2x
	36	**26**	**26**	**25**	

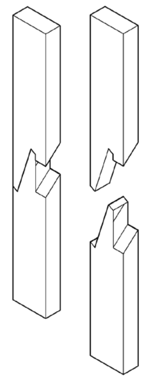
There are different ways for the connection in the corner. Design A scores the most point due to its simplicity. There are only modifications required on the vertical stud and the bracing and this allows the horizontal beam to be completely reused in its original length.

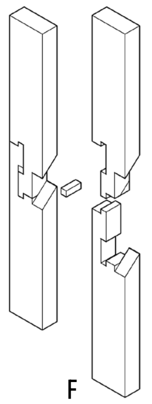

A

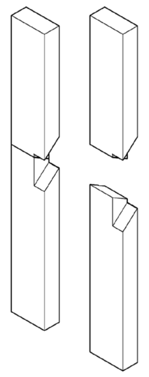

B

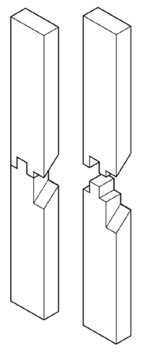

C

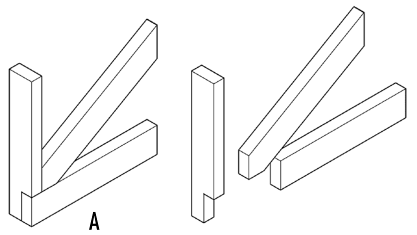

D

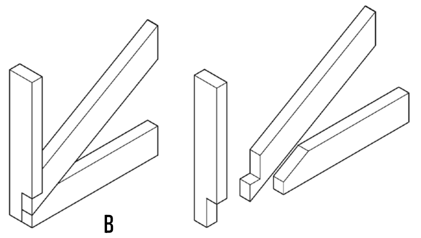

E

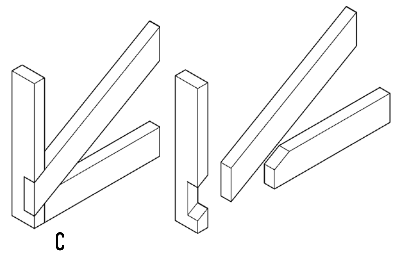

F

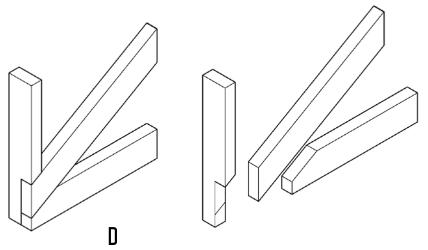

G


H


A


B


C


D

6.6.5 CENTER TO CENTER DISTANCE

CENTER TO CENTER

	A	B	Weight
Expected material loss	2	4	3x
Assembly complexity	4	2	2x
Assembly time	4	3	1x
	18	**19**	

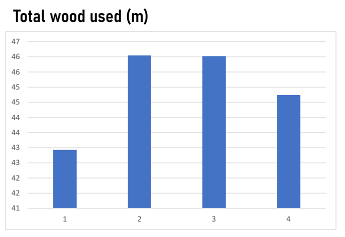
This design problem is discovered during the prototyping phase, where the concept proposal was made parametric.

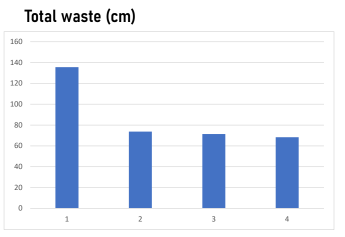
For this specific structural infill, a varying center to center distance can reduce additional waste. The diagonal bracing determines the length of the two pieces of wood that create the stud. When this distance can vary, the most suitable position can be chosen that generates the least amount of waste based on the available wood at that specific moment.

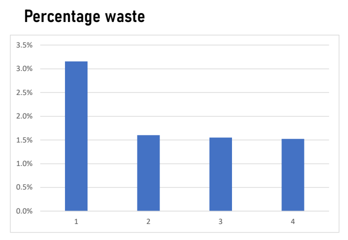
With different scripts and changing database sizes, research is conducted to see the influence of having a variable center to center distance.

Allowing the distance between the studs to vary frequently can positively impact the amount of waste. It can decrease the amount of waste by 50%. However, the amount of waste created in the first place with a fixed distance between the studs is already meager, especially when the database's size keeps increasing. The benefit of the waste reduction is not equal to the additional difficulties this varying center to center results in the assembly. In the future, this can be resolved with robotic manufacturing.

DATABASE SIZE: 100



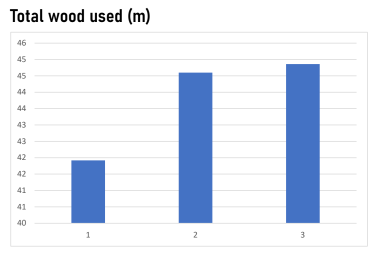


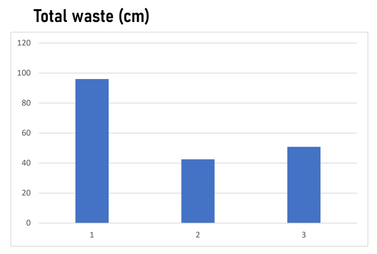


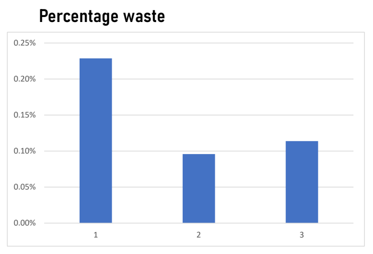
	length (m)	waste (cm)	
Script 1	42.93	135.7	3.2%
Script 2	46.05	73.7	1.6%
Script 3	46.02	71.4	1.6%
Script 4	44.74	68.3	1.5%

Scripts:
1. Fixed centre-centre (c-c) distance
2. Continue on best c-c option within set range
3. Random c-c for every stud, repeat x times, pick best option
4. Try every c-c possible within set range

DATABASE SIZE: 1000







	length (m)	waste (cm)	
Script 1	41.92	9.6	0.229%
Script 2	44.60	4.2	0.096%
Script 3	44.86	5.1	0.114%
Script 4			

Scripts:
1. Fixed centre-centre (c-c) distance
2. Continue on best c-c option within set range
3. Random c-c for every stud, repeat x times, pick best option
4. Try every c-c possible within set range

6.6.6 CLADDING TO SUBSTRUCTURE

CONNECTION CLADDING TO SUB

	A	B	C	D	E	Weight
Expected material loss	3	2	2	2	4	3x
Assembly complexity	3	4	3	3	4	2x
Assembly time	2	4	4	3	3	1x
Machine time	4	4	3	3	4	1x
Machine Complexity	4	4	3	3	4	1x
Flexibility of pieces	4	3	3	2	4	3x
Disassembly	3	4	4	4	3	2x
	43	**43**	**39**	**35**	**49**	

The connection between the cladding and the substructure can affect the cladding as well as the substructure.

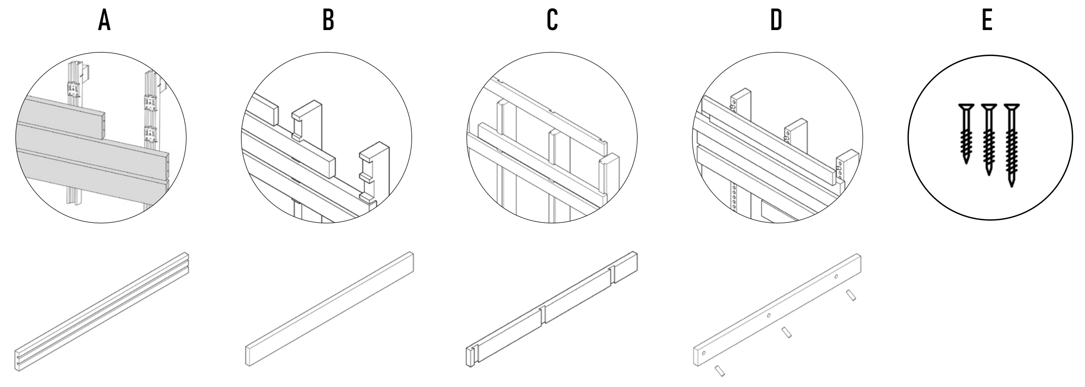
Bracket (A)
With the bracket connection, two parallel grooves are cut on one side of the cladding in the horizontal direction, over the cladding's entire length. The distance between the two grooves on each piece of cladding will always be the same. Therefore, the cladding can be reused on a different wall after its use. The bracket itself is placed on the substructure, and the cladding can be clicked on the substructure. A challenge with this system is connecting the bracket to the substructure at the required height.

Friction (B)
The friction connection allows the cladding to be unmodified. This enables the cladding to be used for a different purpose after its end of life. A downside is that the friction connection needs to be milled in the substructure, resulting in the substructure being unique for that specific wall element. Therefore, the substructure cannot be reused.

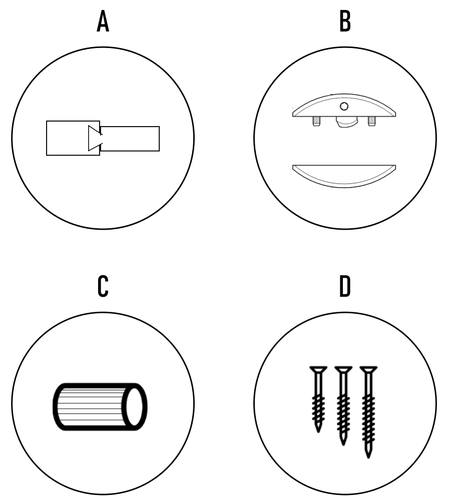
Dovetail (C)
The dovetail can be made with an all-wood connection. However, this system will not allow for individual replacement of the cladding. Also, working with the different center to center distances between the substructure will result in a unique positioning of the dovetail connection on the cladding. This decreases the reusability of the material.

Dowels (D)
Holes are milled in the substructure and the cladding where the dowels can be fitted in. Reusing the cladding or substructure would require new holes. This has a negative effect on the reusability of the system.

Screws (E)
Using screws is a common way to attach facade elements to a substructure. The screws also allow for individual facade plank replacement.



6.6.7 CONNECTION SUBSTRUCTURE TO STRUCTURAL INFILL



CONNECTION SUB TO STRUCTURAL

	A	B	C	D	Weight
Assembly complexity	4	3	4	4	2x
Assembly time	2	2	4	4	1x
Machine time	2	3	2	4	1x
Machine Complexity	1	2	4	4	1x
Disassembly	2	4	3	3	2x
Complexity of connection	2	3	4	4	2x
	21	**27**	**32**	**34**	

The biggest challenge with connecting the substructure against the structural infill is the foil that acts as a water barrier between the substructure and the structural infill. However, the foil is designed so that it will not tear or rip if a cut in the foil is made to make room for the connection element.

The depth of the substructure can vary between 50-110mm. There are screws available that extend this range and allow for an easy and fast connection between the substructure and the structural infill.

A smart connection such as a lamello would result in a more elegant connection and easy disassembly. However, it would require a slot in the substructure and the structural infill where the lamello can be placed. This increases the assembly and machine time.

A dowel would also require a pre-milled hole in the substructure and structural infill to make the connection work. Just like the lamello connection, this increases the machine time and assembly time.

A dovetail connection generates the additional challenge of fitting the foil between/inside the dovetail connection. To get this right, this can require intensive manual labor.

6.7 CONCLUDING

Circled in a solid line is the solution with the highest score. Solutions that have a similar score have a dashed circle.

The choice for the bracing and vertical joint position was quite clear. There the highest scoring options also were the most logical.

For the vertical joint, the best solution is open for debate. Design A is a proper solution due to its simplicity. However, design G is a smarter solution because the two opposite slanted sides lock the two pieces with the diagonal bracing. The downside of these opposite slanted sides is an increased machine complexity. Design H is also simple but is more likely to break due to its smaller connection. In the end, design A is preferred for its simplicity.

For the corner connection, the most significant factor was the fixation of the horizontal beam and the bracing perpendicular on the stud. This specific demand resulted in the most suitable option not be the option with the highest score. Design D requires fewer cuts than design B and is therefore preferred.

The varying center to center distance has a positive impact on the amount of waste. However, with this research, the studs' position was chosen to minimize waste. In reality, if the distance between the studs can vary, it will be determined by the loadbearing capacity of each stud. After this process, a selection procedure can be used to minimize the amount of waste. It is unknown how much impact it will have. This specific research is not conducted for this paper because structural calculations were outside the scope of this research.

Screws are used for the connection between the cladding and the substructure. This is not the most elegant solution but, it was by far the most practical solution. The same applies for the connection between the substructure and the structural infill. After the end-of-life the screws will leave the smallest hole. This is not the same for the other solutions.

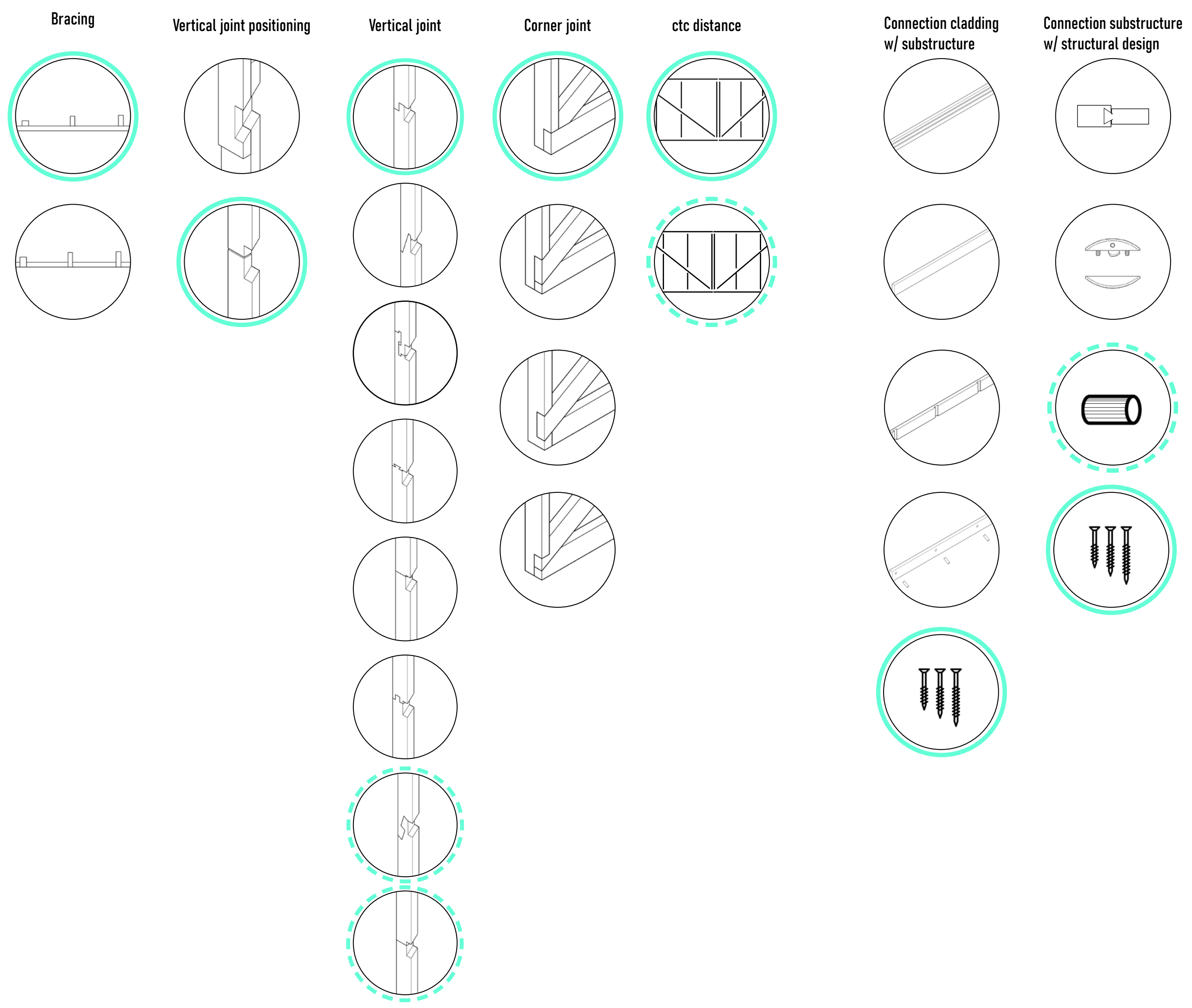

Bracing
Vertical joint positioning
Vertical joint
Corner joint
ctc distance
Connection cladding w/ substructure
Connection substructure w/ structural design

Figure 6.15 Highest scoring design solutions (own ill)

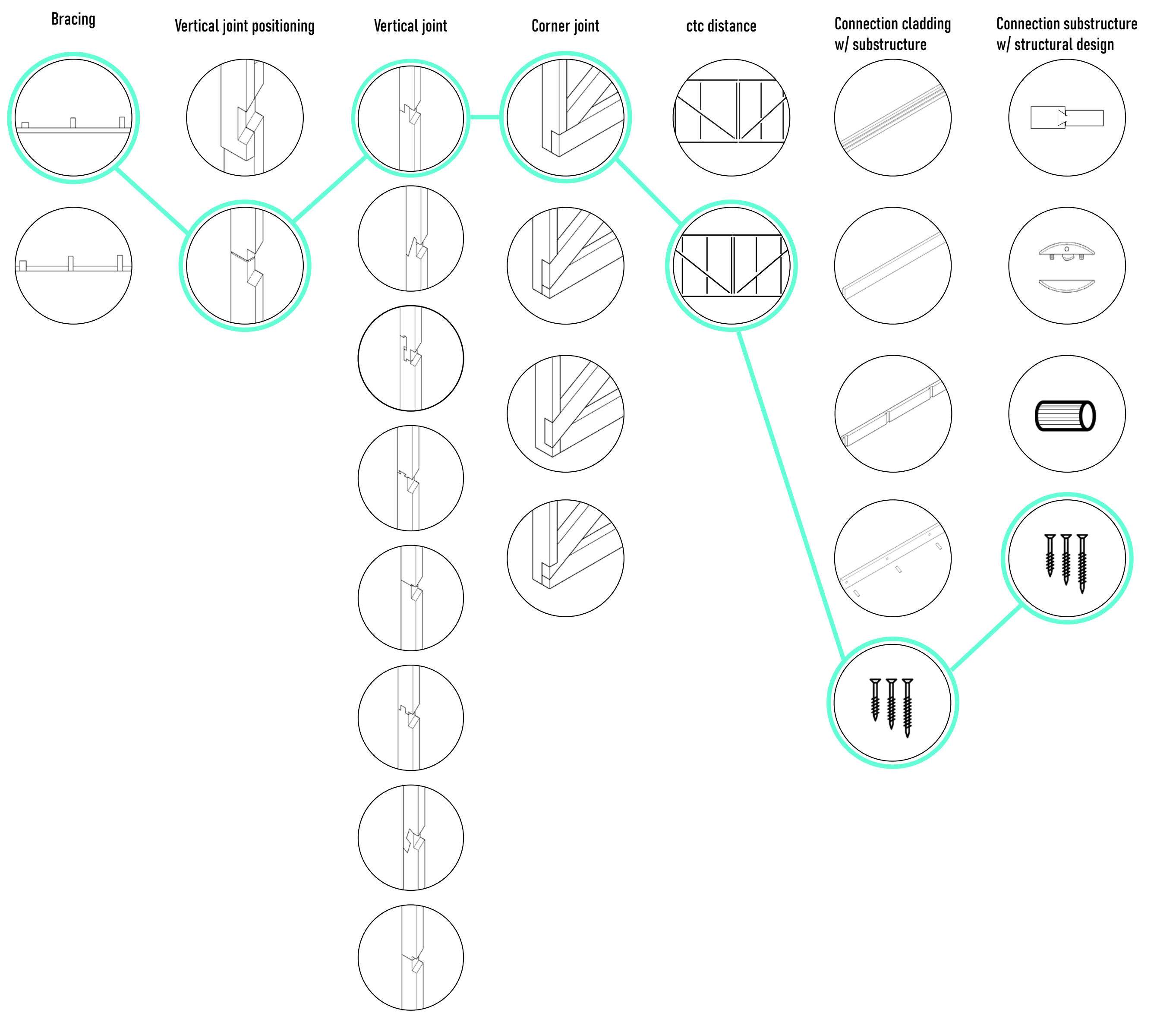

Bracing
Vertical joint positioning
Vertical joint
Corner joint
ctc distance
Connection cladding w/ substructure
Connection substructure w/ structural design

Figure 6.16 Most suitable design solutions second iteration (own ill)

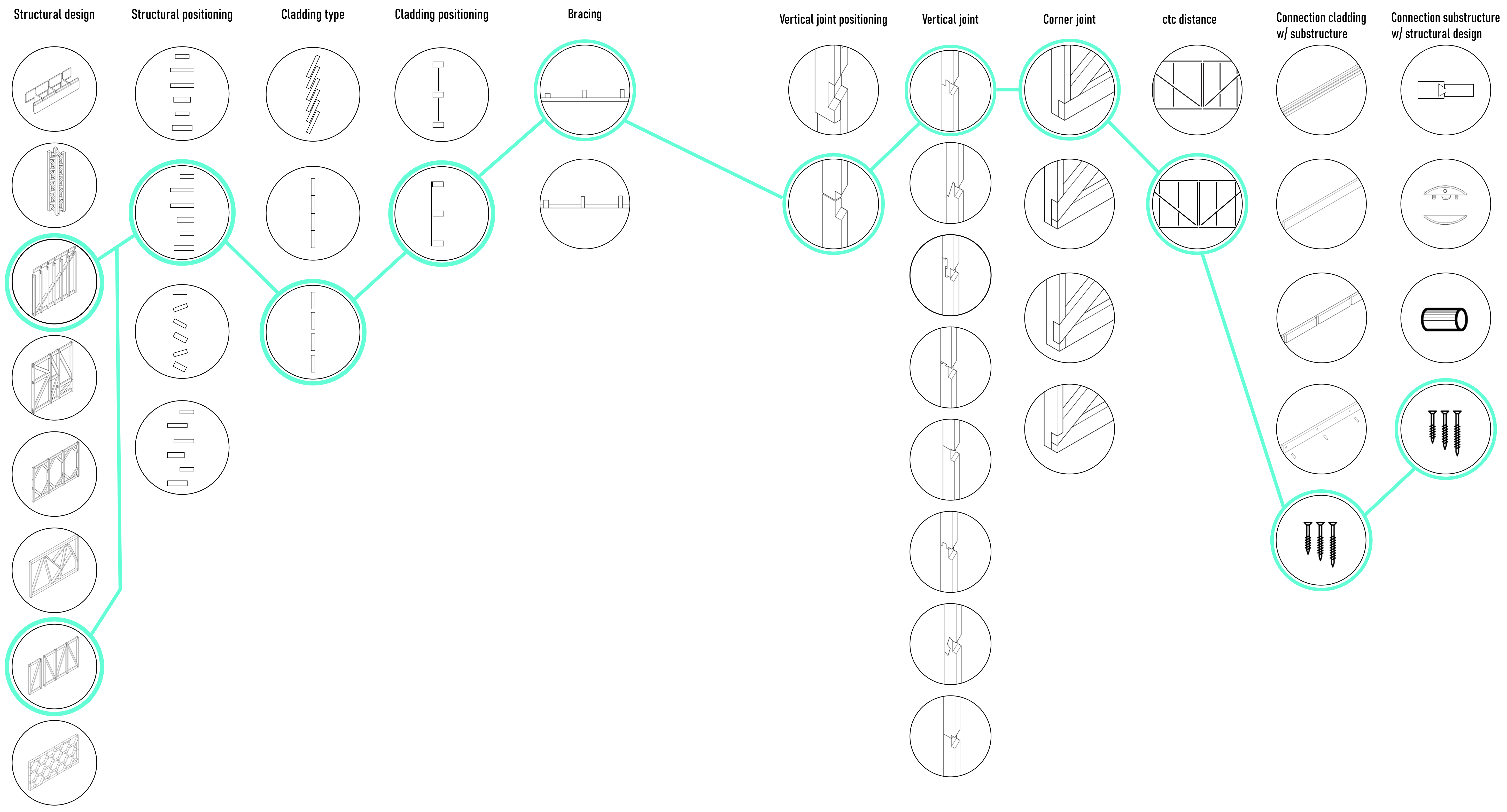


Figure 6.17 Overall most suitable design solutions (own ill)

6.8 PROTOTYPING

For the prototyping an environment was created where multiple tests could be performed (figure 6.28). The grasshopper script has a timer programmed that can re-run the canvas every X milliseconds. This way a new dataset can be generated and the result with that dataset is written to an excel file. A flowchart of the process can be seen in figure 6.27.

6.8.1 WATER RESISTANCE FOIL

During the prototyping phase, a problem occurred with placing the water-resistant foil (figure 6.29). With the selection procedure, the top stud may have a lower depth than the bottom stud. This choice is made because it will save additional waste. However, this difference in depth can create problems when attaching the foil.

For the final design, it is interesting to research if it is better to find a new solution to the foil problem or deal with the additional waste created by this restriction. The table in figure 6.25 below shows the outcome of this research with the values in meters. The prototype was tested with 150 different datasets. The width of the wall element is 3 meters and the height 2.5 meters. The database for structural infill, substructure, and cladding each consisted of 200 pieces of waste wood. So, in total the dataset consisted of 600 pieces of wood.

Having no restrictions means that the bottom and top studs can always be placed. It does not matter if the top stud is wider or the bottom stud has a lower depth. With the depth restriction, the top and bottom stud always need to have the same depth. The width can still vary. With the depth and width restriction, the top studs' width can only be the same or less of that from the top stud. Having these restrictions only affect the waste on the structural infill and the substructure, the cladding will stay the same.

On a first look, the adding of the restrictions do have an impact. Adding the same depth makes the amount of waste increase by 39%. However, if you compare this with the total meters needed to construct the wall, the difference is almost negligible. The waste increases by 23cm, while the wall needs 40 meters for the structural infill and substructure. Without the restrictions, 1.5% of the wood used is waste. With the restriction, this is 2%. The additional waste is increased by 0.5%.
Due to these results, the decision is made to restrict the studs' depth instead of finding a solution to the foil problem.

	Total waste	Total used	waste struc + sub	used struc + sub	waste struc	waste sub	waste cladding	struc used	sub used	cladding used
No restrictions	1,80 m	95,24 m	0,59 m	40,15 m	0,28 m	0,31 m	1,21 m	24,84 m	15,31 m	45,41 m
Depth	2,03 m	95,42 m	0,82 m	40,39 m	0,47 m	0,35 m	1,21 m	25,03 m	15,35 m	45,41 m
Depth & width	2,13 m	95,49 m	0,92 m	40,48 m	0,56 m	0,36 m	1,21 m	25,12 m	15,36 m	45,41 m

Figure 6.18 Restriction research(own ill.)

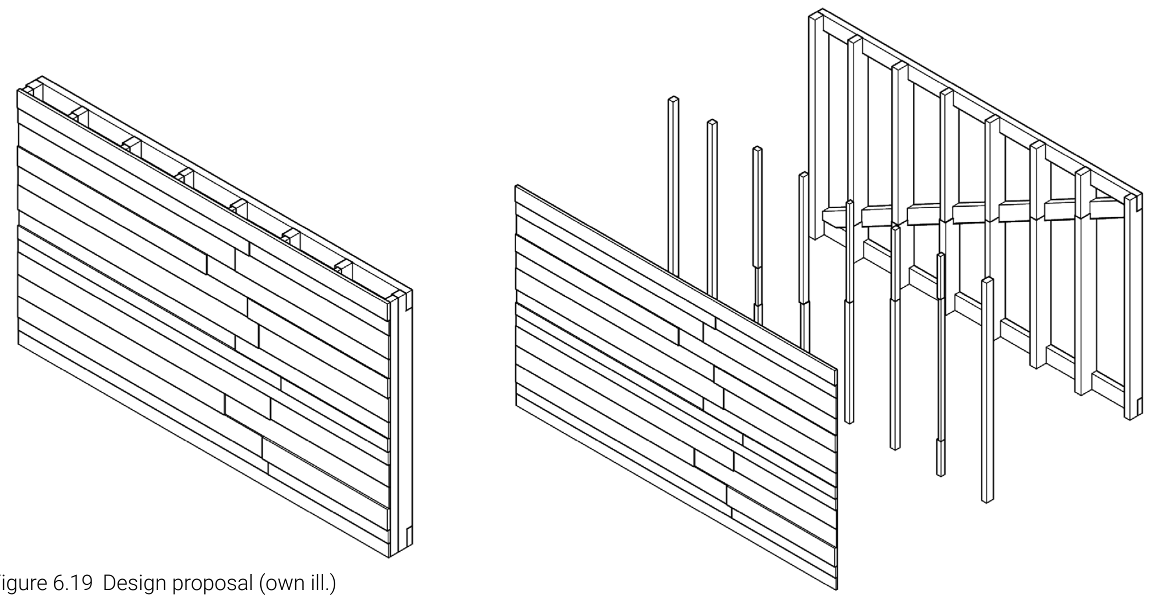
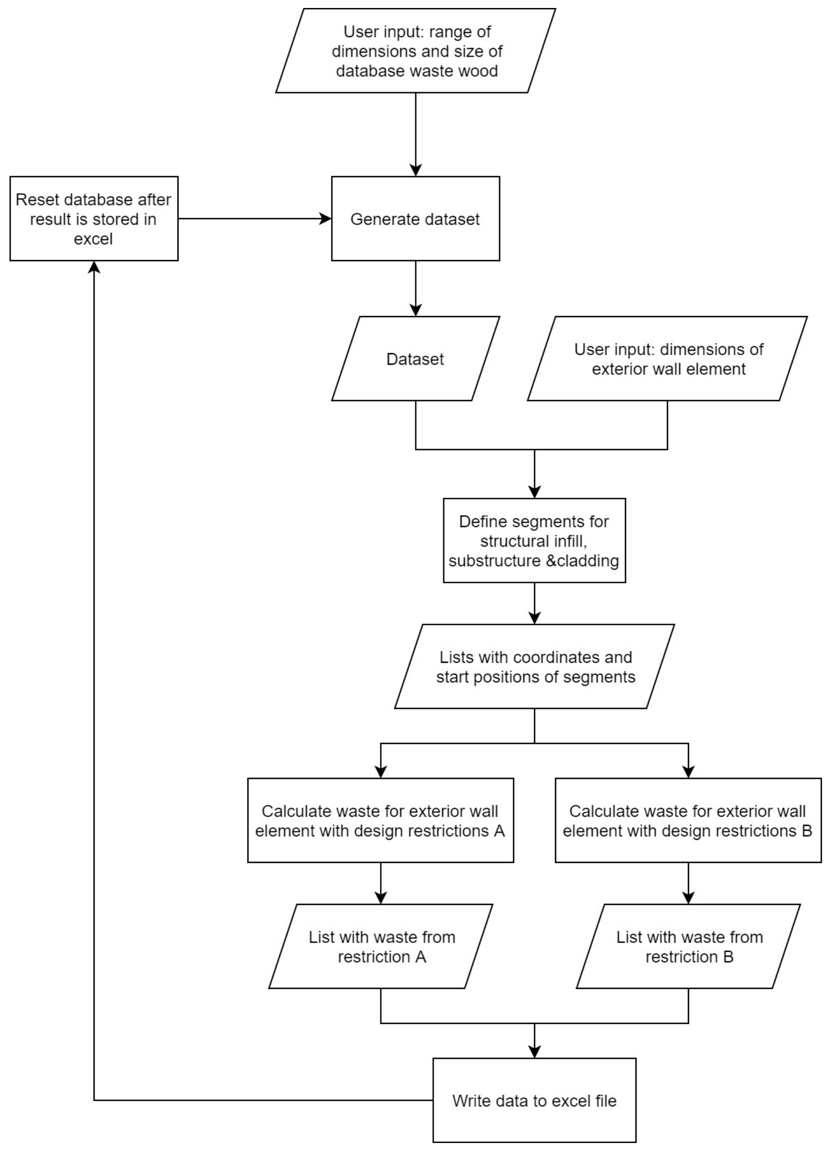


Figure 6.19 Design proposal (own ill.)


User input: range of dimensions and size of database waste wood
Reset database after result is stored in excel
Generate dataset
Dataset
User input: dimensions of exterior wall element
Define segments for structural infill, substructure &cladding
Lists with coordinates and start positions of segments
Calculate waste for exterior wall element with design restrictions A
Calculate waste for exterior wall element with design restrictions B
List with waste from restriction A
List with waste from restriction B
Write data to excel file
Figure 6.20 Digital playground flowchart (own ill.)

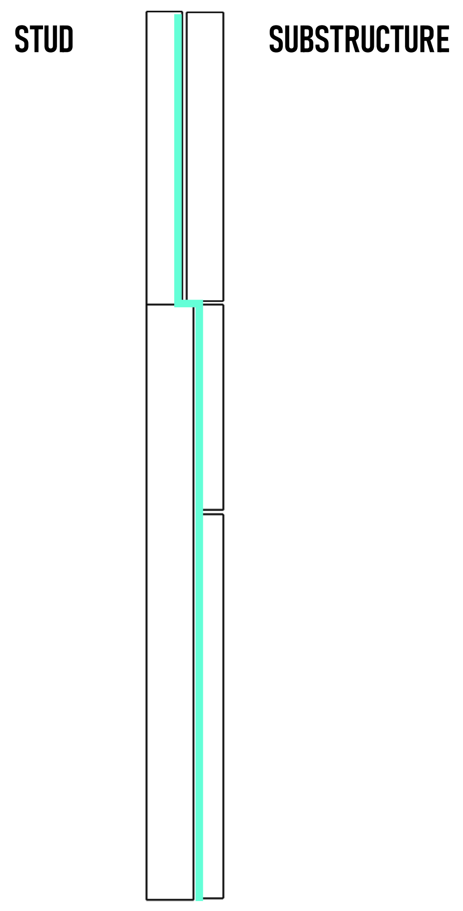


Figure 6.22 Water resistance foil problem (own ill.)

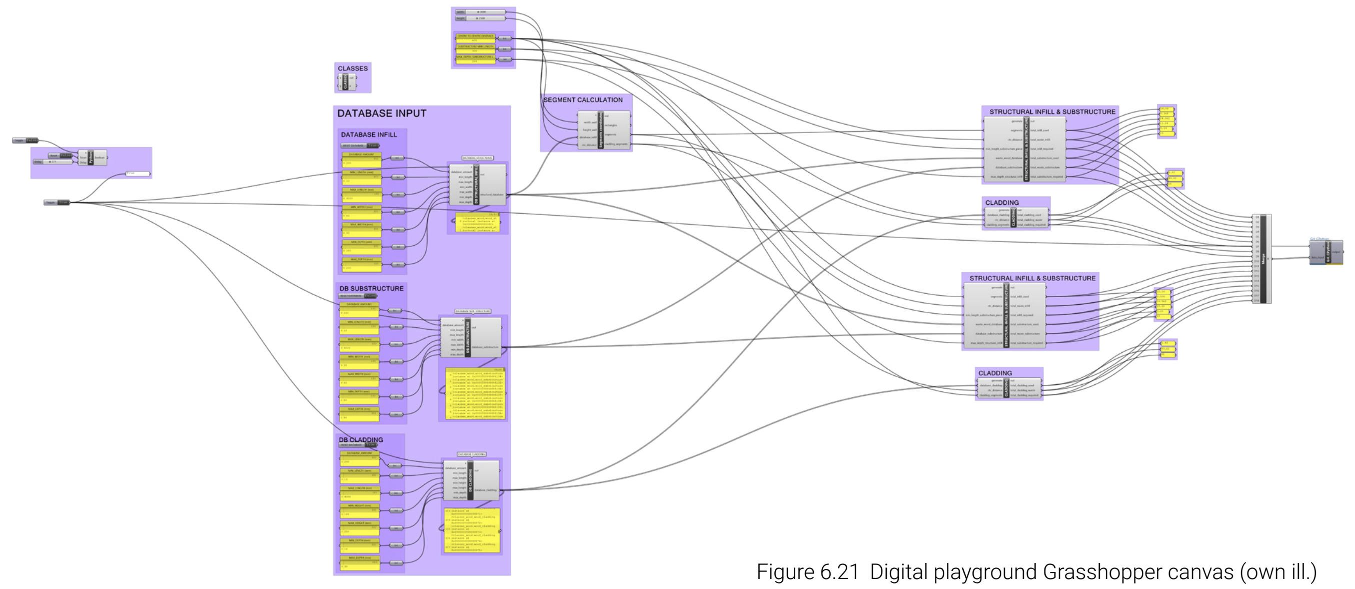
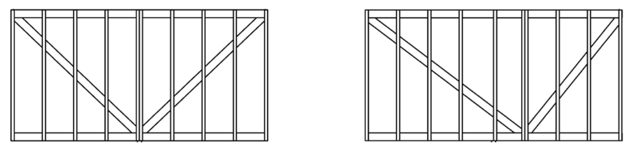


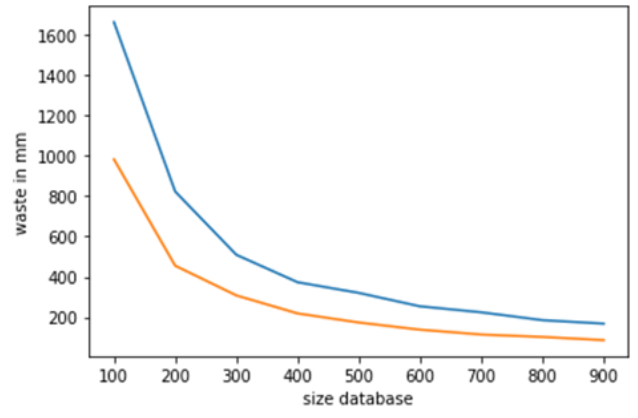
Figure 6.21 Digital playground Grasshopper canvas (own ill.)

6.8.2 SEGMENT WIDTH

During the prototyping phase, some insight was obtained that possibly could reduce the additional waste. The positioning where the bracing meets when the wall element requires two or more diagonal braces. This happens when the length required is longer than the length available. It would be logical to let the diagonals meet in the middle because this will minimize the total meters required for the bracing (left example). However, this results in a symmetric design. With a symmetric design, every piece of wood is needed twice. With scripting, the insight was obtained that symmetry will result in a higher total material waste. This happens because the chance that two pieces with the same dimensions are available is lower.



When testing the two options in the prototype the following results occurred. The height of the wall is 2.5 meters and the width 4.8 meters with a center-to-center distance of 600 mm. The left variant is split into two equal segments with a width of 2.4 meters the right variant is split in segment with 1.8 meters and 3 meters. The left option requires 6931 mm of bracing and the right option 6986 mm. The right option requires 55mm more bracing. However, this is only 0.8% increase of length required. Running the two variants trough the prototype gave the following results:



To see the effect of an increasing database size the number of waste wood in the database is increased in steps of 100. The waste reduces exponential when the database increases. The reason for this is that more wood is available to be selected matching the exact required length. The blue line represents the symmetric design, the orange line represents the asymmetric design. Having an asmmyetric design clearly has a benefit over a symmetric design.

The design option on the right requires more meters for the bracing, but it will generate less waste and therefore uses less material in total. This principle resulted in the criterion for flexibility in pieces. A design that uses a broader range of unique dimensions will result in a lower total waste. This criterion can be used in the matrix based on the subjective opinion of the designer. It is easier to determine how flexible the design is based on a feeling than to program it into a script. Here, scripting resulted in an insight that is subjectively used within the decision matrix.

6.9 FINAL DESIGN

The wall element is built with a vapor diffusion permeable building method. A benefit from this method is the natural regulation of humidity, increasing the indoor climate. An additional benefit is that the vapor barrier foil between the interior and the structural infill is not required. For this building method to work, every element in the wall needs to be vapor permeable. Therefore, the foil that is used to act as a water barrier needs to be vapor permeable.
For the insulation, water barrier foil, and interior finish existing products are used to ensure the designed wall element meets the building code in the Netherlands. Materials are chosen that have the least impact on the climate and follow the philosophy of this thesis.

6.9.1 MATERIALS

For the foil, the Pro Clima Solite Front Quattro FB, flame retardant is used. This foil also has fire resistance properties.

For the insulation, Gutex Thermoflex wood fiber insulation is used. The material has a lambda of 0.036 W/(m.k). On the lowest depth, the studs have a depth of 160mm. This equals to an Rc of 4.44. wood fiber insulation is produced from rest wood from the wood industry. The panels are made with an ecological friendly adhesive, making the insulation suitable for recycling.

For the inside finish, Sono plasterboards are used with a thickness of 30mm this equals to a Rc = 0.75. The plasterboard allows the construction to be plastered.

To calculate the Rc value of the construction the studs need to be added to the calculation. The width of the stud is on average 65 mm with a lambda of 0.13. A center-to-center distance of 600mm results in the following Rc value:
Rc = (0.5 * 0.11 + 4.44 * 0.89) + 0.75 = 4.76

This Rc value meets the requirements of the building code. The calculated Rc value is calculated in the worst scenario. The studs can have a depth between 160 and 200mm. In the best scenario, all the studs hava a depth of 200. This result in a Rc value of:

Rc = (0.5 * 0.11 + 5.56 * 0.89) + 0.75 = 5.75

The Rc value of the exterior wall will be in the range of 4.76 – 5.75.

6.9.2 WEIGHT CALCULATION

The final design combined with the exterior wall element tool can calculate how many wall elements can be constructed from waste wood. Chapter 3 showed that there is 370.000.000 kg of solid B waste wood that has the potential to be reused. The exterior wall element tool with the final design calculates that a wall element with a height of 3 meters, a width of 4,8 meters and a center-to-center distance of 600 mm weighs, on average, 503 kg. For the weight calculation, the density of the wood is 500 kg/dm3. This is the average density of pinewood, the most used construction wood in the Netherlands.

For a 100m2 single floor square house with a height of 3 meters, a 120m2 wall is required. 120m2 exterior wall is 4.192 kg. If 10% of the solid B-wood can be reused, 8.826 houses can be constructed every year. This amount can have a significant contribution to the one million additional houses.

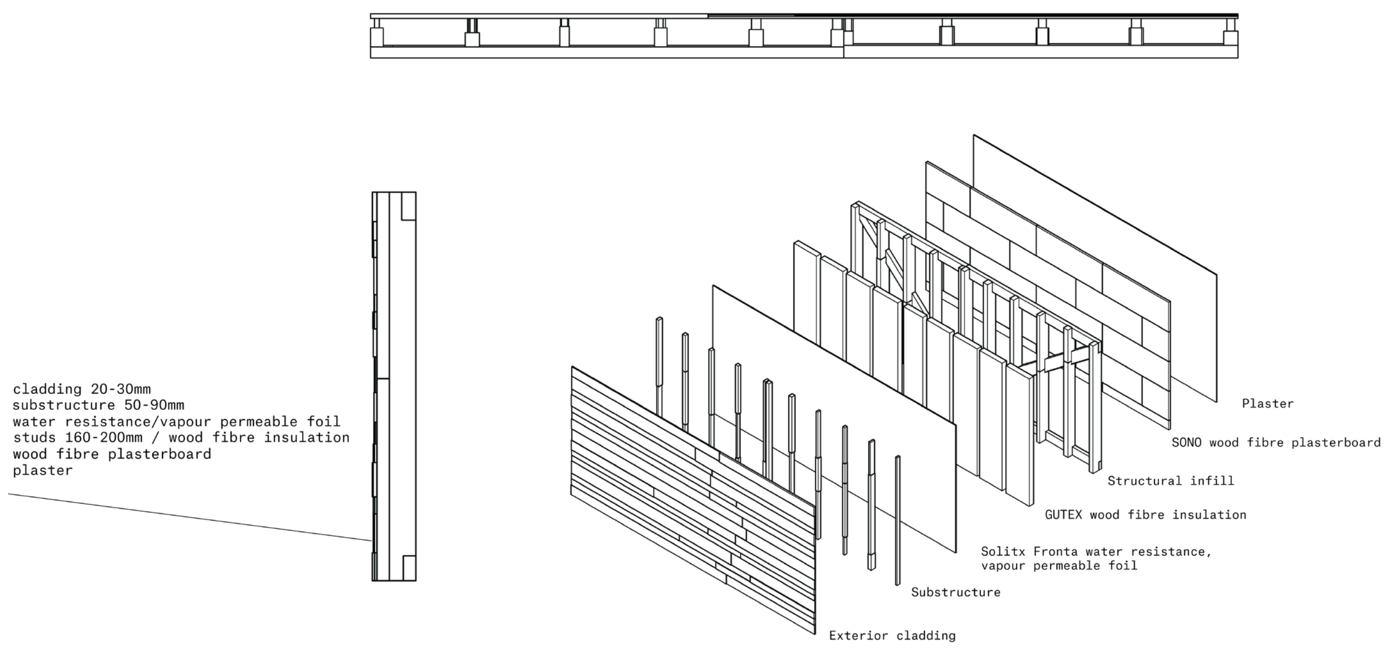
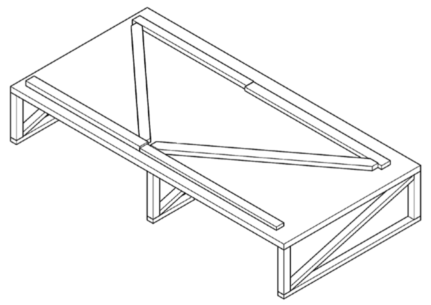
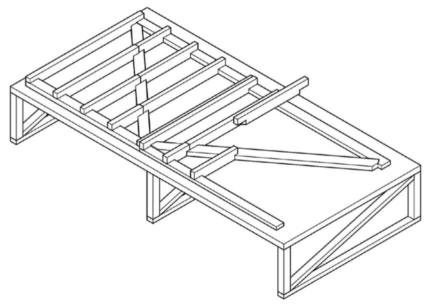

cladding 20-30mm
substructure 50-90mm
water resistance/vapour permeable foil
studs 160-200mm / wood fibre insulation
wood fibre plasterboard
plaster
Plaster
SONO wood fibre plasterboard
Structural infill
GUTEX wood fibre insulation
Solitx Fronta water resistance, vapour permeable foil
Substructure
Exterior cladding

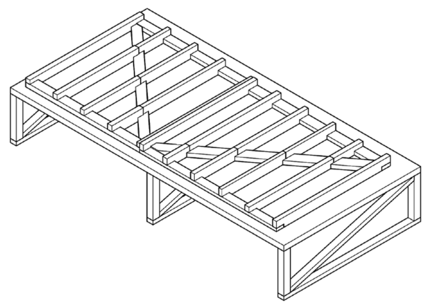
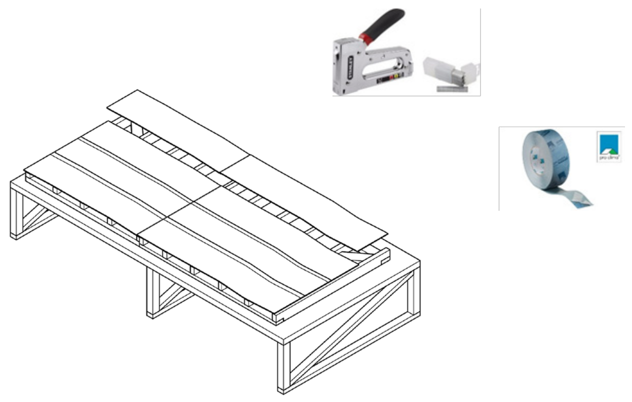
Figure 6.23 Final design (own ill.)

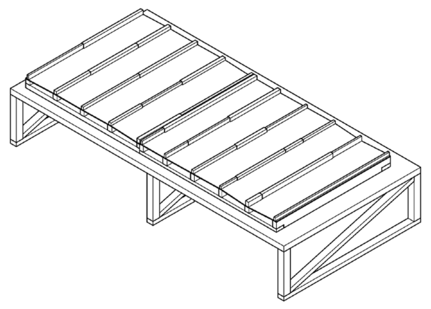
6.10 ASSEMBLY SEQUENCE

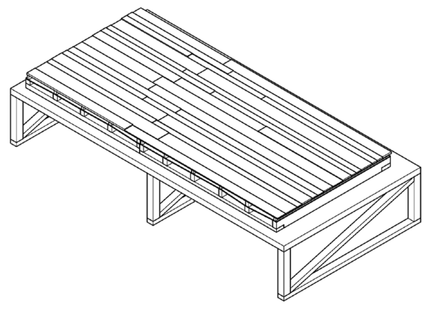
In the image sequence below a suggestion is made how the wall could be assembled.

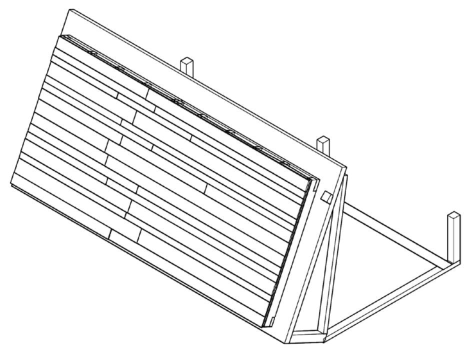

The horizontal frame and the bracing is placed on a flat surface.

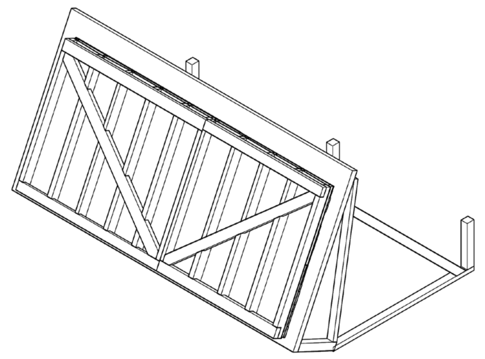

The studs are placed on top of the horizontal frame and the bracing and screwed into place.

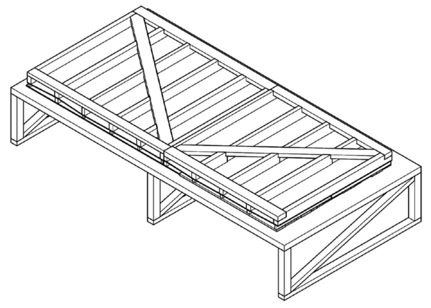


Apply the Pro Clima Solitex Fronta Quattro FB foil. Use a tacker to attach the foil to the studs. The foil needs to be laid in horizontal strokes with overlapping sides of 10 cm. Attach the two overlapping foils with double sided tape and finish the seam with a sealing tape.

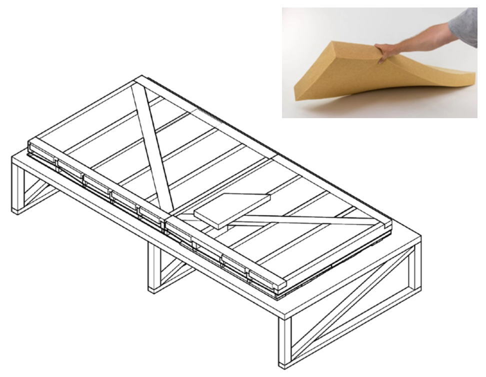

Use screws to attach the substructure to the studs. It is normal for the screws to penetrate the foil.

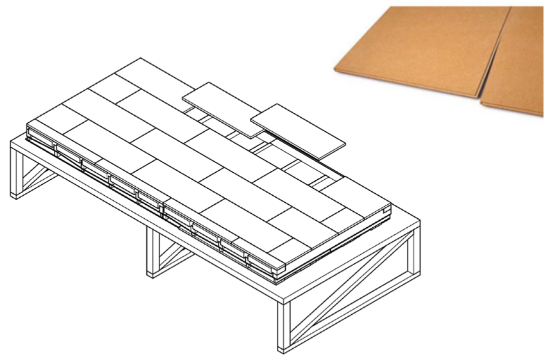

Attach the cladding to the studs. Leaving atleast a gap of 30 mm between the cladding and the foil for a ventilated cavity.


Rotate the workbench vertical so that the wall element can be flipped and placed horizontal again.


The studs are placed on top of the horizontal frame and the bracing and screwed into place.


The studs are placed on top of the horizontal frame and the bracing and screwed into place.


Apply the Gutex thermoflex wood fiber insulation.


Finish the wall element by placing the SONO wood fiber plasterboards. When the wall element is on the building site it can be plastered for the final finishing.

6.11 REFLECTION ON THE METHODOLOGY:

In this chapter, the altered methodology with scripting is being analyzed, and a conclusion is given about its effectiveness.

6.11.1 THE METHODOLOGY

The methodology used in this thesis consisted of eight stages.

a. Design Problems
b. Criteria
c. Alternative selection
d. Concept comparison
e. Concept proposal
f. Prototyping and testing
g. Sub-solutions
h. Final design

6.11.2 CRITERIA

The criteria are a direct extension of the objectives in chapter two and are determined during the criteria phase. These criteria define the essentials and fundamentals of the design. For each design problem, different solutions are tested against this beforehand specified set of weighted criteria. Different solutions get points ranging from 1 to 4 for each criterion, and these points are then multiplied with the weighted factor of the criterion. For the design of the exterior wall element, the following design criteria were specified:

- Expected material loss
- Total material use
- Full length usage
- Flexibility of pieces
- Assembly complexity
- Assembly time
- Machine time
- Machine complexity
- Complexity of connection
- Disassembly
- Loading efficiency

A design problem can have multiple design solutions. Each solution can score points for each criterion. In general, more points results result in a more suitable solution. All the possible solutions for a given design problem and all the applied criteria were put in a decision matrix. The matrix can show the designer the most suitable design solution based on the score. With the existing methodology of van der Knaap & van Veen these points are based on a subjective scale. For example, with this methodology, the criterion total material use would use the scale: high amount of material use (1 point) – low amount of material use (4 points). The difference between 2 or 3 points is subjective, and there is not a quantified threshold. Therefore, the points given are based on a feeling. This feeling is backed up with experience and research so that the designer can substantiate it.

An alternative from the original methodology was made during the criteria & alternative selection. As a result of this, scripting was used to quantify the criteria' thresholds and, therefore, create an objective scale. Each solution was feed through the script, and the outputs could be placed in the scale. For the criterion of total material use, the scale can be quantified as follow: 1 (> Z%) – 2 (Y-Z%) – 3 (X-Y%) – 4 (<X%). The percentage is the output compared with a reference. For this criterion, every output was compared with the material use of a standard TFC element.

From the eleven criteria, four of them were suitable for quantifying:

- Expected material loss
- Total material use
- Machine time
- Flexibility of pieces

With the alternative selection stage, every design solution is tested upon the weighted criteria and given a score in the decision matrix.

6.11.3 DECISION MATRIX

During the concept alternative (step c) phase, each design solution was tested against the applied criteria in the decision matrix. It became clear that the original methodology was not that effective for every design problem. Sometimes the best solution was already so apparent that it almost felt useless creating the matrix. For example, this occurred with the cladding positioning problem. The choice was made between placing cladding in front of the substructure or placing it between the substructure. Filling in the matrix would give a confirmation. However, the number of criteria that the two solutions were tested on was limited. This can be a reason why the answer was so logical in the first place. The decision matrix can give the designer guidance with an intricate design problem where the solutions have different benefits and downsides. Here, a particular choice can have a substantial impact on other design problems. During the development of the wall element, this was not the case for some design problems.

6.11.4 ANALYZING THE ALTERED METHODOLOGY DURING THE ALTERNATIVE SELECTION STAGE

The goal was to write a script for the total material use, expected material loss, machine time, and pieces' flexibility. These scripts would be applied to multiple design problems. Eventually, only a script was written for the total material use criterion and applied to two problems, the structural infill problem and the decision between a variable or a fixed center to center distance.

With the structural infill problem, a script was written for eight possible design solutions. The output of the script gave the total material use of that design solution in meters. However, it became clear that the time it took to write the scripts did not weigh up against the benefit of having a quantified output from the scripts. If the scripts were written for the other suitable criteria, an additional 24 scripts were needed. This would be too time-consuming.

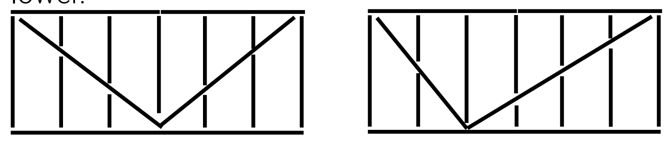
With the right background knowledge and experience, the designer can fill in the matrix and have a substantiated outcome. This outcome will not be as accurate as of the script, but it is enough to guide the designer in making the right decisions.

The scripts allow for easy adjustments of the parameters and, thereby, see its effect on the design solution. This tool can be useful during brainstorms because it generates an instant output. It also gives a clear overview of how the different parameters relate and the impact a change in the parameters had on the script's output. Before the script, this was just an estimated guess.

The script was more suitable for the decision with the variable center to center. Here there were only two design solutions. This resulted in fewer scripts that needed to be written in comparison with the structural infill problem. Again, the script confirmed an already logical estimation. With common sense, the conclusion can be made that having a varying distance between the studs would reduce the additional material waste. A downside is the increased complexity for assembly. A script can only define how much this positive impact is on the material waste and if it can compete with the downside of the increased assembly complexity. This increased complexity in assembly could be solved by implementing robotic assembly. However, the decision was made to exclude robotic assembly because this would mean the exterior wall element tool would be less accessible due to the need for robots for assembly. The assembly complexity remained on a subjective scale. Therefore, the output of the quantified total material use criterion still had to be compared with the subjective scale of the assembly complexity criterion. Therefore, the process remained subjective despite the script.

6.11.5 INSIGHT BY SCRIPTING

While writing the script, some insights were obtained that positively impacted the waste reduction and even altered the criteria. For example, the positioning where the bracing meets when the wall element requires two or more diagonal braces. This happens when the length required is longer than the length available. It would be logical to let the diagonals meet in the middle because this will minimize the total meters required for the bracing (left example). However, this results in a symmetric design. With a symmetric design, every piece of wood is needed twice. With scripting, the insight was obtained that symmetry will result in a higher total material waste. This happens because the chance that two pieces with the same dimensions are available is lower.



The design option on the right requires more meters for the bracing, but it will generate less waste and therefore uses less material in total. This principle resulted in the criterion for flexibility in pieces. A design that uses a broader range of unique dimensions will result in a lower total waste. This criterion can be used in the matrix based on the subjective opinion of the designer. It is easier to determine how flexible the design is based on a feeling than to program it into a script. Here, scripting resulted in an insight that is subjectively used within the decision matrix.

6.11.6 PROTOTYPING

During the prototyping stage, the script showed its potential. After the concept proposal, a new design problem emerged with the water barrier foil and the studs. In the first prototype, the bottom and top studs could have different depths. As a result of this, a kink happened in the foil between the top and bottom stud. This kink could result in problems during assembly. The script could be easily adjusted to restrict the top stud from having the same depth as the bottom stud. After running multiple iterations with changing database sizes, the result was that the additional waste resulting from these restrictions was insignificant. Therefore, a new solution was not required. Within the prototyping stage, the script can act as a playground, and with minor adjustments, multiple tests can be performed under different circumstances. This playground also allowed the selection procedure, where the script decides to pick which wood from the database, to be improved continuously—the script allowed for direct feedback.

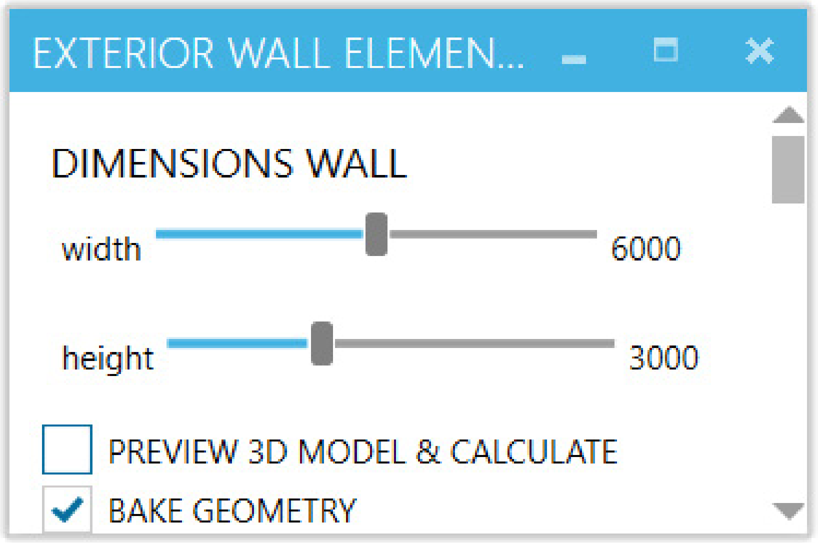
6.11.7 CONCLUSION ON THE ALTERNATED SCRIPT

Scripting did have a beneficial impact in guiding the designer with finding the most suitable design. However, the designer should consider if it is worth implementing it into the early stages of the design process. The impact of the script increases when the design gets more in detail.

The value of a script can also come to light when the designer wants to quantify certain decisions. The script is a playground, where experiments and small studies can happen within a specific base model. This has the most benefit when a design proposal is already offered.

Within the general script, these experiments about the influence of certain restrictions or small alterations can be programmed rather quickly because elements just have to be implemented or modified within the general scripts. Different restrictions can also be combined to see what the effect is. Here the value of a parametric script can be more useful for the designer to give some insights.

7 EXTERIOR WALL ELEMENT TOOL


EXTERIOR WALL ELEMEN...

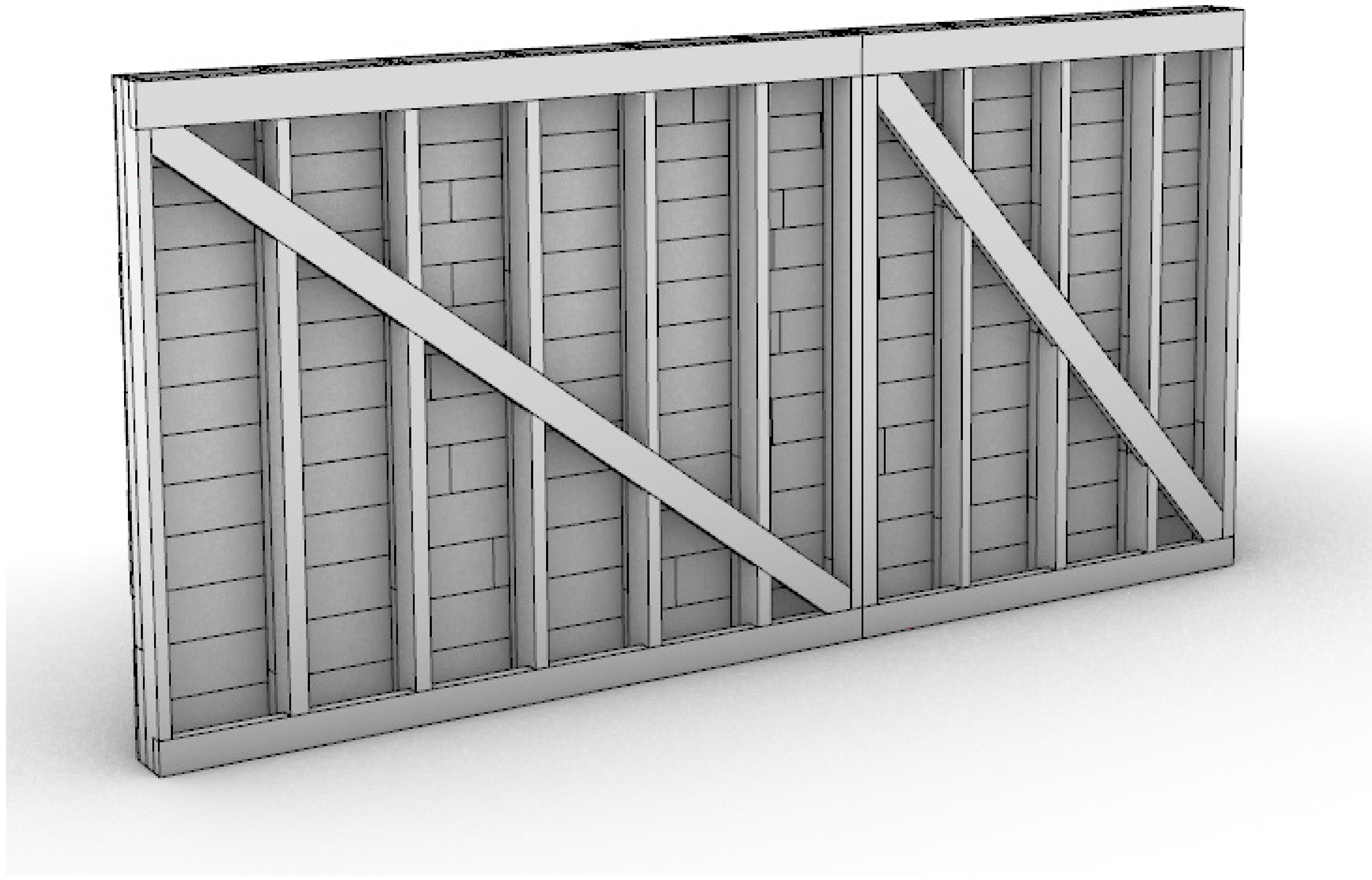
DIMENSIONS WALL

width 6000

height 3000

☐ PREVIEW 3D MODEL & CALCULATE

✓ BAKE GEOMETRY



7.1 TOOL PROTOTYPE

To create a proof of concept, a prototype of the tool is developed. The prototype is developed in a single Grasshopper file to retain a clear overview.

An overview of the Grasshopper canvas, the Python code and screenshots of the developed tool and all the python code can be seen in Appendix B,C & E.

A video with a demonstration of the exterior wall element tool can be seen on youtube:
https://youtu.be/bLwf9CknQGA

All the files can be downloaded from github:
https://github.com/JvE-TU/Waste-wood-graduation-project

The Human UI plug-in will create a user-friendly interface where the user can give the dimensions and properties of the wall as an input. After a 3d model is generated, the Human UI interface shows statistics about the amount of wood required for the design and how much waste is generated.

Inside the grasshopper file, a python script will generate a fake dataset and sent this set to an SQL database. Another script will read the database and select the waste wood that is required for the parametric model. This rather unlogic workflow will be for this prototype only, to prove that the parametric model can work with the data from the database.

To write Python scripts in Grasshopper the GH_Python and GH_CPython plugins are used. From now on, Python refers to GH_Python, and CPython refers to GH_CPython. GH_Python has a better workflow, while GH_CPython can import more external Python libraries. GH_Python is preferred, but GH_CPython is used where necessary.

With the programmer's given restrictions and the user's desired dimensions, multiple components in the grasshopper file will generate the 3d model. The script will also export a separate 3dm file for each modified piece of wood. These files can be easily translated to files that are suitable for digital manufacturing, such as CNC milling.

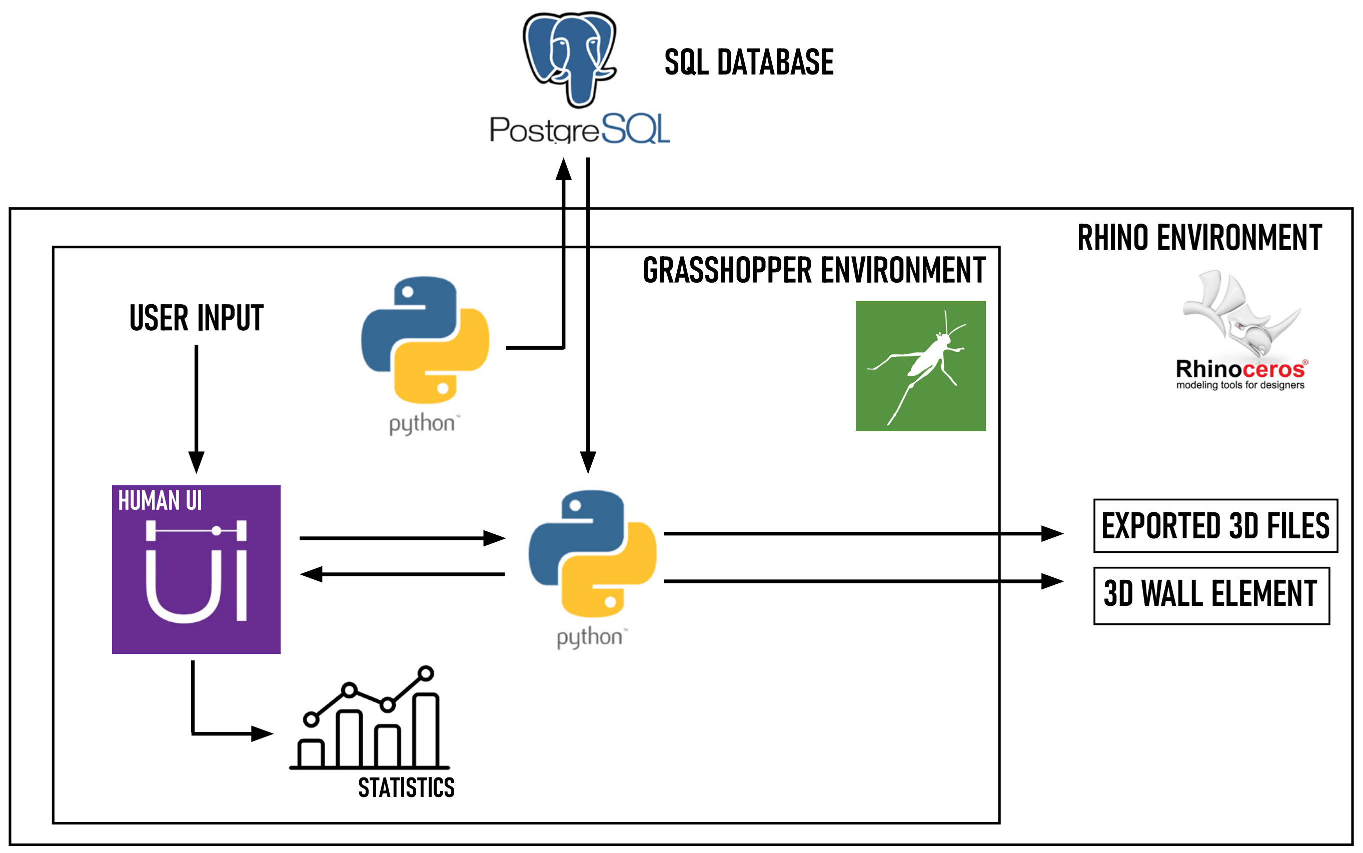


Figure 7.1 Design environment exterior wall element tool (own ill.)

This page is intentionally left blank

7.2 PROTOTYPE TOOL FLOWCHART

Figure 7.2 gives an overview of the complete prototype. The different functions will be explained in the following pages.

It starts with the user defining the desired dimensions and properties of the exterior wall element. The dimensions can be defined by adjusting sliders. There is a 2d feedback preview in Rhino so the user can see the proportions of the choices made.

When checking the box preview 3d model & calculate, the script will start running in grasshopper. A query is sent to the database, and extracting three lists with waste wood suitable for the structural infill, substructure, and cladding.
The parameters of the database (number of pieces, dimensions) are adjustable by the programmer in this grasshopper file because there does not exist a real database with waste wood. This allows for experimenting with different database sizes.

The script splits the wall into multiple segments based on the wall's dimensions and the choice for a window and/or door. The longest available wood limits the max-width of a segment for bracing.
The structural infill, substructure, and cladding function determines which wood is selected from the database lists to generate the least additional waste. The programmer can add additional restrictions to this selection procedure. These restrictions can improve certain design factors such as assembly complexity but can also result in additional waste. Examples of restrictions are:

- The top studs cannot be wider than the bottom stud.
- Segments cannot have the same width.
- Bottom and top studs need to have the same depth.
- Center to center distance needs to be fixed between studs.

The scripts will generate three types of lists that will be used later.

1. List with selected wood from the database lists.
2. List with Breps for 3d preview and baking in Grasshopper.
3. Lists with waste, remaining of the pieces of wood that are not used in the design.

Combining the lists with selected wood results in the total amount of wood used. The same can be done for the waste. The results of these calculations are shown in the Human UI interface. These results give the user direct feedback on the inputs and allow for adjusting when the user is not satisfied.

When the user is satisfied with the design, the model can be exported. Every piece of wood used in the model is exported as a separate 3dm file, where the name of the file corresponds with the id of the piece of wood in the SQL database. This way, the physical piece of wood can be linked with the 3dm file when it is used.

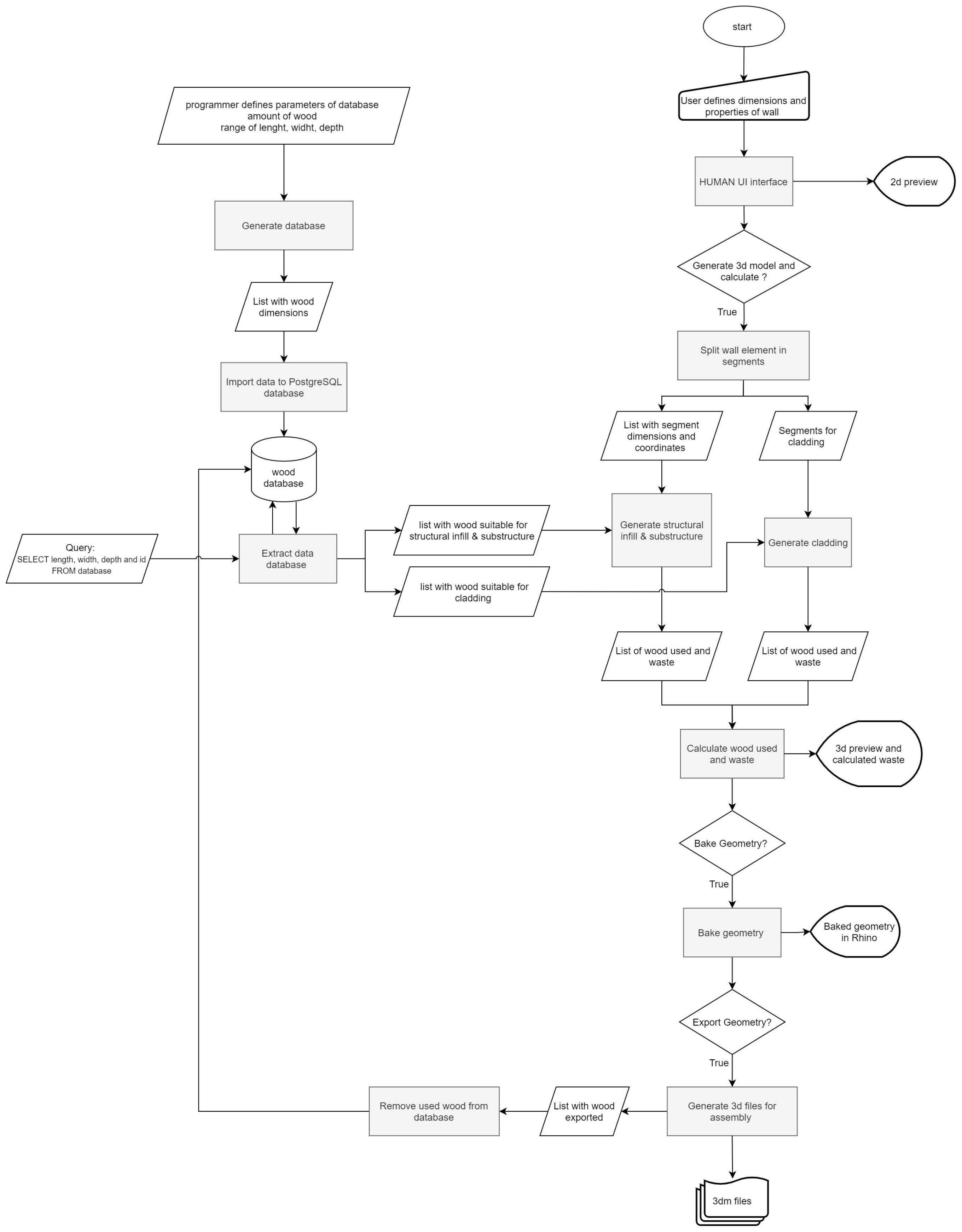

start
User defines dimensions and properties of wall
programmer defines parameters of database amount of wood range of lenght, widht, depth
HUMAN UI interface
2d preview
Generate database
Generate 3d model and calculate ?
True
List with wood dimensions
Split wall element in segments
Import data to PostgreSQL database
List with segment dimensions and coordinates
Segments for cladding
wood database
list with wood suitable for structural infill & substructure
Generate structural infill & substructure
Query: SELECT length, width, depth and id FROM database
Extract data database
Generate cladding
list with wood suitable for cladding
List of wood used and waste
List of wood used and waste
Calculate wood used and waste
3d preview and calculated waste
Bake Geometry?
True
Bake geometry
Baked geometry in Rhino
Export Geometry?
True
Remove used wood from database
List with wood exported
Generate 3d files for assembly
3dm files

Figure 7.2 Flowchart exterior wall element tool (own ill)

7.3 STEP BY STEP EXPLANATION

7.3.1 DATABASE CREATION

The script is added to the appendix in E.2 - E.5.

There is no official dataset available about waste wood dimensions in the Netherlands. Therefore, it is necessary to create a dummy database. With the current script, it is possible to adjust the range of dimensions of the pieces of wood created for the database. For this prototype, the dimensions of the wood are randomized within the given parameters. The range of the parameters are defined by previous research in this thesis. The number of pieces of wood in the database is also adjustable. This is implemented to what the effect of the database size has on the additional waste.

To send the data to the database from a grasshopper environment CPython is used with the psycopg2 module. This module allows the CPython component to connect with a PostgreSQL database and edit the database.

When the script connects to the database, it removes all previous data and resets the id of the database. This allows the new entered dataset to start with an id of 1.

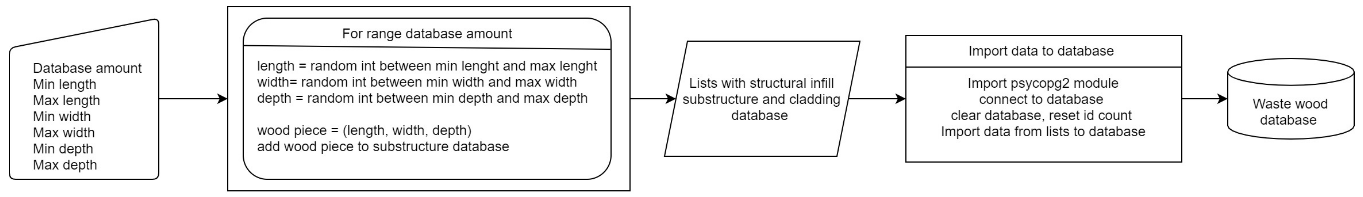
7.3.2 EXTRACT DATA FROM DATABASE

The script is added to the appendix in E.6 & E.7

With a different CPython component, the data from the database is extracted. The wood is sorted into three lists:

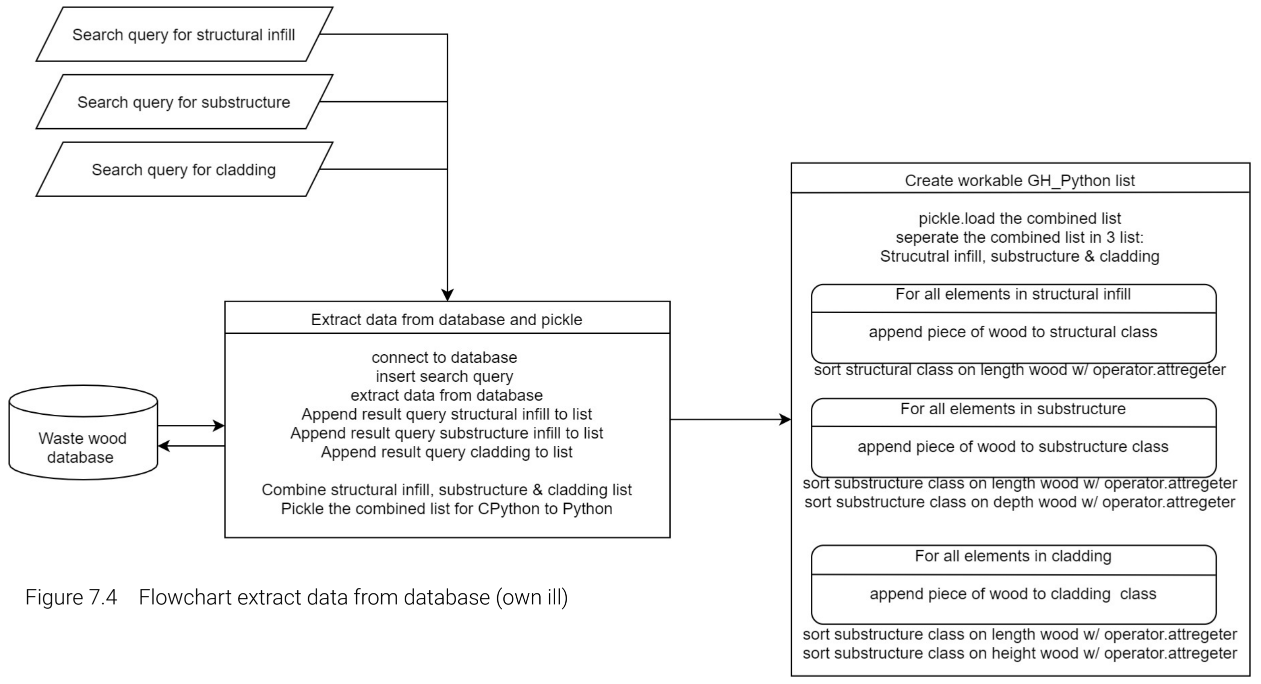
- Structural infill database
- Substructure database
- Cladding database

To bring the data from the CPython to the Python component, the Pickle module is used. This is not ideal because the user needs to assign a directory/location on the pc where the data can be stored so that it can be read by the Python component. The CPython cannot directly transfer the list to the Python component.



Database amount
Min length
Max length
Min width
Max width
Min depth
Max depth
For range database amount
length = random int between min lenght and max lenght
width= random int between min width and max width
depth = random int between min depth and max depth
wood piece = (length, width, depth)
add wood piece to substructure database
Lists with structural infill substructure and cladding database
Import data to database
Import psycopg2 module
connect to database
clear database, reset id count
Import data from lists to database
Waste wood database

Figure 7.3 Flowchart database creation (own ill)



Search query for structural infill
Search query for substructure
Search query for cladding
Waste wood database
Extract data from database and pickle
connect to database
insert search query
extract data from database
Append result query structural infill to list
Append result query substructure infill to list
Append result query cladding to list
Combine structural infill, substructure & cladding list
Pickle the combined list for CPython to Python
Create workable GH_Python list
pickle.load the combined list
seperate the combined list in 3 list:
Strucutral infill, substructure & cladding
For all elements in structural infill
append piece of wood to structural class
sort structural class on length wood w/ operator.attregeter
For all elements in substructure
append piece of wood to substructure class
sort substructure class on length wood w/ operator.attregeter
sort substructure class on depth wood w/ operator.attregeter
For all elements in cladding
append piece of wood to cladding class
sort substructure class on length wood w/ operator.attregeter
sort substructure class on height wood w/ operator.attregeter

Figure 7.4 Flowchart extract data from database (own ill)

The second script is written within the Python component. With pickle the database lists can be processed. Every piece of wood gets assigned to a class and then sorted on a specific attribute that is desirable for that class.

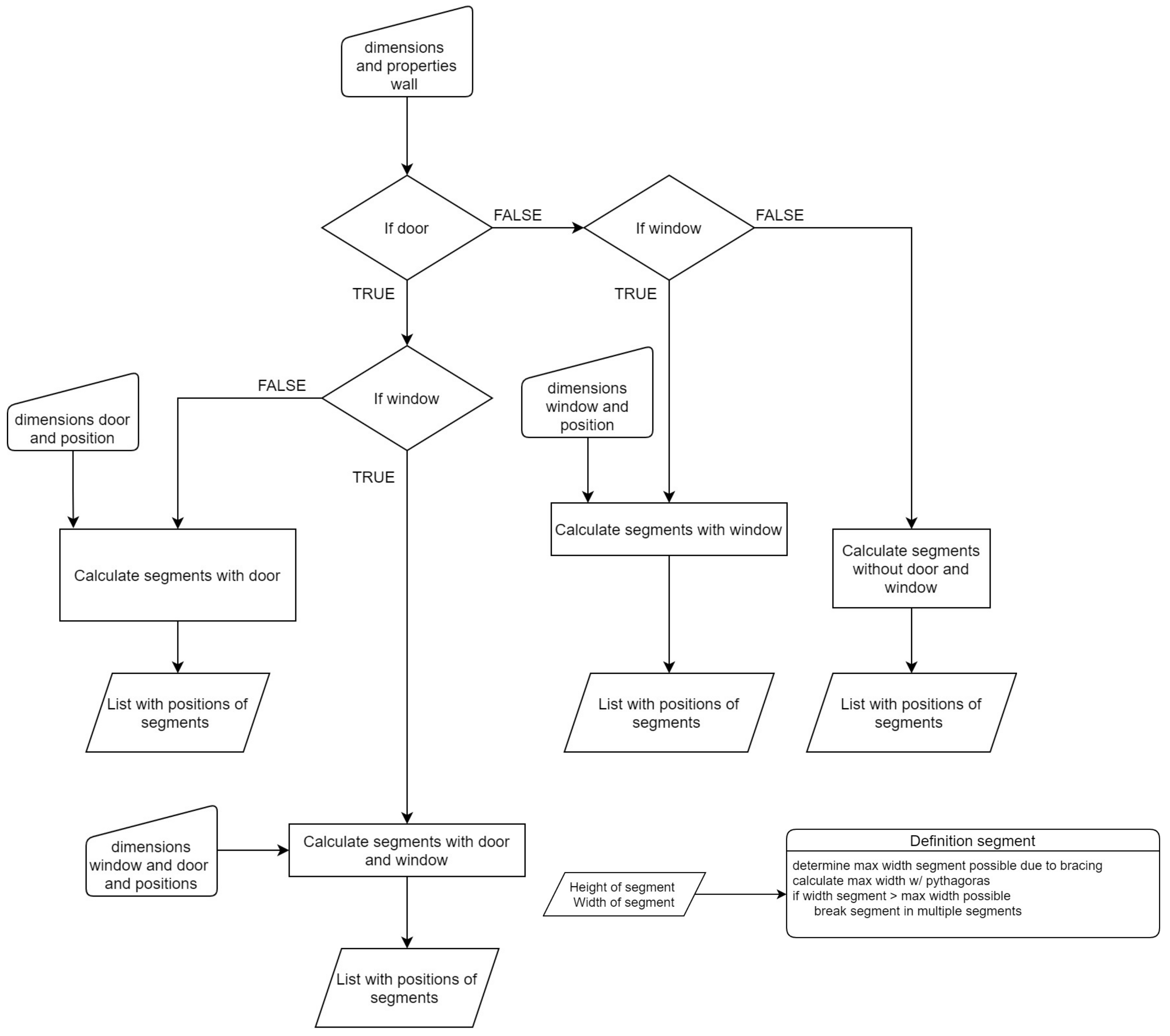
7.3.3 SEGMENTS

The script is added to the appendix in E.8.

Based on the dimensions of the wall and the choice for a door and/or window the wall element is split into segments.

The max-width of the segment is defined by the longest piece of wood available for bracing. The restriction is that a bracing element cannot consist of two pieces of wood. The max-width can be calculated with the Pythagoras formula, which can be seen in line 19 of the script. The longest piece of wood in the database gets used for this calculation.

A while loop allows for the process to repeat until the width of the segment left is smaller than the maximum width. The top row shows the divisions into segments for the structural infill and substructure. If the segments still exceed the max width, the segment is split into two segments. This can be seen with the orange dashed line.
The bottom row shows the segment division for the cladding.



dimensions and properties wall
If door
FALSE
If window
FALSE
TRUE
TRUE
If window
FALSE
TRUE
dimensions door and position
dimensions window and position
Calculate segments with door
Calculate segments with window
Calculate segments without door and window
List with positions of segments
List with positions of segments
List with positions of segments
dimensions window and door and positions
Calculate segments with door and window
List with positions of segments
Height of segment
Width of segment
Definition segment
determine max width segment possible due to bracing
calculate max width w/ pythagoras
if width segment > max width possible
break segment in multiple segments

Figure 7.5 Flowchart wall properties (own ill)

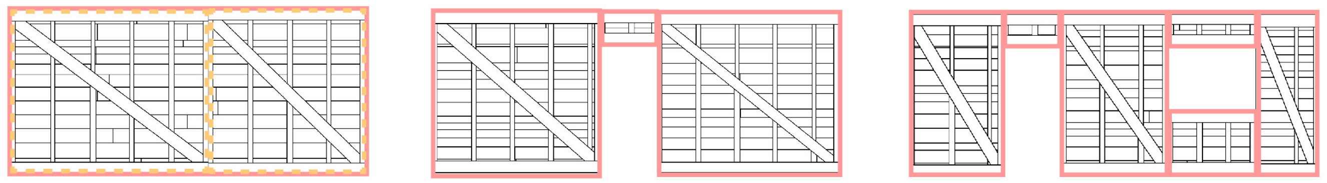


Figure 7.6 Segment divisions structural infill (own ill)

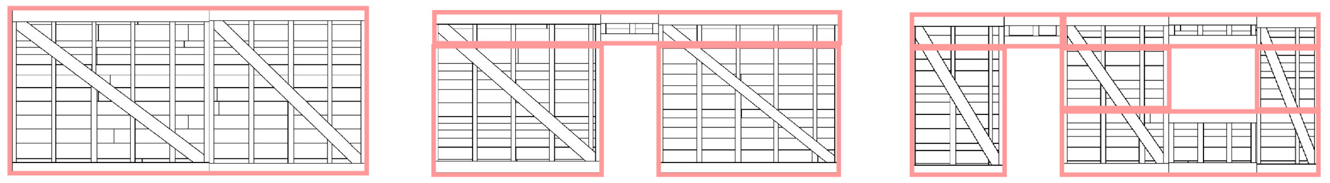
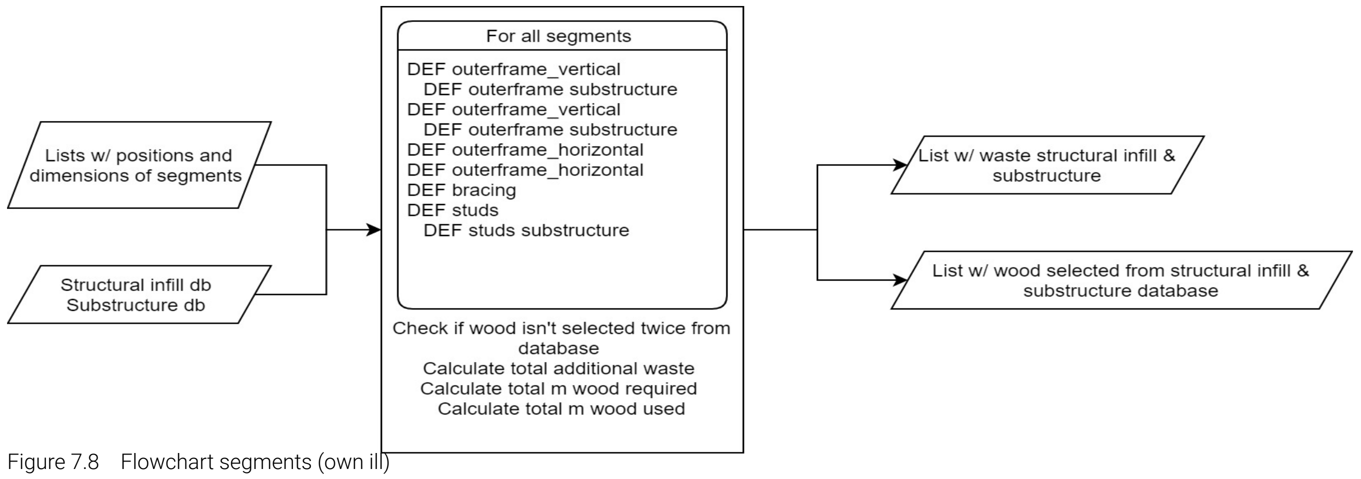


Figure 7.7 Segment divisions cladding (own ill)


Lists w/ positions and dimensions of segments
Structural infill db Substructure db
For all segments
DEF outerframe_vertical
DEF outerframe substructure
DEF outerframe_vertical
DEF outerframe substructure
DEF outerframe_horizontal
DEF outerframe_horizontal
DEF bracing
DEF studs
DEF studs substructure
Check if wood isn't selected twice from database
Calculate total additional waste
Calculate total m wood required
Calculate total m wood used
List w/ waste structural infill & substructure
List w/ wood selected from structural infill & substructure database
Figure 7.8 Flowchart segments (own ill)

7.3.4 STRUCTURAL INFILL & SUBSTRUCTURE

The script is added to the appendix in E.9.

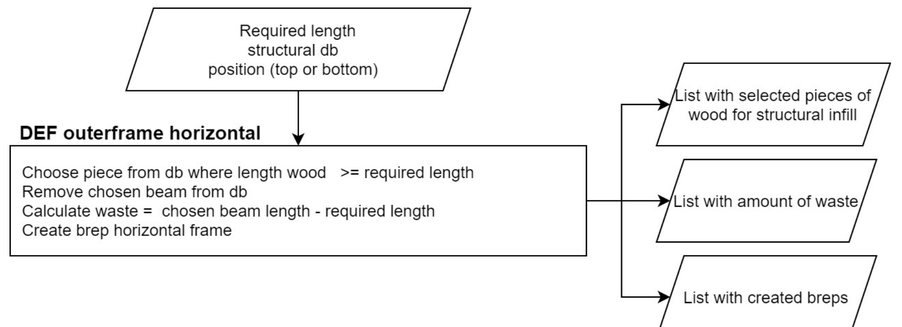
The flowchart below gives a general overview of how the scripts select wood for the structural infill and the substructure. The structural infill can be divided into three parts: the outer frame, the bracing, and the studs.

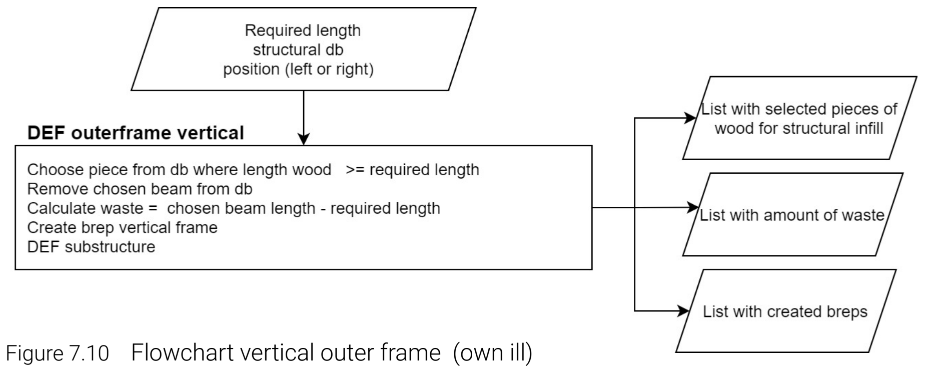
The function that creates the outer frame can be split into two parts: the vertical and the horizontal. There are two distinct differences. With the horizontal part, the required length is the segment's width, and with the vertical part, the required length is the height of the segment. The second distinction is that on the vertical outer frame substructure is added for the cladding. There is no substructure required on the horizontal frame.

Function outer frame

The list that contains the wood from the Structural infill database and the substructure database is already sorted ascending, based on the length of the wood. With the next function in Python, the wood piece can be selected that results in the least amount of waste.

The difference between the length of the selected wood and the required length is considered waste. During the prototype phase, the observation was made that these amounts of waste for every piece of wood were so small that it is not possible to reuse this small piece of additional waste


Required length
structural db
position (top or bottom)
DEF outerframe horizontal
Choose piece from db where length wood >= required length
Remove chosen beam from db
Calculate waste = chosen beam length - required length
Create brep horizontal frame
List with selected pieces of wood for structural infill
List with amount of waste
List with created breps
Figure 7.9 Flowchart horizontal outer frame (own ill)


Required length
structural db
position (left or right)
DEF outerframe vertical
Choose piece from db where length wood >= required length
Remove chosen beam from db
Calculate waste = chosen beam length - required length
Create brep vertical frame
DEF substructure
List with selected pieces of wood for structural infill
List with amount of waste
List with created breps
Figure 7.10 Flowchart vertical outer frame (own ill)

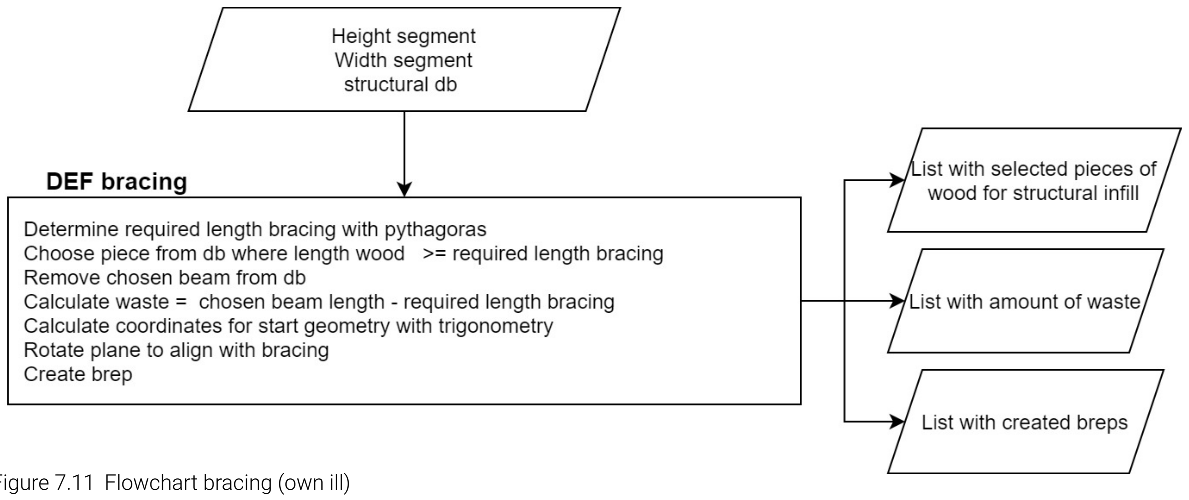
somewhere in the design again.

The selected piece of wood is appended to a list to be used again for different scripts.

Function bracing

The length of the bracing can be calculated with the Pythagoras formula. The selected wood for the outer frame is subtracted from the width and the height of the segment.

Generating the geometry requires some

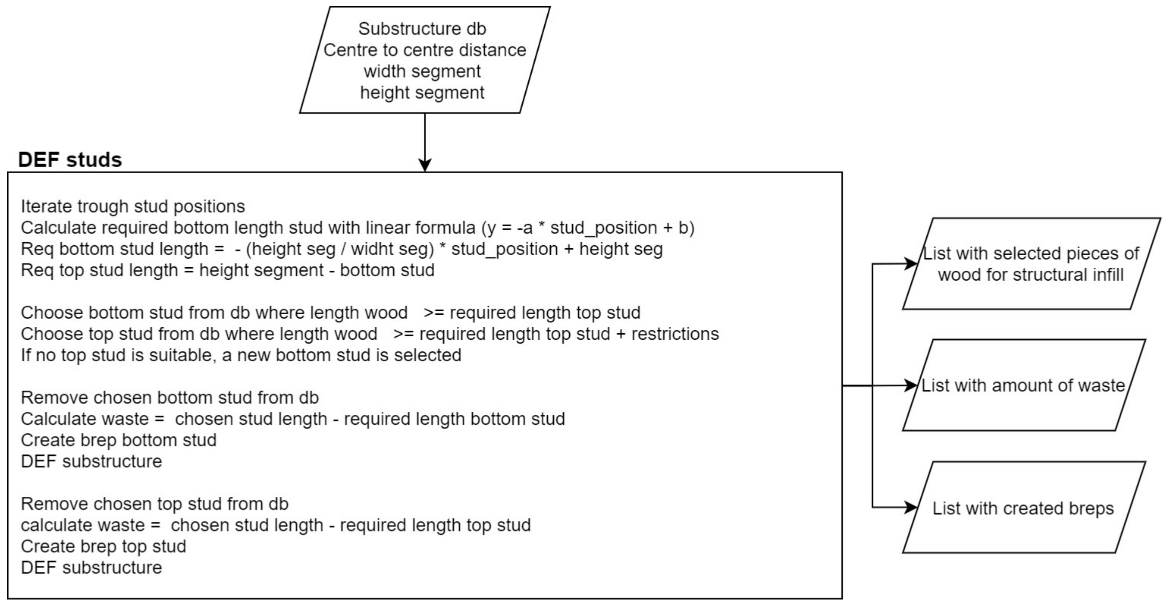

Height segment
Width segment
structural db
DEF bracing
Determine required length bracing with pythagoras
Choose piece from db where length wood >= required length bracing
Remove chosen beam from db
Calculate waste = chosen beam length - required length bracing
Calculate coordinates for start geometry with trigonometry
Rotate plane to align with bracing
Create brep
List with selected pieces of wood for structural infill
List with amount of waste
List with created breps
Figure 7.11 Flowchart bracing (own ill)

trigonometry to obtain the starting coordinates. When the starting coordinates are obtained, the z-axis of the construction planed need to be aligned with the bracing.

Function studs

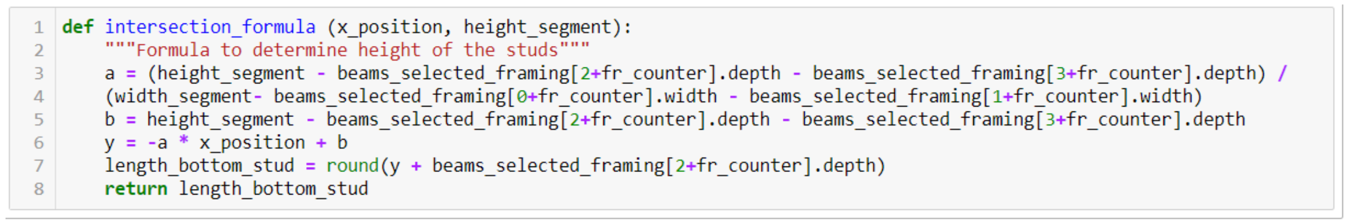
Within the outer frame, studs are placed in a fixed center to center distance, and this distance is adjustable within the script. The length of the studs is determined by the intersection point with the bracing.

To calculate this intersection point on any given position within the wall element, a formula is used:


Substructure db
Centre to centre distance
width segment
height segment
DEF studs
List with selected pieces of wood for structural infill
List with amount of waste
List with created breps

```
Iterate trough stud positions
Calculate required bottom length stud with linear formula (y = -a * stud_position + b)
Req bottom stud length =  - (height seg / widt seg) * stud_position + height seg
Req top stud length = height segment - bottom stud

Choose bottom stud from db where length wood   >= required length top stud
Choose top stud from db where length wood   >= required length top stud + restrictions
If no top stud is suitable, a new bottom stud is selected

Remove chosen bottom stud from db
Calculate waste =  chosen stud length - required length bottom stud
Create brep bottom stud
DEF substructure

Remove chosen top stud from db
calculate waste =  chosen stud length - required length top stud
Create brep top stud
DEF substructure
```

Figure 7.12 Flowchart studs (own ill)



```python
def intersection_formula (x_position, height_segment):
    """Formula to determine height of the studs"""
    a = (height_segment - beams_selected_framing[2+fr_counter].depth - beams_selected_framing[3+fr_counter].depth) /
    (width_segment- beams_selected_framing[0+fr_counter].width - beams_selected_framing[1+fr_counter].width)
    b = height_segment - beams_selected_framing[2+fr_counter].depth - beams_selected_framing[3+fr_counter].depth
    y = -a * x_position + b
    length_bottom_stud = round(y + beams_selected_framing[2+fr_counter].depth)
    return length_bottom_stud
```

Figure 7.13 Python code to determine intersection point (own ill)

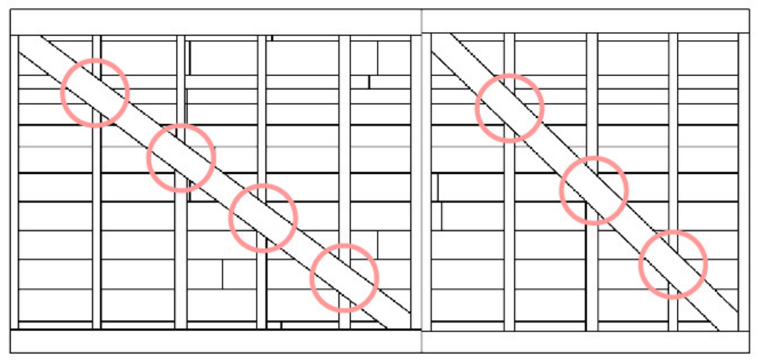


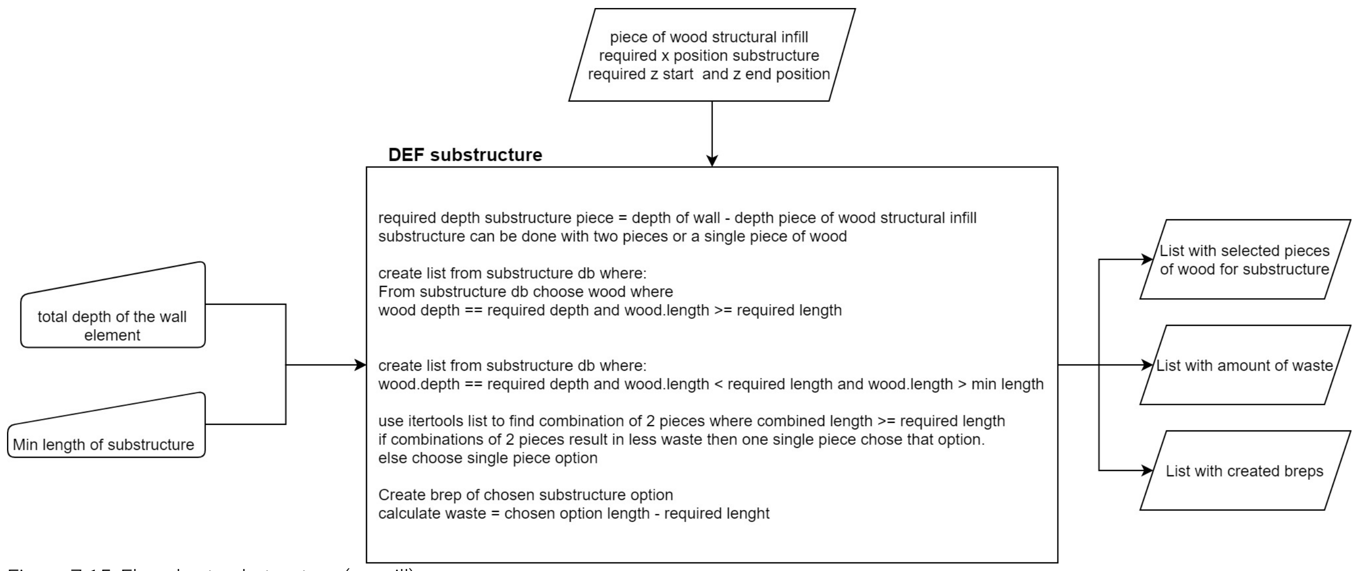
Figure 7.14 Intersection point between studs and bracing (own ill)

A while loop is used to generate studs until it reaches the end of the segment. In this while loop, additional restrictions can be given to the selection of the top stud.

Function substructure

The substructure is being attached to the structural infill to create a ventilated cavity. The cladding is also attached to the substructure, but it does not have to carry the weight of the cladding. The structural infill will do that. Therefore the substructure can be attached in multiple pieces. This allows for combining pieces of wood so that less waste is generated.

The substructure function will calculate how much waste is generated when the substructure is done with one piece and when it is done with two pieces.


piece of wood structural infill
required x position substructure
required z start and z end position
total depth of the wall element
Min length of substructure
DEF substructure
List with selected pieces of wood for substructure
List with amount of waste
List with created breps

```
required depth substructure piece = depth of wall - depth piece of wood structural infill
substructure can be done with two pieces or a single piece of wood

create list from substructure db where:
From substructure db choose wood where
wood depth == required depth and wood.length >= required length

create list from substructure db where:
wood.depth == required depth and wood.length < required length and wood.length > min length

use itertools list to find combination of 2 pieces where combined length >= required length
if combinations of 2 pieces result in less waste then one single piece chose that option.
else choose single piece option

Create brep of chosen substructure option
calculate waste = chosen option length - required lenght
```

Figure 7.15 Flowchart substructure (own ill)

The option with the least waste will be chosen.

7.3.5 CLADDING

The script is added to the appendix in E.10.

.
The cladding function calculates which combinations of pieces of wood with the same height can fill the entire width of the segment with the least waste.

The minimum length of a piece of cladding is twice the center-to-center distance. This guarantees that a cladding piece can always be mounted to at least two pieces of substructure. With this restriction, the maximum number of pieces for one row can also be calculated. For example, a wall with a width of 4.8 meters and a center-to-center distance of 600mm, can consist of maximum 4800 / (2 * 600) = 4 pieces. So, in this situation, a row can consist of 1,2,3 or 4 pieces of cladding. Each piece of wood in a single row must have the same height.

The script first searches for a combination of pieces with the same height where the combination of lengths results in the required length of the facade, this will result in zero waste. When all those options are examined, an additional marge is added. The marge represents the waste that is created.

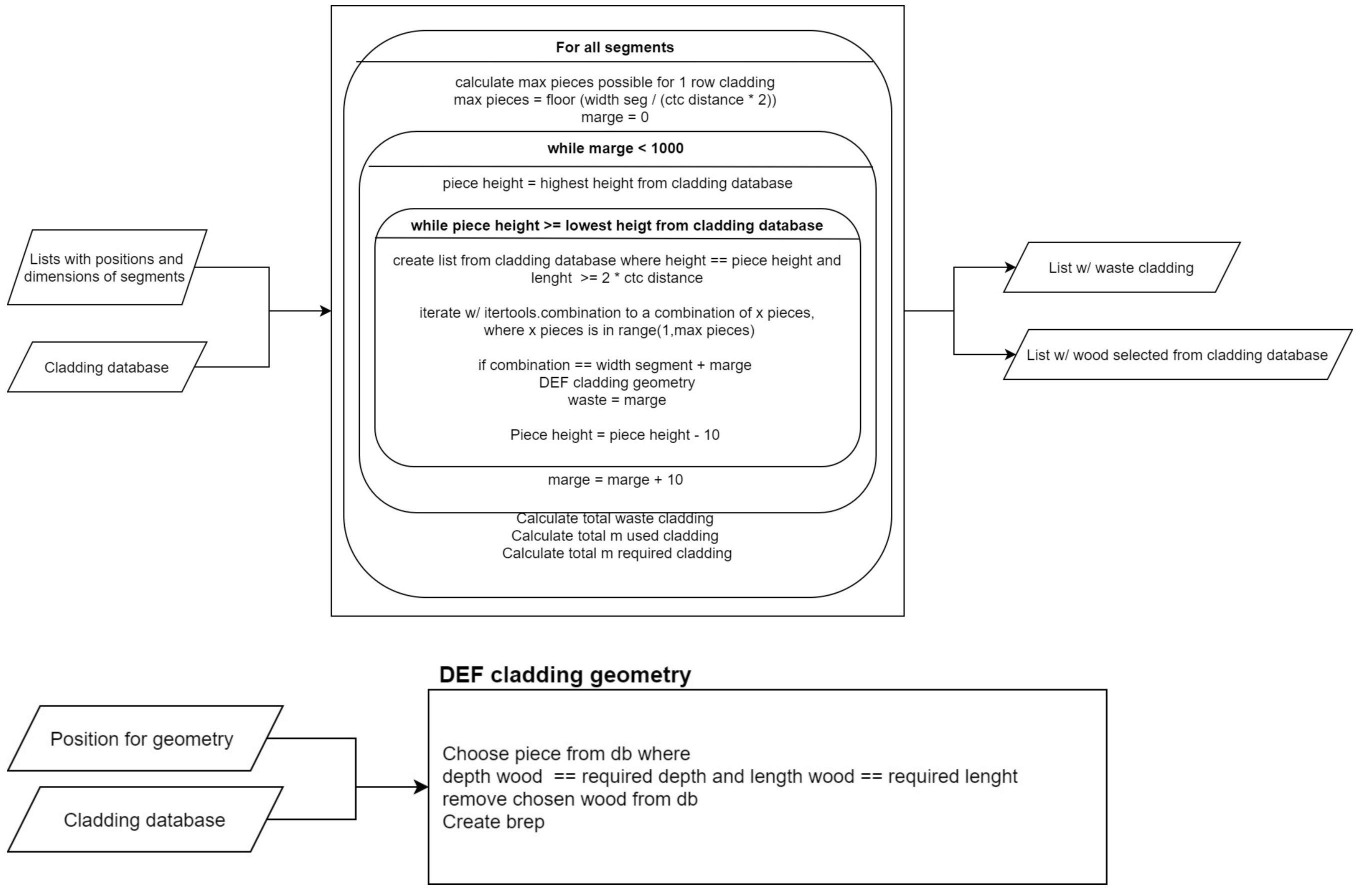

For all segments
calculate max pieces possible for 1 row cladding
max pieces = floor (width seg / (ctc distance * 2))
marge = 0
while marge < 1000
piece height = highest height from cladding database
while piece height >= lowest heigt from cladding database
create list from cladding database where height == piece height and lenght >= 2 * ctc distance
iterate w/ itertools.combination to a combination of x pieces, where x pieces is in range(1,max pieces)
if combination == width segment + marge
DEF cladding geometry
waste = marge
Piece height = piece height - 10
marge = marge + 10
Calculate total waste cladding
Calculate total m used cladding
Calculate total m required cladding
Lists with positions and dimensions of segments
Cladding database
List w/ waste cladding
List w/ wood selected from cladding database
DEF cladding geometry
Choose piece from db where
depth wood == required depth and length wood == required lenght
remove chosen wood from db
Create brep
Position for geometry
Cladding database

Figure 7.16 Flowchart cladding (own ill)

7.3.6 BOOLEAN DIFFERENCE AND BAKING

The script is added to the appendix in E.12 & E.13.

Boolean difference is performed when all the wood is selected, and the geometry is generated. With Boolean difference, the overlapping geometry gets removed. A final check is performed to remove any unwanted geometry that is created with the Boolean Difference. This can happen when the bracing is wider than the outer frame. The check removes all Breps that have a volume smaller than a specific value. The final Breps are baked in rhino.

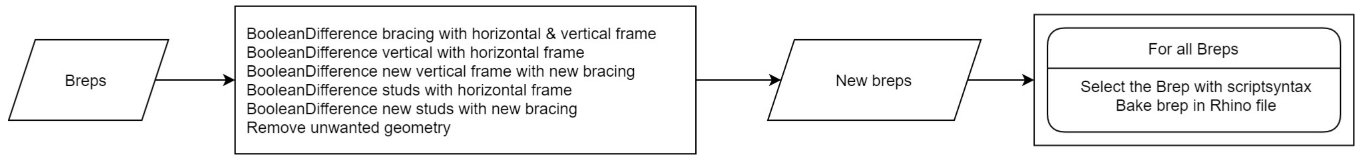

Breps
BooleanDifference bracing with horizontal & vertical frame
BooleanDifference vertical with horizontal frame
BooleanDifference new vertical frame with new bracing
BooleanDifference studs with horizontal frame
BooleanDifference new studs with new bracing
Remove unwanted geometry
New breps
For all Breps
Select the Brep with scriptsyntax
Bake brep in Rhino file

Figure 7.17 Flowchart boolean difference (own ill)

7.3.7 EXPORT 3DM FILES

The script is added to the appendix in E.14 & E.15.

The final baked geometry needs to be linked to the id numbers in the lists of selected wood. This way, each 3dm file can be named with the correlating id number. When the files are exported, the correlating id numbers are removed from the database.

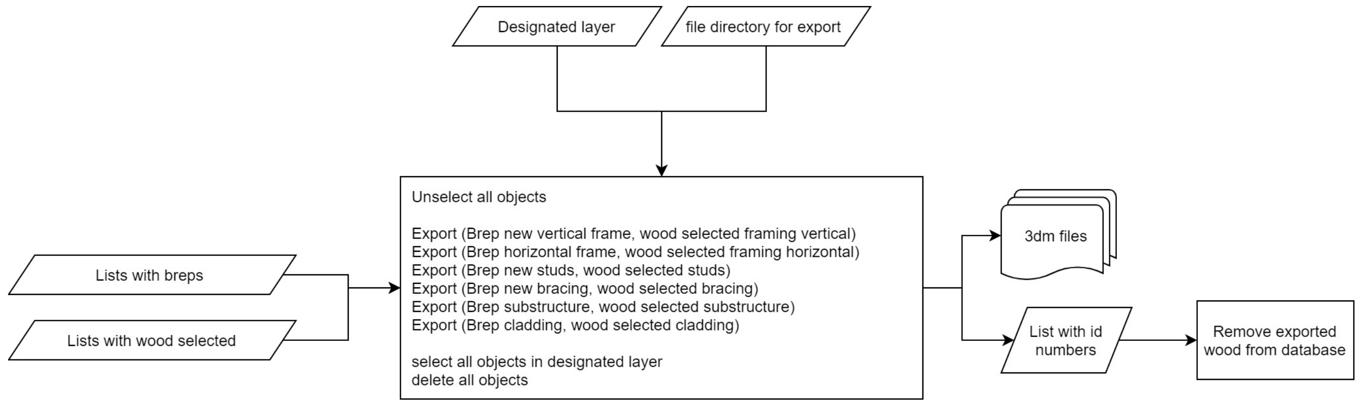
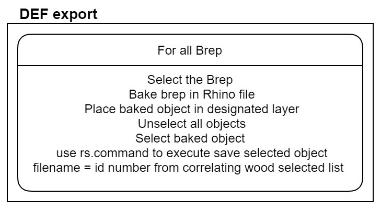

Designated layer
file directory for export
Unselect all objects
Export (Brep new vertical frame, wood selected framing vertical)
Export (Brep horizontal frame, wood selected framing horizontal)
Export (Brep new studs, wood selected studs)
Export (Brep new bracing, wood selected bracing)
Export (Brep substructure, wood selected substructure)
Export (Brep cladding, wood selected cladding)
select all objects in designated layer
delete all objects
Lists with breps
Lists with wood selected
3dm files
List with id numbers
Remove exported wood from database

Figure 7.18 Flowchart export 3dm files(own ill)


DEF export
For all Brep
Select the Brep
Bake brep in Rhino file
Place baked object in designated layer
Unselect all objects
Select baked object
use rs.command to execute save selected object
filename = id number from correlating wood selected list

7.4 DATABASE QUERY

All the wood suitable for the structural infill, substructure and cladding is extracted from the database and divided into three lists to allow for a simpler workflow. In reality, this would not be efficient when the size of the database keeps increasing. Instead of using the next function in Python to select the suitable piece of wood from the lists, a search query is required to obtain the required piece of wood directly from the database. The code in figure x shows how this would look. Figure x shows the original code where wood is selected for the horizontal framing. The code below shows how this would look with the search query.

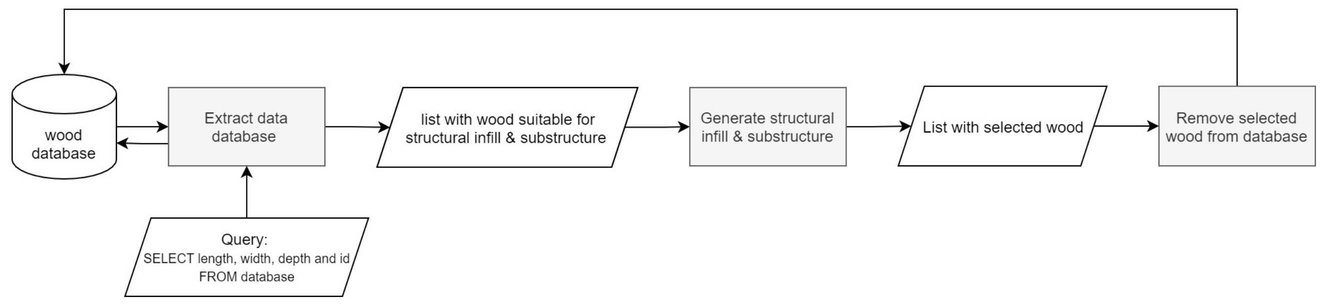

wood database
Extract data database
list with wood suitable for structural infill & substructure
Generate structural infill & substructure
List with selected wood
Remove selected wood from database
Query:
SELECT length, width, depth and id
FROM database

Figure 7.19 Working with lists (own ill)

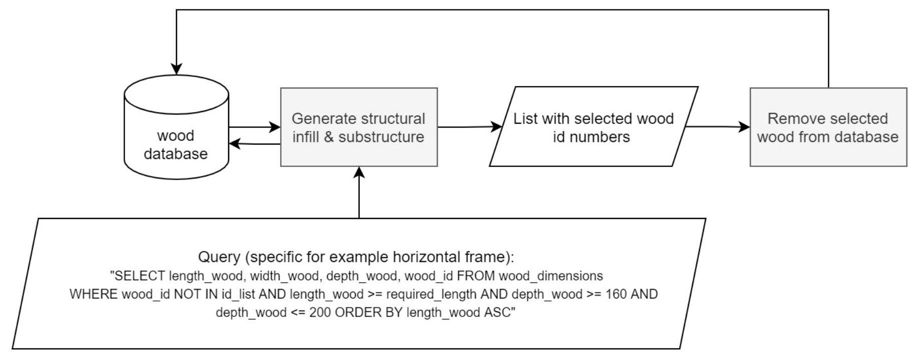

wood database
Generate structural infill & substructure
List with selected wood id numbers
Remove selected wood from database
Query (specific for example horizontal frame):
"SELECT length_wood, width_wood, depth_wood, wood_id FROM wood_dimensions
WHERE wood_id NOT IN id_list AND length_wood >= required_length AND depth_wood >= 160 AND
depth_wood <= 200 ORDER BY length_wood ASC"

Figure 7.20 Working with direct search query (own ill)

```python
class select_structural:
    def __init__(self, selected_wood, waste, geometry):
        self.selected_wood = selected_wood
        self.waste = waste
        self.geometry = geometry


def outer_frame_horizontal(required_length, waste_wood_database, position):
    """Selects wood from db and generates geometry for horizontal frame"""
    #pick piece of wood from database for horizontal outerframe (top & bottom)
    chosen_beam = next(beam for beam in waste_wood_database if beam.length >= required_length)

    #determine additional waste with picked piece of wood
    waste = chosen_beam.length - required_length

    if position == 'bottom':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_start)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam.width, seg_ver.z_cor_start + chosen_beam.depth)

    if position == 'top':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_end -chosen_beam.depth)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam.width, seg_ver.z_cor_end)

    # generate geometry in grasshopper
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    return select_structural(chosen_beam, waste, brep

picked_hor = outer_frame_horizontal(width_segment, waste_wood_database, 'bottom')

waste_wood_database.remove(picked_hor.selected_wood)
structural_infill_waste.append(picked_hor.waste)
beams_selected_framing_hor.append(picked_hor.selected_wood)
beams_selected_framing.append(picked_hor.selected_wood)
structural_infill_used.append(picked_hor.selected_wood.length)
geometry_infill_horizontal.append(picked_hor.geometry)
```

```python
import psycopg2 as pg2
id_list = []

class select_structural:
    def __init__(self, selected_wood, waste, geometry):
        self.selected_wood = selected_wood
        self.waste = waste
        self.geometry = geometry


def outer_frame_horizontal(required_length, waste_wood_database, position, id_list):
    """Selects wood from db and generates geometry for horizontal frame"""
    #pick piece of wood from database for horizontal outerframe (top & bottom)
    conn = pg2.connect(database='hout', user = 'postgres', password = 'baksteen')
    cur = conn.cursor()

    #search query
    query = "SELECT length_wood, width_wood, depth_wood, wood_id FROM wood_dimensions \
    WHERE wood_id NOT IN %s AND length_wood >= required_length AND depth_wood >= 160 AND \
    depth_wood <= 200 ORDER BY length_wood ASC"

    #the id list is to make sure the same piece of wood is not selected twice
    cur.execute(query, (tuple(id_list), required_length))
    chosen_beam = cur.fetchone()

    #determine additional waste with picked piece of wood
    waste = chosen_beam[0] - required_length

    if position == 'bottom':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_start)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam[1], seg_ver.z_cor_start + chosen_beam[2])

    if position == 'top':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_end -chosen_beam[2])
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam[1], seg_ver.z_cor_end)

    # generate geometry in grasshopper
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    return select_structural(chosen_beam, waste, brep)

picked_hor = outer_frame_horizontal(width_segment, waste_wood_database, 'bottom', id_list)

structural_infill_waste.append(picked_hor.waste)
beams_selected_framing_hor.append(picked_hor.selected_wood)
beams_selected_framing.append(picked_hor.selected_wood)
structural_infill_used.append(picked_hor.selected_wood[0])
geometry_infill_horizontal.append(picked_hor.geometry)
id_list.append(picked_hor.selected_wood[4])

"""If the user accepts the generated design the scripts removes all the selected wood from the databas
e"""
conn = pg2.connect(database='hout', user='postgres', password='baksteen')
cur = conn.cursor()
for id_number in id_list:
    delete_id(id_number)
```

8 CONCLUSION

Waste wood (Print waste)

8.1 SUB QUESTIONS

To conclude this research and answer the research question, the sub-questions must be answered first.

How can a digital database with waste wood properties communicate with a parametric model of an exterior wall element?

For the prototype, a SQL database is created with PostgreSQL. SQL can handle large amounts of data and still update in real-time, something that cannot be done with Excel. In this database, all the different properties and characteristics of the waste wood are stored. The database can filter the wood that is suitable for structural infill, cladding, or substructure. With a search query, the desired data can be extracted from the database. For the prototype, the connection to the database is made with the Psycopg2 module in Python.

The following columns are integrated into the database.

- Id number
- Length in mm
- Depth in mm
- Width in mm
- Strength class
- Weight in kg
- Type of wood
- Structural integrity
- Painted
- Density
- Volume
- Archived on
- Sustainability class
- Seller id

How can the script minimize the additional wood waste with the selection procedure when selecting the waste wood from the database for the exterior wall element?

The script extracts the data from the database and places it in different lists for the structural infill, cladding, and substructure. The lists with wood available are sorted on the desired attribute (length, height, etc.). This allows the script to select the wood that is closest to the required attribute. The cladding and substructure allow for a more successful selection procedure were waste is minimized due to the possibility of making combinations. The script iterates through more possible combinations and can therefore find better solutions. Within the script, itertools is used for this function. This method cannot be used for the structural infill. The length required for the structural infill is determined by the intersection between the studs and the bracing. Using more pieces for the studs will decrease the structural stability. Because there are no structural calculations conducted for this thesis, it cannot be proven that the studs can still carry the load of the exterior wall element when it consists of more than two pieces.

Which parts of the exterior wall element can be constructed with waste wood?

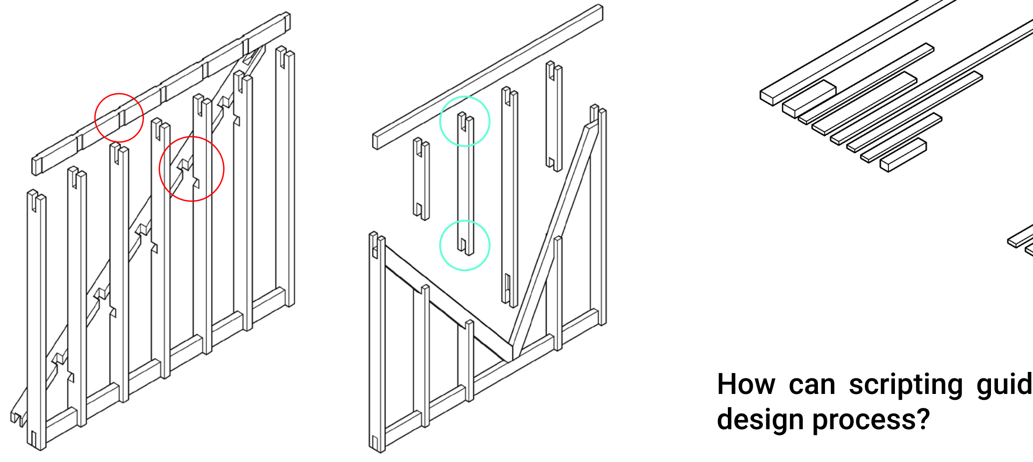
In the Netherlands, there are requirements for the construction of buildings. These requirements are written in the Bouwbesluit. For the developed exterior wall element to be built in the Netherlands, it is necessary to meet these requirements. Waste wood is wood that is already used. Therefore, it can have imperfections. These imperfections can make it more complicated for the construction to meet the requirements. For example, a closed facade system is not recommended with waste wood because the wood can be crooked, and therefore make it difficult for the facade to be watertight. Using a foil to act as a water barrier will solve this problem.
Within this research, the decision was made to fulfill the relevant building code demands with external and already proven materials, such as a water barrier foil.

As a conclusion of the literature phase, the exterior wall can be divided into three segments: the interior part, the structural middle part, and the exterior part. It is preferable to do the interior part with other materials than waste wood. For the interior part it is desirable to have a surface that allow for easy finishing. This is difficult to realize with waste wood. It became clear that it is a better idea to choose a material with additional sound insulating and fire-resistant properties. If the user wants an interior consisting of waste wood, which was the case for the project at GPGroot/Sortiva, it is still recommended to use a different material between the structural middle part and the waste wood interior.
The exterior cladding, the structural infill, and the substructure can be constructed with waste wood. However, the structural infill is still debatable because calculations do not yet confirm this. After developing the prototype, it became clear that the most waste wood used, was for the cladding and the substructure. These two combined make up for 74% of the wood used while dealing with the least restrictions from the building codes.

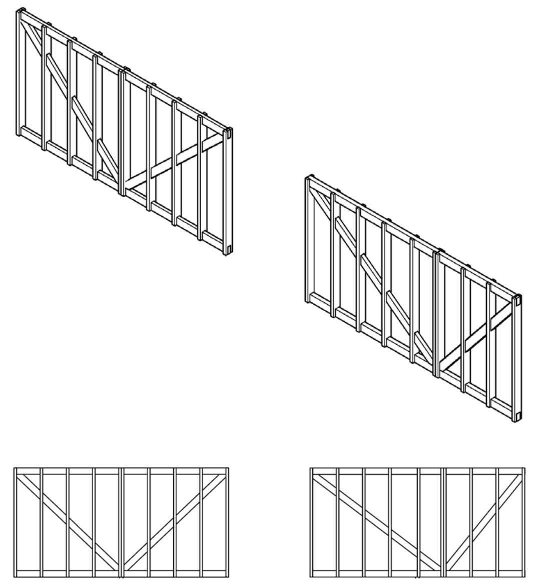
What design principles allow the construction of the exterior wall element to be done with waste wood without unnecessary cuts and minimizing the material waste?

The design principles are a result of following the design methodology. Trough research by design & scripting some principles were obtained in the design process that were later converged into criteria. With the new criteria and design principles the design methodology was repeated. The design principles are:

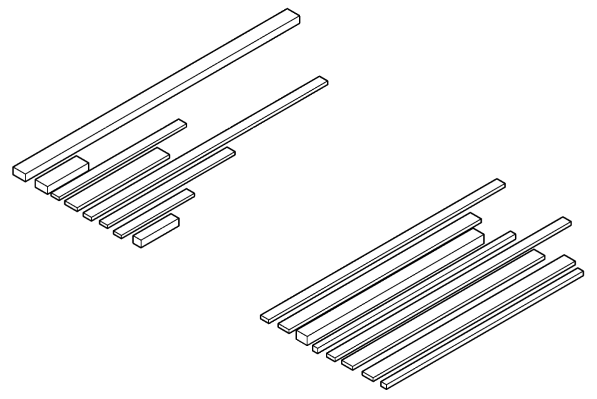
Modifications on the pieces of wood within the chosen design should only be on the outside. This allows for the wall element to maximize the reusable pieces of wood after its first use as a wall element. The modified parts can be removed, and the remaining piece of wood can be reused again in a new exterior wall element.



Symmetry of the design should be prevented. A symmetric design can result in additional waste because the same dimensions of the wood have to be selected twice. The chance increases that the wood is not available in that exact dimensions twice, and therefore, a larger piece of wood has to be cut to fulfill the required dimensions.



A broad range of dimensions should be used in the design. This way, the chance increases in selecting the pieces of waste wood that fits closest to the required dimensions. Having only small pieces required for the design can result in splitting larger pieces of waste wood into smaller pieces. This will result in a decrease in value. Having a design with only large pieces of wood will result in not having a function for the small pieces of wood.



How can scripting guide the designer with the design process?

Within this thesis, scripting was added to an existing design methodology to see how and if scripting can help guide the designer in finding the most suitable design. The conclusion is that scripting can guide the designer in the design process. However, the designer should consider if it is worth implementing it into the early stages of the design process. During the early stages of the design process, it became clear that the script did not add that much value in comparison to the time invested. The parameters are still too vague and the design possibilities to broad.

The benefit of scripting increases when the number of solutions to a certain design problem is limited and the desired output is preferably quantified.

Writing scripts can help the designer with confirming a certain suspicion. It can also generate new insights and see how certain aspects of the design relate to each other. The more specified the design already is, the better scripting can confirm a certain expectation.

The script can act as a playground, where experiments and small studies can happen within a specific base model. This playground has the most benefit when a design proposal is already drafted.

These experiments about the impact of specific restrictions can be programmed rather quickly within the general script. Different restrictions can also be combined to study how they relate to each other and what the combined effect is on the additional waste.

8.2 MAIN RESEARCH QUESTION

How can a database, a parametric model, and scripting be used to develop an exterior wall element from waste wood that minimizes the material loss and takes full benefit of the wood dimensions?

8.2.1 ANSWERING THE MAIN RESEARCH QUESTION

In 2017, 1610 million kg of waste wood was collected in the Netherlands. The building and demolition industry was, and still is, the most significant contributor. The estimation is made that in 2017 23% of this wood, 370 million kg, consisted of solid B waste wood. This solid B waste wood has the potential to be reused as a building material without having to shred it. To this day, with the current waste wood processing methods, this high potential waste wood ends being shredded for the engineered board industry or incinerated for bioenergy. The problem with engineered boards is that they cannot be recycled again into an engineered board due to the adhesives. They can only be incinerated for bio energy. If all the 370 million kg of solid B waste wood is downcycled into engineered boards instead of reused, an additional 399.900.000 kg of CO2 is emitted.

Reusing waste wood again as a building material instead of downcycling it into an engineered board is considered circular. Therefore, it is in line with the Dutch government goal to create a circular economy by 2050. However, the current commonly used timber frame construction methods in the Netherlands are not suitable for construction with waste wood. Waste wood has certain characteristics that differ from regular wood for construction on a larger scale. Right now, it is possible to create an exterior wall element from waste wood, but this is a time-consuming manual process and not suitable for production on a larger scale. Because manual labor is expensive in the Netherlands. So, a carpenter that needs to modify every piece of wood manually to get everything in the correct dimensions will probably be more expensive than buying new wood.

Combining a waste wood database in SQL and a parametric model can offer a solution to the challenges of constructing an exterior wall element with waste wood. The benefit of a parametric model is that it can fulfill the user's preference in dimensions and properties. Combining this with a database and the computer can link the preference of the user together with the available waste wood without a human having to find the most suitable pieces of waste wood manually. The database can register all the characteristics and availability of the waste wood. This data can be used to categorize the wood so that is suitable for certain functions within the design. For example, a piece of wood with a crack cannot be used for a load-bearing function but can be used in the cladding. This way the piece of waste wood can still find a purpose in the exterior wall element instead of being thrown away. Using python scripts, the most suitable pieces of wood can be selected from the database, thereby minimizing any additional material loss. With scripting, all the pieces of wood can be modified digitally and exported as separate 3dm files that can be used for digital fabrication.

To design an exterior wall element that can be made parametric and work with the exterior wall element tool, a design methodology was followed. The design methodology was used to identify the design problems, determine the criteria, and guide the designer in finding the most suitable design that can work with the challenges of waste wood. The chosen methodology was altered by implementing scripting to see if it could improve the methodology. Scripting allowed the designer to adjust different parameters for the design solutions, and therefore the designer could study how these parameters relate to certain criteria and to each other. This direct feedback allowed the designer to improve the different design solutions to a design problem. However, scripting showed to be more effective in the later stages of the design process because the number of solutions to a certain design problem is limited, and the desired output is preferably quantified. In the earlier phase of the methodology, the design possibilities are too broad, and this required to much time invested in writing scripts. The time it takes to write these scripts does not weigh up against the additional obtained knowledge.

In later stages of the design process, the script can act as a playground, where experiments and small studies can happen within a specific base model. This playground has the most benefit when a design proposal is already drafted. These experiments about the impact of specific restrictions can be programmed rather quickly within the general script. Different restrictions can also be combined to study how they relate to each other and what the combined effect is on the additional waste.

By following the methodology, new criteria emerged that resulted in certain design principles that allowed the design objectives of this thesis, having a design that can be constructed as much as possible with waste wood, minimize any additional material waste and be in line with the circular economy, to be fulfilled. The developed design principles are:

- Modifications on the pieces of wood should only be on the outside.
- Symmetry in the design should be prevented.
- A broad range of dimensions should be used in the design.

The final design was made parametric so that it could communicate with the database. Using Rhino, Grasshopper and Python, a prototype of the exterior wall element tool was made. These components bridged the parametric model with a waste wood database and gave the user some insight into the design's waste wood statistics. The user could see how certain design decisions influenced the waste, material use and how much could be reused after its end of life. This could be narrowed down to the three elements of cladding, structural infill, and substructure. Python scripts were used for an efficient selection procedure whereby additional material waste could be minimized.

The final design combined with the exterior wall element tool can calculate how many wall elements can be constructed from waste wood. The exterior wall element tool with the final design calculates that a wall element with a height of 3 meters, a width of 4,8 meters and a center-to-center distance of 600 mm weighs, on average, 503 kg. For a 100m2 single floor square house with a height of 3 meters, a 120m2 wall is required. 120m2 exterior wall is 4.192 kg. If 10% of the solid B-wood can be reused, 8.826 houses can be constructed every year. This amount can have a significant contribution to the one million additional houses that are required by 2030.

9 RECOMMENDATION

Spatial timber assembly (ETH Zurich)

9.1 RECOMMENDATIONS

With the increasing interest in a circular economy, the reuse of waste wood is very relevant. This thesis suggests a design of an exterior wall element that can work with waste wood, and a tool is developed that allows for this wall element to be implemented in the Dutch market. Reflecting on the research process, some recommendations can be made.

For the design, the goal was to see what parts of the exterior wall could be made from waste wood. In the Netherlands, the construction of buildings must meet the requirements of the building code. Because of the varying quality of the waste wood, it is nearly impossible for the wall element to meet these requirements only with waste wood. Therefore, this resulted in additional materials added to the wall element.

9.1.1 STRUCTURAL CALCULATIONS

The structural part of the wall element must also meet certain safety requirements to prevent the wall from collapsing. For this thesis, there are no structural calculations conducted due to the graduation project's limited time frame. For the design to be realistic, reference projects were used to determine the studs' minimum size. It would be interesting to calculate the load bearing capacity of waste wood in combination with the varying center to center distance.

Research in this thesis showed that having a variable center-to-center distance can reduce the additional waste created with the wall element. This principle can be combined with the structural calculations. In this situation the distance between the studs is determined by the load bearing capacity of the chosen stud. A combination of studs can be selected that optimize the structural capacity while minimizing the waste. The structural capacity of a single stud can be determined by the cross section and strength class. A larger cross section results in a higher load bearing capacity. The variable center-to-center distance over the width of the wall element would result in an increased assembly complexity. This could be solved by integrating a robotic assembly.

Combining this robotic assembly with structural load calculations can result in interesting follow-up research. Having structural calculations would also allow for a more generative design performed by the script.

9.1.2 FACADE DESIGN

Looking at the bigger picture. The tool calculated that, on average, only 26% of the wall element consist of the structural infill, were the remaining 74% is the substructure and cladding. Using an open facade and a water resistance foil, the substructure and cladding do not have to meet that many requirements as the structural infill. Therefore, it is less complicated to implement this realistically.

New research can be done by focusing on the connection between the cladding and the substructure and allow for a new facade system. Here the script can create a design that is not restricted by structural calculations. Without these restrictions the selection procedure were the scripts selects the waste wood to be used for the facade can also be more efficient and minimizing waste.

During the location visit at GP Groot/Sortiva it became clear that there is already a demand for such a facade system. The facade can act as an interesting marketing tool.

It would be interesting to see if this facade system could be assembled with robots. In chapter 3, the suggestion was made to use robotic sorting to filter the reusable solid b wood from the wood stream. If a wood processor would use this, the next step can be relatively easy to implement robots in the assembly process.

9.1.3 ENGINEERED BOARDS

To conclude, bracing was used to make the wall element stiff and stable. The decision was made to use bracing instead of engineered boards because engineered boards cannot be recycled again. The production of engineered boards has a negative impact on the climate. However, the use of bracing increased the complexity of the design. In the future, it could be possible for the engineered boards to be manufactured with more biobased advises. This allows for the engineered board to be recycled into an engineered board again. Using engineered boards allows for more room for the insulation and easier placement. It would also allow for a more flexible design with the studs because the intersection point is not restricted.

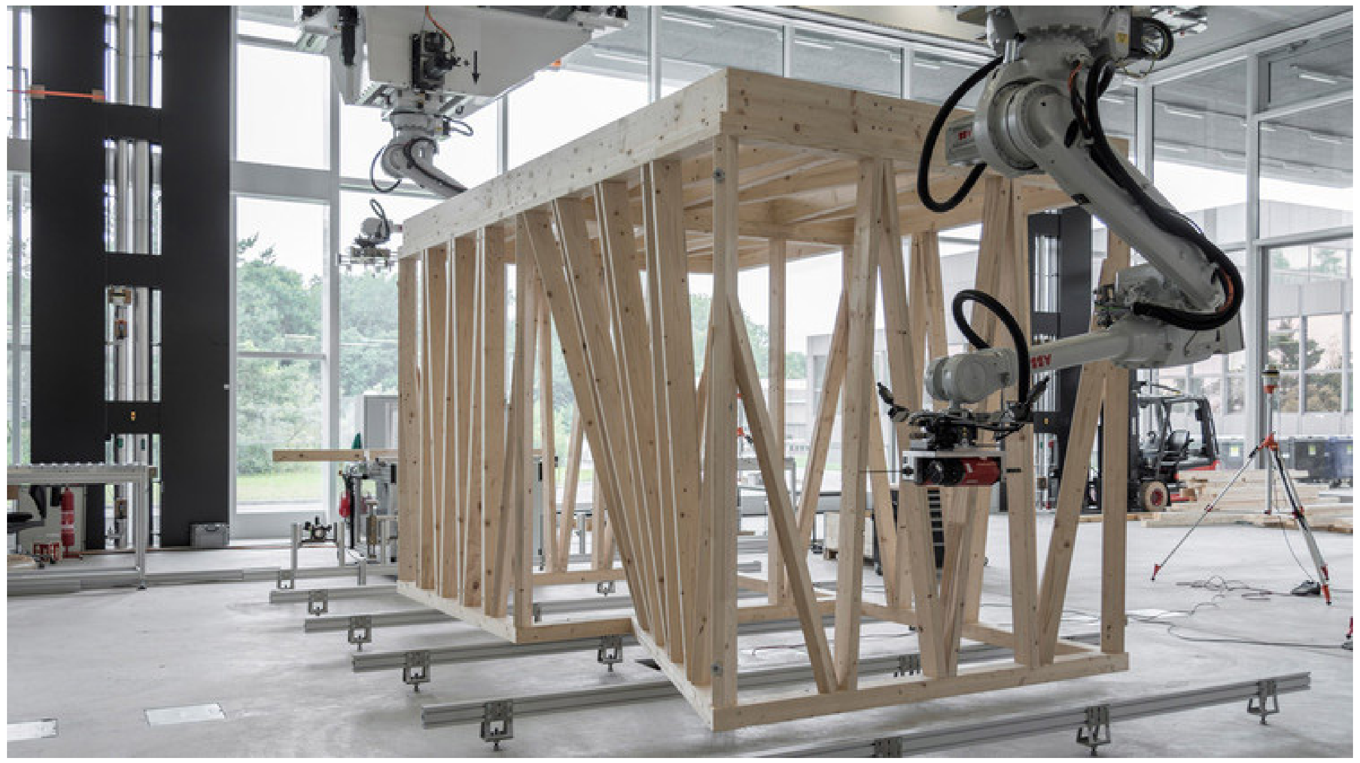


Figure 9.1 Spacial timber assembly (ETH Zurich)

10 BIBLIOGRAPHY

Piled Up Waste Wood Man (Piet Hein Eek)

ABF Research. (2018). 1 miljoen woningen. Retrieved February 15, 2021 from https://www.abfresearch.nl/nieuws/1-miljoen-woningen/

Afman, M. A., Bergsma, G. C., Bijleveld, M. M., Krutwagen, B. T. J. M. (2014). Milieu-impacts van Nederlandse bouw- en sloopactiviteiten in 2010 (Publicatienummer: 14.2746.25). Delft, The Netherlands: Bouwend Nederland

BBC. (2018). Climate change: The massive CO2 emitter you may not know about. Retrieved June 5, 2020 from https://www.bbc.com/news/science-environment-46455844

Briel, J., Oldenburger, J. (2009). Houtsoorten voor de Woningbouw, Utiliteitsbouw en Grond-, Weg- en Waterbouw. Wageningen, Stichting Probos. Retrieved houtdatabase website: https://www.houtdatabase.nl/infobladen/infoblad_houtsoortenkeuze_versus_toepassing.pdf

Brikawood. (2012). Construire avec Brikawood. Retrieved February 15, 2021 from https://www.brikawood-ecologie.fr/concept-kit-maison-passive-bois/construction-maison-bois-passive

Bruggen, R., Zwaag, N. (2017). Knelpuntenanalyse houtrecycling (Kenmerk R001-1250953RPB-hgm-V05-NL). Deventer, The Netherlands: Tauw BV.

CBS. (2019). Milieuvoetafdruk van Nederlander licht toegenomen. Retrieved March 23, 2021 from https://www.cbs.nl/nl-nl/nieuws/2019/20/milieuvoetafdruk-van-nederlander-licht-toegenomen

Centrum Hout. (2005). Houtwijzer, naaldhout in de bouw. Almere, The Netherlands.

Cross, N. (1984). Developments in design methodology. New York, USA: Wiley

Death, D., Mann, R., Picnic, D., Reilly, M., Taylor, J., Warnken, M. (year). Recycling and End-of-life Disposal of Timber Products (Project no: PN05.1017). Victoria, Australia: Forest & Wood Products Research & Development Corporation.

Duivenvoorde, G., Elburg, G., & Krebbekx, J. (2016). Detaillering roadmap recycling hout. Utrecht, The Netherlands: Berenschot.

Ecohousemart. (2020). Modern log house construction. Retrieved February 15, 2021 from https://ecohousemart.com/technology/construction-process/

Eekels, J. (2003). Productontwerpen, structuur en methoden (2e dr. ed.). Utrecht: Lemma.

Gijsbers, R. (2011). Aanpasbaarheid van de draagstructuur. TU/e, Eindhoven.

Hansen, H. J. (1971). Architecture in Wood; A History of Wood Building & It's Techniques in Europe and North America. Hamburg, Germany: Gerhard Stalling.

Hebel, D. E., Wisniewska, M. H., & Heisel, F. (2014). Building from waste, recovered materials in architecture and construction. Basel, Switzerland: Birkhäuser.

Houtinfo. (2013). Hygroscopische eigenschappen. Retrieved February 15, 2021, from https://www.houtinfo.nl/sites/default/files/Hout_Hygroscopische_eigenschappen_dec2013_0.pdf

Houtinfo. (2014). Infoblad houteigenschappen. Retrieved May 23, 2020, from https://www.houtinfo.nl/sites/default/files/Infoblad_Houteigenschappen_Sterktegegevens_mrt2014_0.pdf

Hugues, T., Steiger, L., Weber, J. (2014). Timber Construction; Details, Products & Case studies. Basel, Switzerland: Birkhauser.

JoostdeVree. (2012). balloon-framing, balloon-frame, balloon-methode, balloon-constructie. Retrieved February 15, 2021 from https://www.joostdevree.nl/shtmls/balloon-framing.shtml

Masson-Delmotte, V., P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J.B.R. Matthews, Y. Chen, X. Zhou, M.I. Gomis, E. Lonnoy, T. Maycock, M. Tignor, and T. Waterfield (2018). Global Warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty. IPCC

Milieu centraal. (2020). Wat is je CO2-voetafdruk? Retrieved March 23, 2021 from https://www.milieucentraal.nl/klimaat-en-aarde/klimaatverandering/wat-is-je-co2-voetafdruk/

Nieman, H. M. (2012). SBR-referentiedetails woningbouw, studenteneditie 2012. Amersfoort, The Netherlands: ThiemeMeulenhoff.

Pugh, S. (1981). Concept selection: a method that works. Proceedings international conference on engineering design, March 1981, Rome. Zürich: Heurista, 1981, pp. 497 – 506.

Recylcling today. (2019). How robots are revolutionizing recycling. Retrieved June 7, 2020 from https://www.recyclingtoday.com/article/recycling-robots-ai-sorting/

Rijksoverheid. (2021). Bouwbesluit 2012. Retrieved February 15, 2021 from https://rijksoverheid.bouwbesluit.com/Inhoud/docs/wet/bb2012

Rijksoverheid. (2018). Transitieagenda Bouw. Retrieved June 5, 2020 from https://www.rijksoverheid.nl/documenten/rapporten/2018/01/15/bijlage-4-transitieagenda-bouw

United Nations. (2020). Forest Products Annual Market Review 2019-2020. New York, USA: United Nations Publications.

Van Veen, J. (2016). PD_LAB a file-to-factory envelope. TU Delft, Delft University of Technology.

Van der Knaap, N. (2016). PO_LAB: InDetail. TU Delft, Delft University of Technology.

Woodcampus. (2017). Strength grading. Retrieved February 19, 2021, from: https://www.woodcampus.co.uk/builder-and-trade_strength-grading/

World Green Building Council. (2019). New report: the building and construction sector can reach net zero carbon emissions by 2050. Retrieved June 5, 2020, from https://www.worldgbc.org/news-media/WorldGBC-embodied-carbon-report-published

Zweger, K. (2011). Wood and Wood Joints: Building Traditions of Europe, Japan and China. Basel, Switzerland: Birkhauser.

11 APPENDIX

Waste wood (Lidner Recyclingtech)

A CO2 CALCULATION

In 2017, there was 370.000.000 kg of waste wood that had the potential to be reused as a building material. This number is an estimation by TAUW published in 2017 and not officially registered data. There is only official data available about the amount of B-wood. The estimated number from TAUW is based on different interviews with stakeholders. The number is based on the mixture of solid B-wood and A wood that wood processors make and shred for the recycling industry. The actual amount of solid B wood may be higher because solid B wood can end up on a pile with non-solid B wood that is shredded for the bio energy industry. No laws are preventing this from happening. The percentage of solid B wood that is suitable for reuse is unknown. For the calculation, the assumption is made that all the 370.000.000 kg is downcycled and not being incinerated for bioenergy.

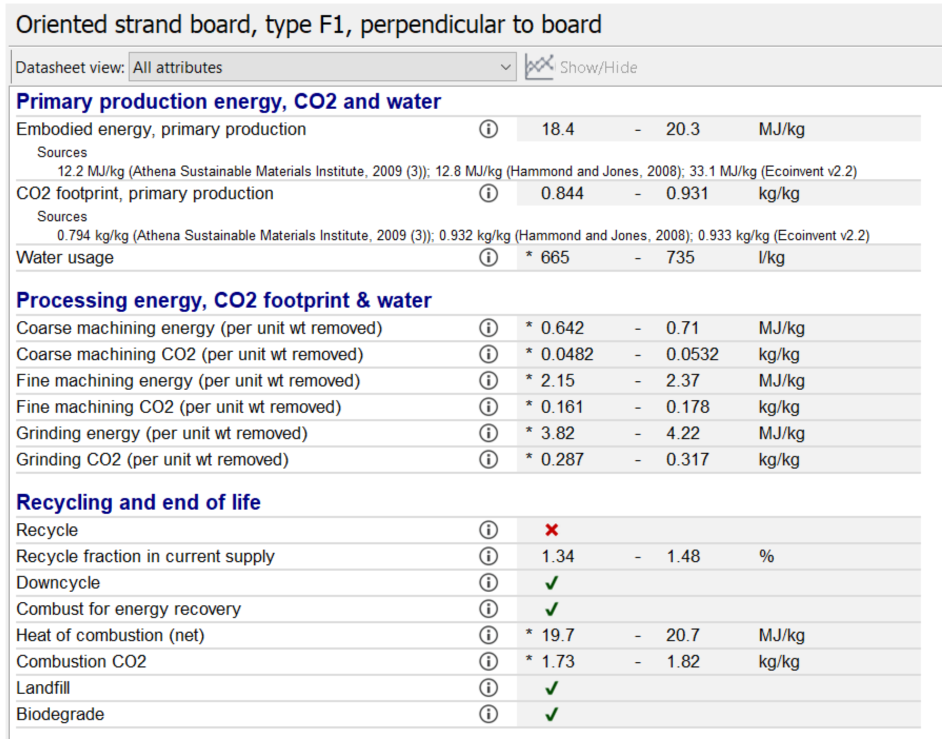
CES EduPack states that the production of 1 kg oriented stranded board is responsible for 0,91 kg of CO2. With the assumption that 1kg of OSB requires 1kg of waste wood, the 370.000.000 kg of waste wood result in 337.600.000 kg of CO2.

The Belgium factory for oriented stranded boards is in Wielsbeke. Renewi has it waste wood process facility in Nieuwegein. The driving distance between these two places is 210 kilometre. With the TLNplanner, the CO2 emissions for the transportation of the wood chips can be calculated. A 5-axle truck, capable of loading 25.000 kg of woodchips, generates 170 kg CO2 during the 208 kilometres drive. This results in an additional 2.300.000 kg of CO2 for all the solid B waste wood. This is just 0,7% of the CO2 emissions that are produced when manufacturing the OSB.

The total emission with transportation is 339.900.000 kg of CO2.

Milieu centraal calculated that the energy consumption of an average household (2,2 persons) in the Netherlands is responsible for 3.800 kg of CO2 every year. If 10% of the estimated solid B wood can be reused, this would save 33.990.000 kg of CO2. This is equal to the energy consumption of 8.950 households, the same as a village.

The incineration of these engineered boards for bio energy will result in an additional 656.750.000 kg CO2

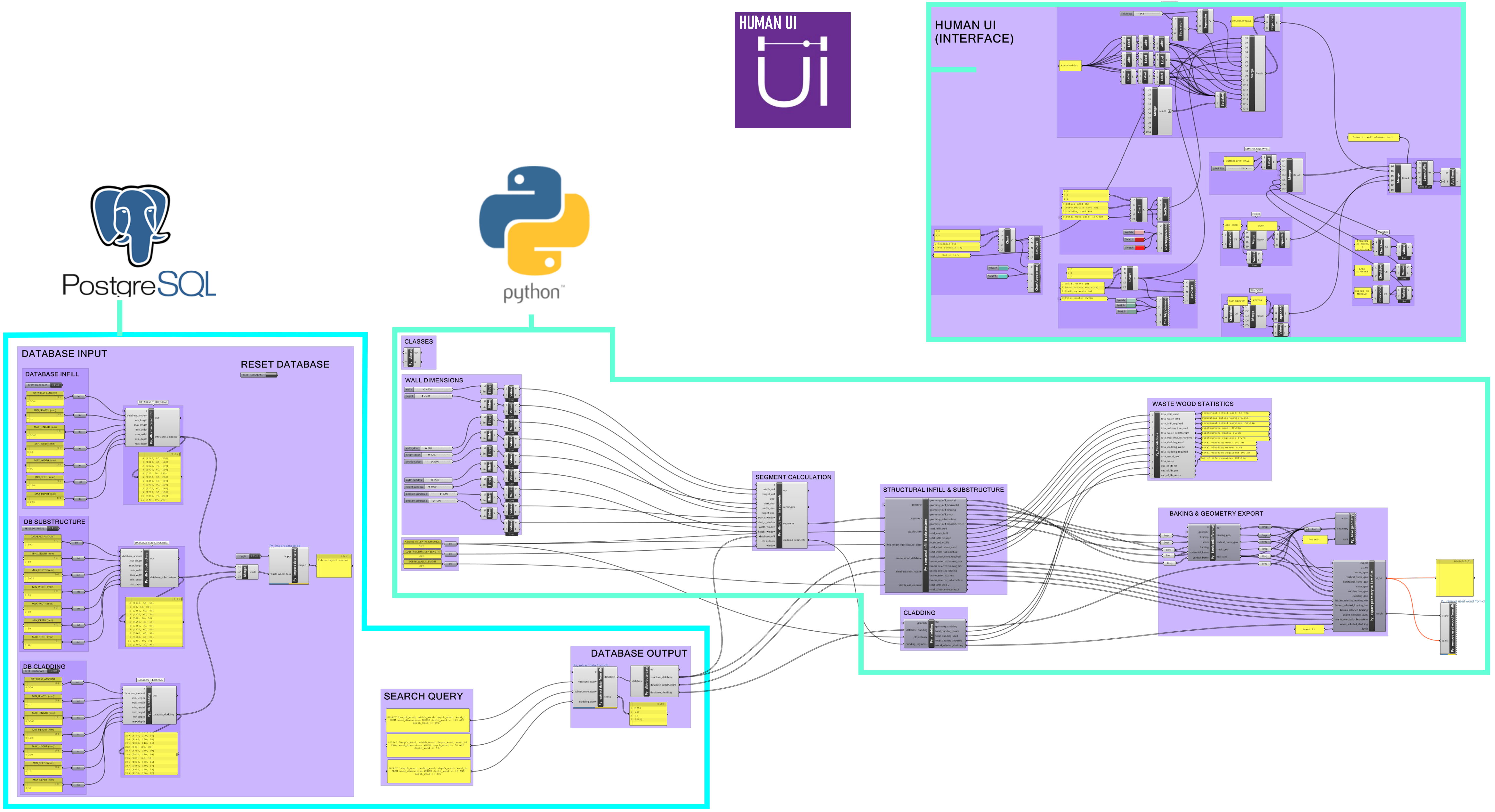


Oriented strand board, type F1, perpendicular to board

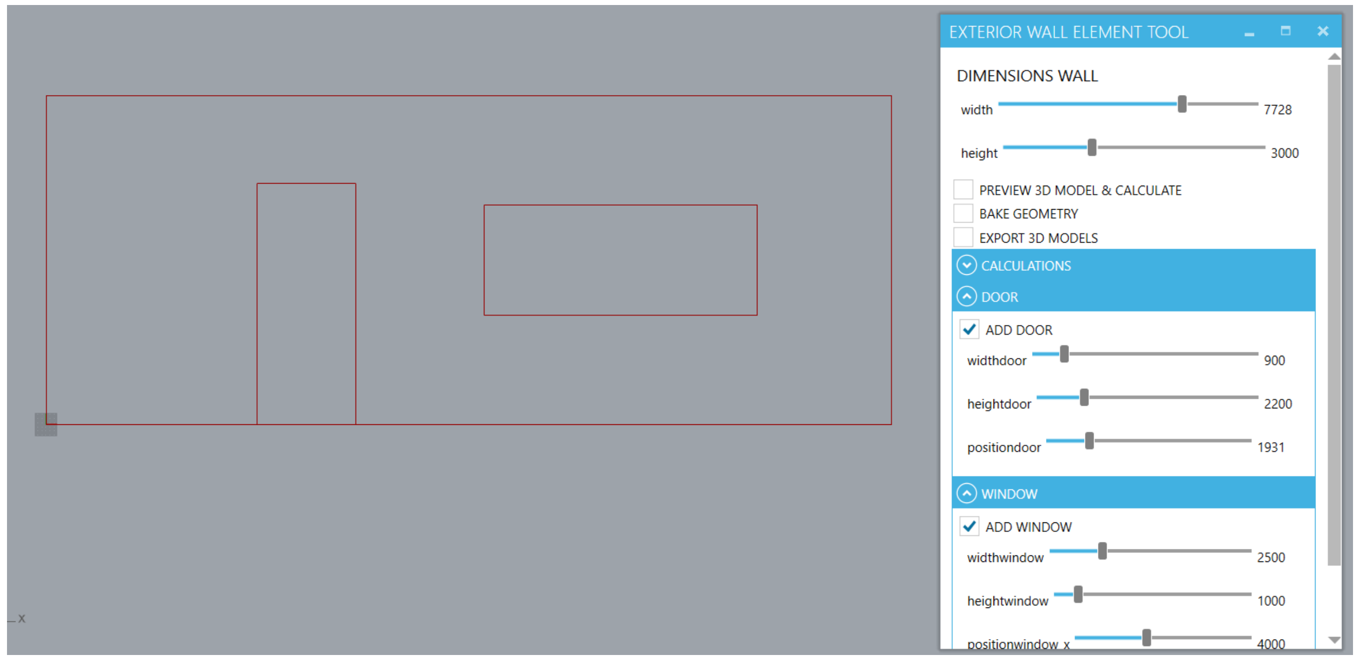
Datasheet view: All attributes — Show/Hide

Primary production energy, CO2 and water					
Embodied energy, primary production	ⓘ	18.4	-	20.3	MJ/kg
Sources: 12.2 MJ/kg (Athena Sustainable Materials Institute, 2009 (3)); 12.8 MJ/kg (Hammond and Jones, 2008); 33.1 MJ/kg (Ecoinvent v2.2)					
CO2 footprint, primary production	ⓘ	0.844	-	0.931	kg/kg
Sources: 0.794 kg/kg (Athena Sustainable Materials Institute, 2009 (3)); 0.932 kg/kg (Hammond and Jones, 2008); 0.933 kg/kg (Ecoinvent v2.2)					
Water usage	ⓘ	* 665	-	735	l/kg
Processing energy, CO2 footprint & water					
Coarse machining energy (per unit wt removed)	ⓘ	* 0.642	-	0.71	MJ/kg
Coarse machining CO2 (per unit wt removed)	ⓘ	* 0.0482	-	0.0532	kg/kg
Fine machining energy (per unit wt removed)	ⓘ	* 2.15	-	2.37	MJ/kg
Fine machining CO2 (per unit wt removed)	ⓘ	* 0.161	-	0.178	kg/kg
Grinding energy (per unit wt removed)	ⓘ	* 3.82	-	4.22	MJ/kg
Grinding CO2 (per unit wt removed)	ⓘ	* 0.287	-	0.317	kg/kg
Recycling and end of life					
Recycle	ⓘ	✗			
Recycle fraction in current supply	ⓘ	1.34	-	1.48	%
Downcycle	ⓘ	✓			
Combust for energy recovery	ⓘ	✓			
Heat of combustion (net)	ⓘ	* 19.7	-	20.7	MJ/kg
Combustion CO2	ⓘ	* 1.73	-	1.82	kg/kg
Landfill	ⓘ	✓			
Biodegrade	ⓘ	✓			

B GRASSHOPPERCANVAS



C EXTERIOR WALL ELEMENT TOOL



DIMENSIONS WALL

width 7728

height 3000

☐ PREVIEW 3D MODEL & CALCULATE

☐ BAKE GEOMETRY

☐ EXPORT 3D MODELS

CALCULATIONS

DOOR

✓ ADD DOOR

widthdoor 900

heightdoor 2200

positiondoor 1931

WINDOW

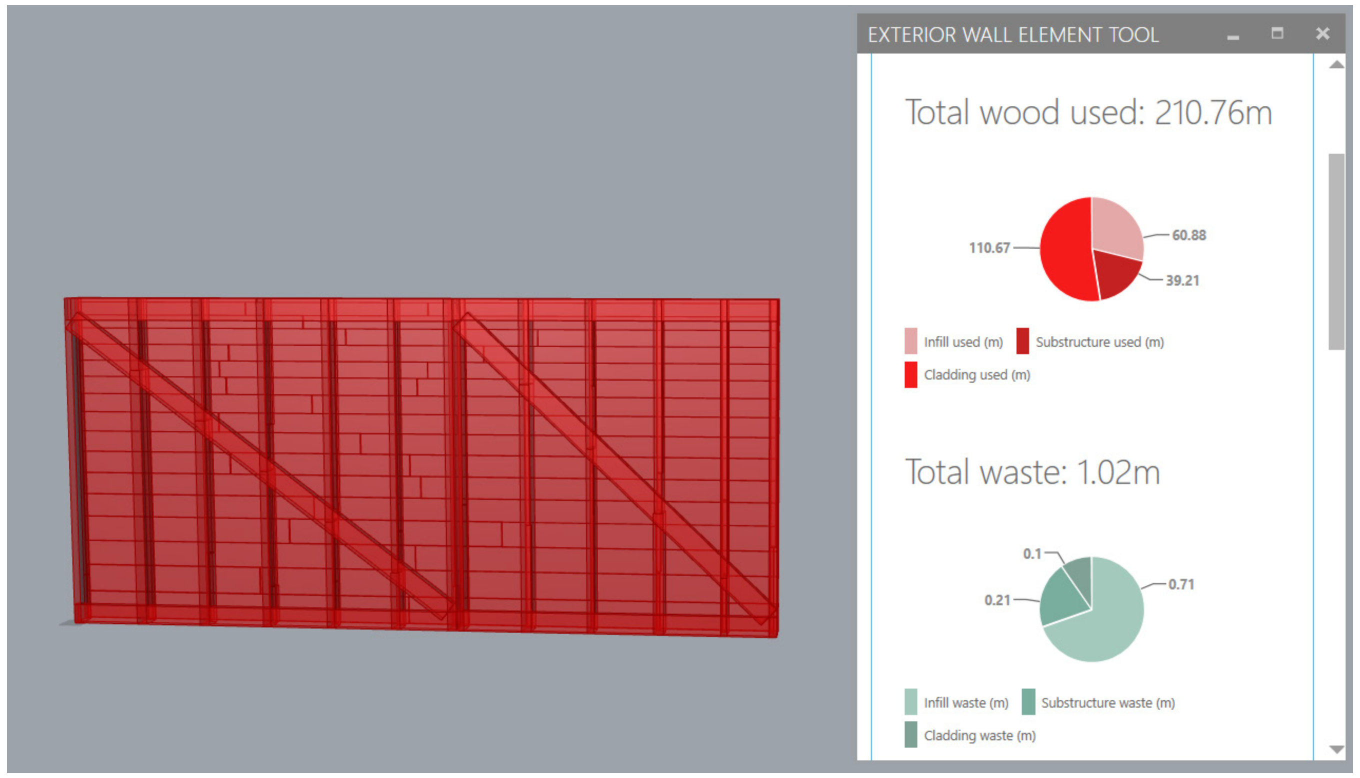
✓ ADD WINDOW

widthwindow 2500

heightwindow 1000

positionwindow x 4000

In Rhino a 2d preview is shown so that the user can see the proportions of the exterior wall element and the desired properties. The sliders in the tool allow the user to change the dimensions.



Total wood used: 210.76m

110.67

60.88

39.21

🟥 Infill used (m) 🟥 Substructure used (m)

🟥 Cladding used (m)

Total waste: 1.02m

0.1

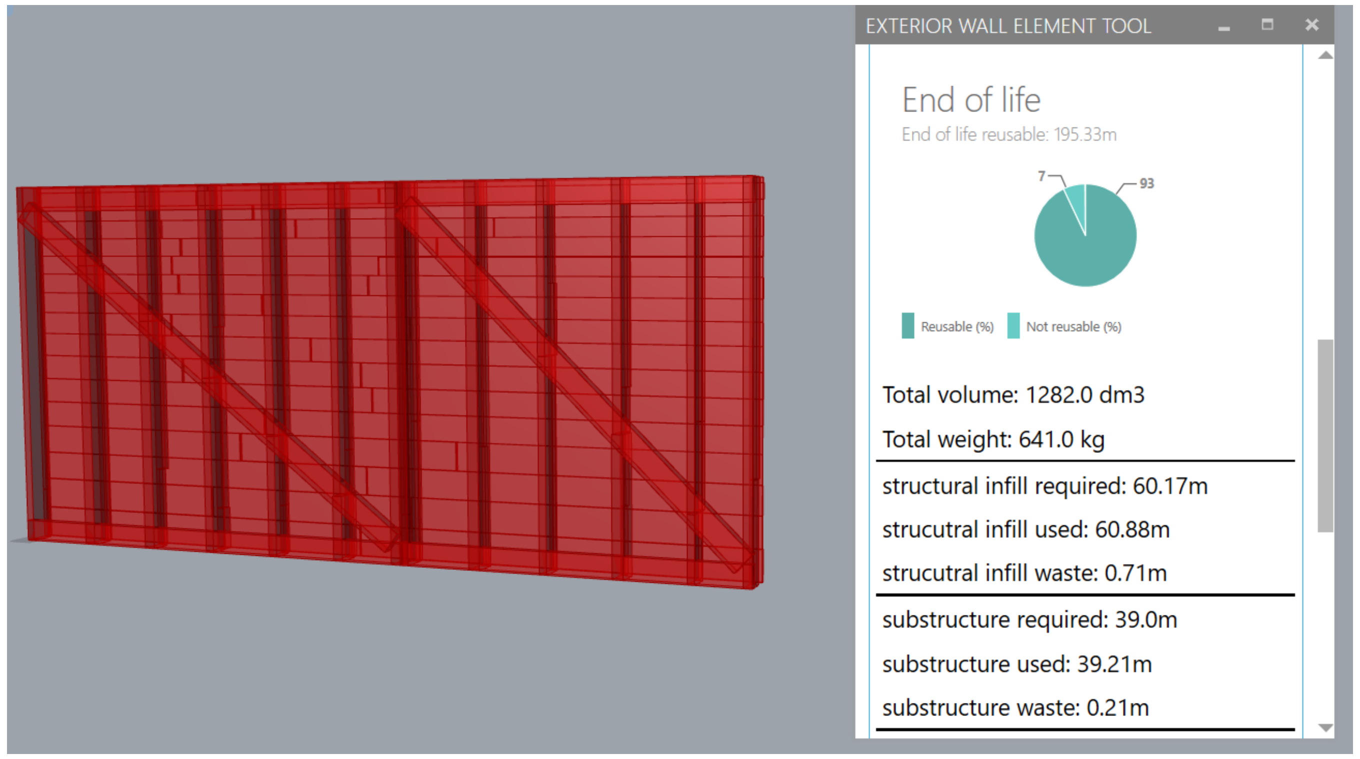
0.21

0.71

🟩 Infill waste (m) 🟩 Substructure waste (m)

🟩 Cladding waste (m)

When the preview 3d model & calculate option is checked a 3d preview is generated in grasshopper and shown in Rhino. All the different statistics about the material use, material waste are also visible.



End of life

End of life reusable: 195.33m

7

93

🟩 Reusable (%) 🟩 Not reusable (%)

Total volume: 1282.0 dm3

Total weight: 641.0 kg

structural infill required: 60.17m

strucutral infill used: 60.88m

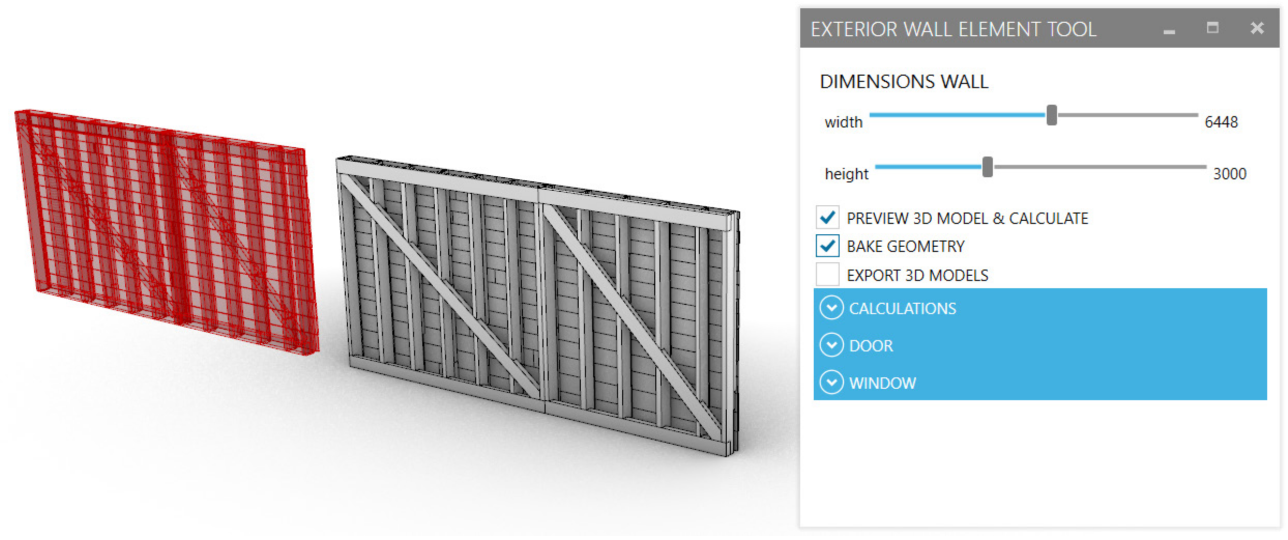
strucutral infill waste: 0.71m

substructure required: 39.0m

substructure used: 39.21m

substructure waste: 0.21m

The end of life is also calculated and the user can see how much wood is used for the structural infill, substructure and cladding.


EXTERIOR WALL ELEMENT TOOL

DIMENSIONS WALL

width 6448

height 3000

[✓] PREVIEW 3D MODEL & CALCULATE

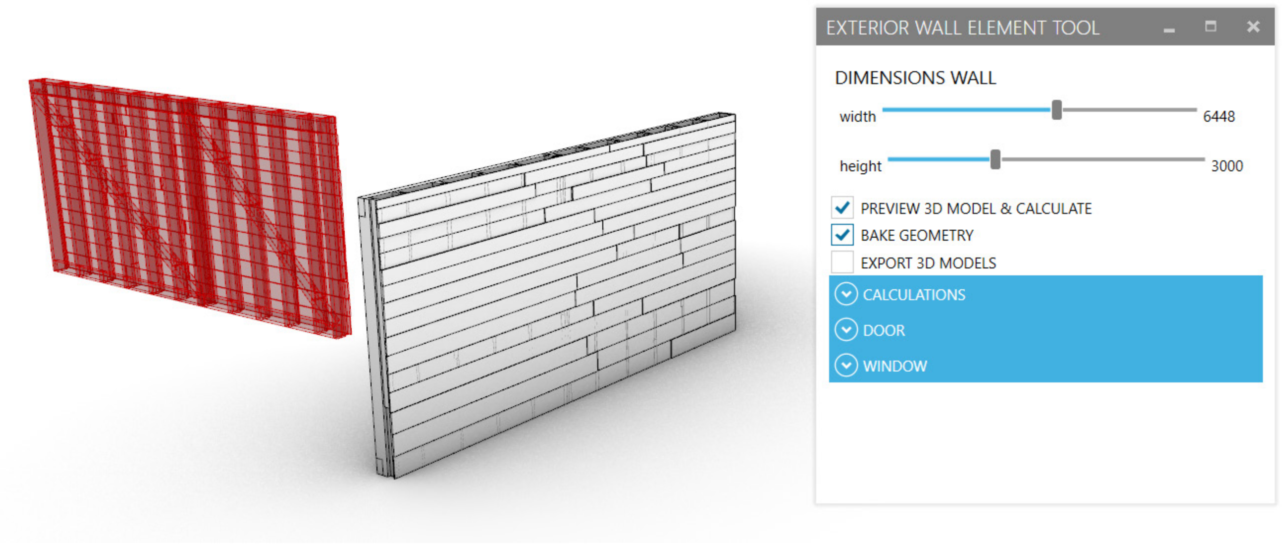
[✓] BAKE GEOMETRY

[ ] EXPORT 3D MODELS

CALCULATIONS

DOOR

WINDOW


EXTERIOR WALL ELEMENT TOOL

DIMENSIONS WALL

width 6448

height 3000

[✓] PREVIEW 3D MODEL & CALCULATE

[✓] BAKE GEOMETRY

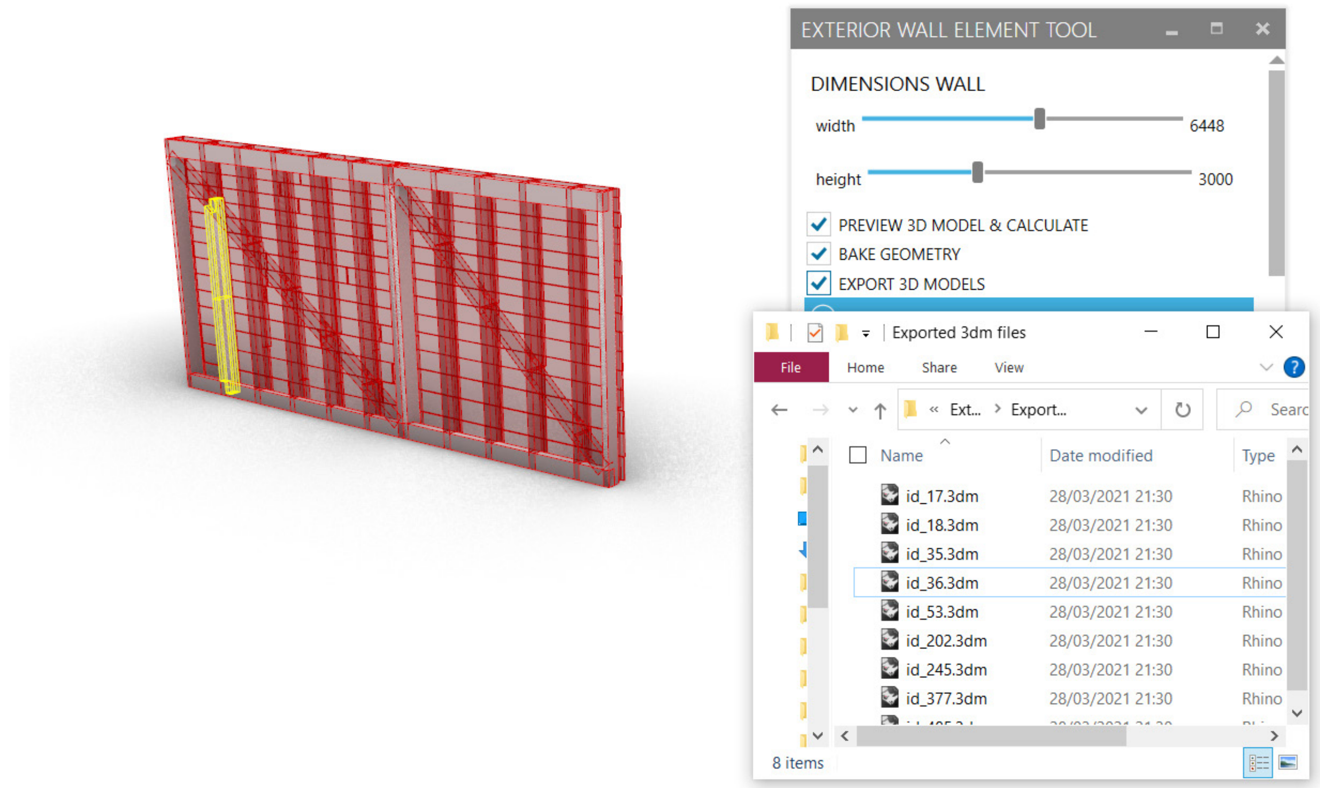
[ ] EXPORT 3D MODELS

CALCULATIONS

DOOR

WINDOW

The geometry can be baked in Rhino. The baked geometry shows the pieces of wood after the modifications.


EXTERIOR WALL ELEMENT TOOL

DIMENSIONS WALL

width 6448

height 3000

[✓] PREVIEW 3D MODEL & CALCULATE

[✓] BAKE GEOMETRY

[✓] EXPORT 3D MODELS

Exported 3dm files

Name	Date modified	Type
id_17.3dm	28/03/2021 21:30	Rhino
id_18.3dm	28/03/2021 21:30	Rhino
id_35.3dm	28/03/2021 21:30	Rhino
id_36.3dm	28/03/2021 21:30	Rhino
id_53.3dm	28/03/2021 21:30	Rhino
id_202.3dm	28/03/2021 21:30	Rhino
id_245.3dm	28/03/2021 21:30	Rhino
id_377.3dm	28/03/2021 21:30	Rhino

8 items

When the user is satisfied all the pieces of wood can be exported as a separate 3dm file that can be used for the digital fabrication. The number of the 3dm file correlates with the id number in the database. After this process all the exported wood will be removed from the database.

D TFC RESEARCH

In this chapter, timber frame construction methods are analyzed, and a standard timber frame construction element is dissected in different components required for an exterior wall.

HISTORY OF TIMBER CONSTRUCTION

The history of timber and wood construction is studied to see what techniques were used and what can be learned from the past before modern techniques were available.

Throughout history, there have been continuous recordings of buildings with wood. Until the beginning of the nineteen-century wood was the most widely used building material in most of Northern Europe and North America (Hansen, 1971).

One of the first forms of wooden structures were shelters in the form of wattle work. Wattle work combined with plaster such a mud created a shield against the elements. Wattle work finds in origin in weaving. Baskets were woven to transport goods during the hunters and gatherers era. When people started to settle, the wattle technique was used to create cages for the animals, fences, and housing (Zweger, 2011).

Over time tools were Improved, and this allowed for new timber construction techniques. Traditional timber constructions can be divided into two categories: log construction and skeleton construction. The appearance of log construction is characterized exclusively by the horizontal arrangements of its logs. In skeleton construction, the vertical members take on the load-bearing function (Hansen, 1971).

LOG CONSTRUCTION

The first log cabins were probably built in Northern Europe around 3500 BC (Bronze age). Basic tools, such as sharp stones, were used to chip the end of logs. These logs were placed on top of each and thereby creating stable structures. Between the gaps, moss or other soft materials were applied to make the structure watertight. This technique was also seen in wattle works (Zweger, 2011).

Over the years, better tools were developed and allowed for more elegant log construction, for

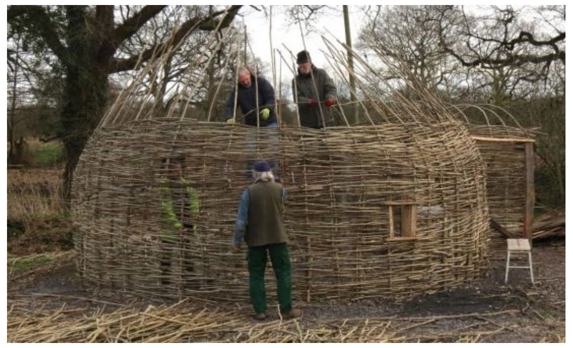


Figure Wattle shelter (avalon archaeology)

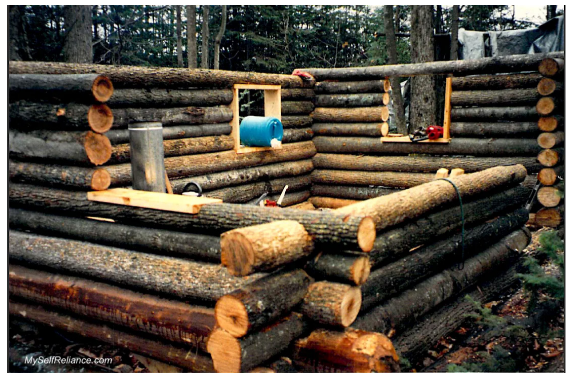


Figure Traditional log construction (myselfreliance)

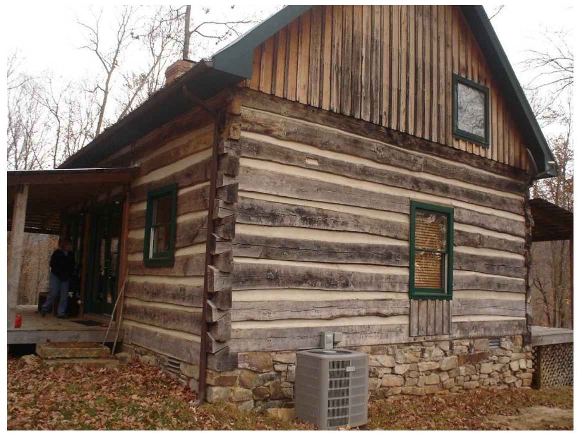


Figure Modern log construction with dovetail joint (small-cabin.com)

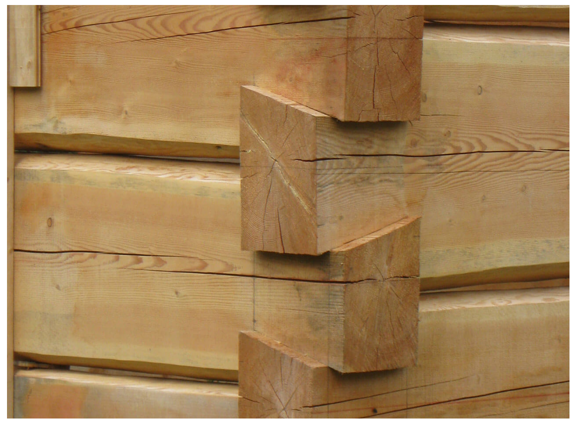


Figure Dovetail log construction (stonehouse woodworks)

example, the dovetail connection. Log construction requires the use of solid tree trunks and is therefore only applied in forest areas. The transportation cost and effort would be too much to achieve this building technique in non-forest areas.

The technique of log construction is still applied today but in a more modern way. EcoHouseMart created a system where the processed logs can be placed on top of each other with a tongue and groove joint. This system has a resemblance to LEGO. The log construction technique can also be seen in garden sheds. The small dimension of the garden shed will allow for the use of planks instead of logs.

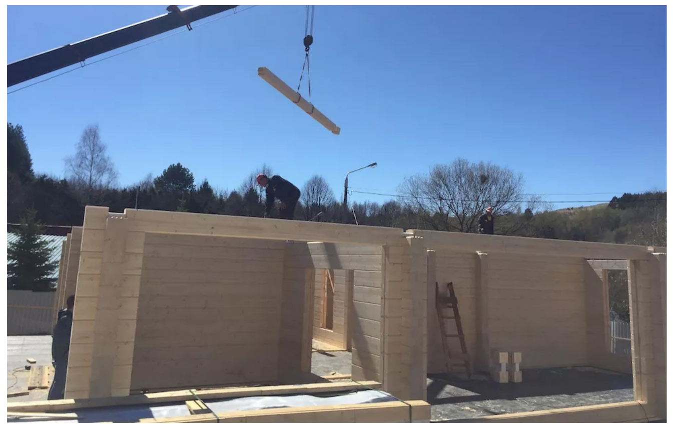
THE SKELETON CONSTRUCTION

When the logs are placed vertically, you speak of skeleton construction. Logs that are placed inside the ground are called posts. Logs that are vertically above the ground are defined as columns (Zweger, 2011). The benefit of working with this vertical construction is the freedom of design. With (horizontal) log construction, the building area is limited to the length of the log. With skeleton construction, the building area can be defined by the designer. The downside of skeleton construction is the sinking from the post in the ground due to its weight. This sinking resulted in new construction techniques to prevent this from happening.

One of the earlier forms of skeleton construction is the post and plank construction or corner-post construction. Between the vertical post, horizontal planks were placed into slots in the post. The horizontal planks can be thinner because they do not carry the load of the building.

POST AND BEAM CONSTRUCTION

When cities grew, techniques improved, and more buildings were built. This resulted in an increasing wood price due to scarcity. The forests around the cities were harvested, and the wood had to come from further. Therefore, new techniques emerged within the skeleton construction. Post and beam construction created a loadbearing frame with open spaces. These open spaces were filled up using an ancient wattle and daub technique. When the productions of bricks grew, bricks became more used as infill material.



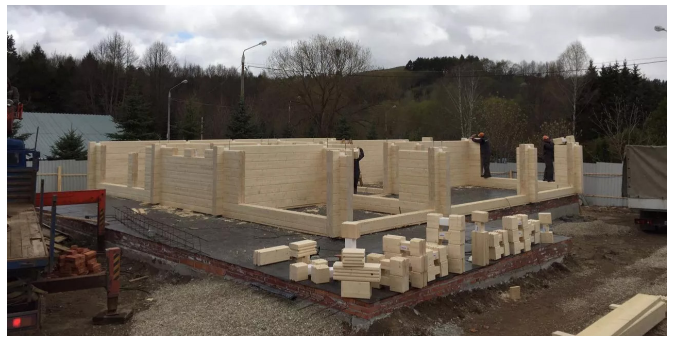


Figure Ecohousemart log construction (Ecohousemart)

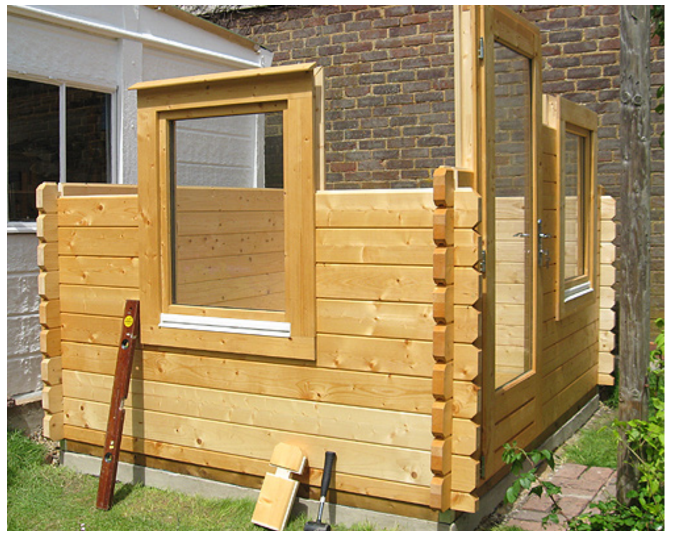


Figure Typical garden shed (logcabinadvice.co.uk)

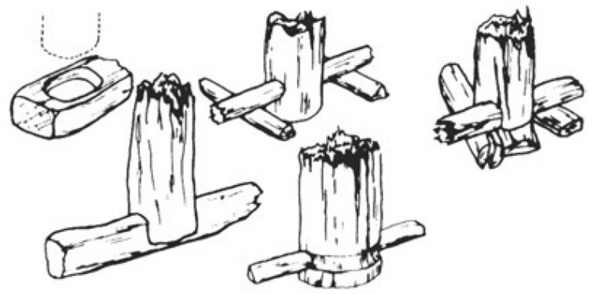


Figure Methods to prevent a post from sinking (Zweger, 2011)

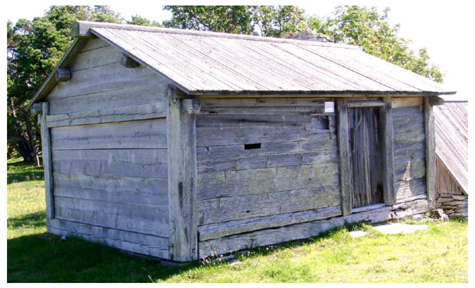
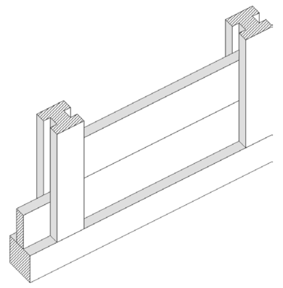
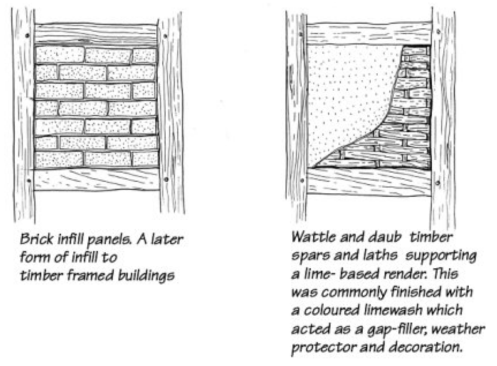



Figure Post and plank construction in Gotland, Sweden (Wikipedia)

Figure Wood and plank well, dating back 5256 B.C. (thehistoryblog)

Figure Post and beam housing, Germany (pinimg)


Brick infill panels. A later form of infill to timber framed buildings
Wattle and daub timber spars and laths supporting a lime- based render. This was commonly finished with a coloured limewash which acted as a gap-filler, weather protector and decoration.
Figure Brick or wattle and daub infill (UWE)

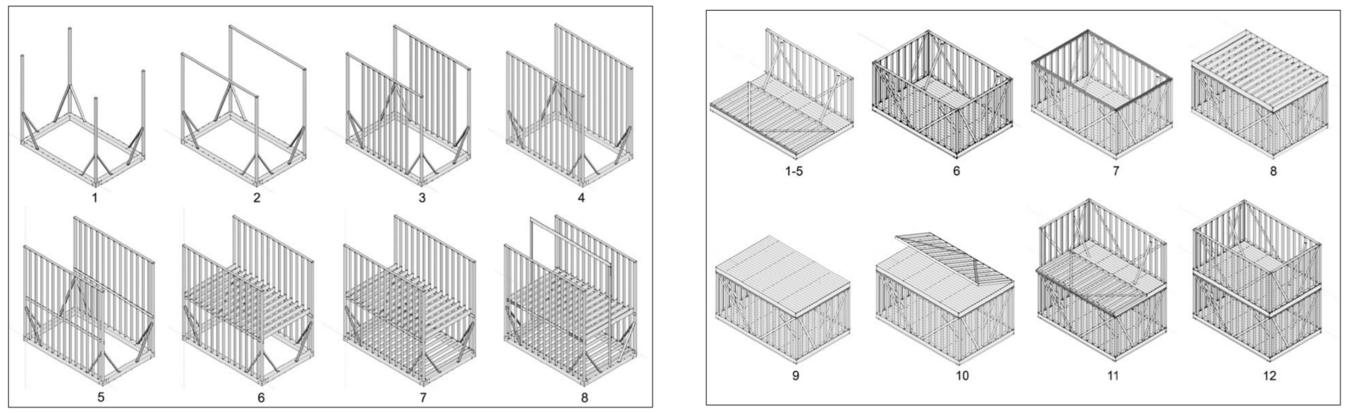


Figure Difference between balloon framing (left) and platform framing (right), (O'Brien, 2010)

BALLOON FRAMING

Balloon framing originated in 1830 in Chicago, USA. This method is close to what we use today with timber framing. With balloon framing, lightweight studs are used instead of heavy posts. Balloon framing components are nailed together rather than using wood joinery. Balloon framing is a cheaper alternative and does not require an experienced carpenter to make handcrafted dovetail or mortises and tenons joints. Balloon framing emerged due to the invention of the water-powered sawmill and the cheap production of steel nails (Joostdevree, 2012). Diagonal studs were used for bracing. Balloon framing used continuous studs from the bottom to the top.

PLATFORM FRAMING

When the long continuous wooden studs needed for balloon framing became scarce, a new type of framing was developed, platform framing. With platform framing, the studs have the height between two floors.

Again, diagonal bracing was necessary to make the structure stable. The invention of Oriented stranded board in 1960 made these bracing unnecessary. To this day, Platform framing is the most common method to build housing. Platform framing is similar to the Dutch word Houtskeletbouw (HSB), which can be translated to Timber Frame Construction (TFC) in this thesis.

TIMBER FRAME CONSTRUCTION

Today TFC is the most used construction method to build wooden walls within the Netherlands. In general, a TFC exterior wall element consists of 3 parts:

- Interior part
- Structural middle part
- Exterior part

STRUCTURAL MIDDLE PART

The structural middle part consists of the studs, the insulation, and an element to make the construction stable and stiff—for example, Oriented Strand Boards (OSB) or bracing. The wall element can be load bearing or non-load bearing. Load bearing means that the wall element also carries the weight of other building elements such as the floors or the roof. Non-load bearing means that the wall element only needs to be capable of holding its weight. The choice between load bearing and non-load bearing has an impact on the dimensions of the studs.

In the Netherlands, the strength class that is most used in the construction of wall elements are C18 & C24. A strength class consists of a letter and a number. The letter indicates if it is hardwood (D) or softwood (C). The most frequently used softwood for construction in the Netherlands is spruce wood (Vuren) (Centrum hout, 2005).

The studs in a TFC element always have a fixed center-to-center (CTC) distance. This allows for standardizations of other elements, for example, for the dimensions of insulation panels and engineered boards. The standard options for a center-to-center distance are 400mm and 600mm, with 600mm being more commonly used.

There is a variety of insulation available, ranging in

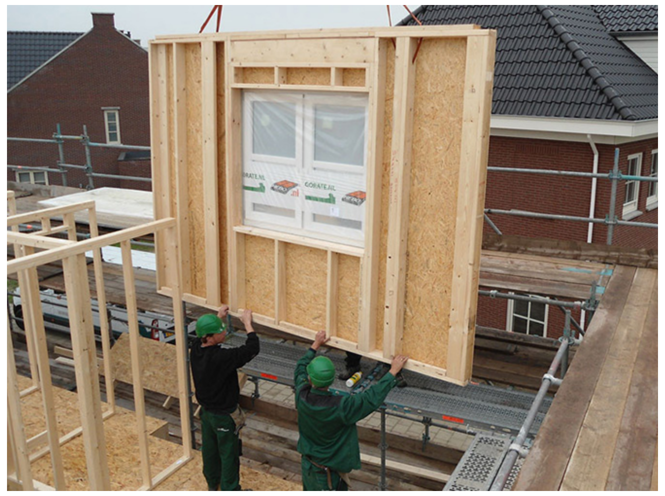


Figure Prefab structural middle part with OSB (De kroon prefab)

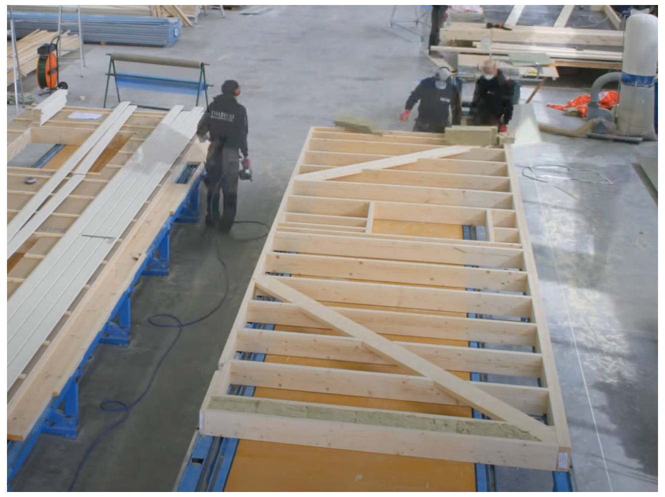


Figure Prefab structural middle part with bracing (Timbeco woodhouse)

Figure Flexible sheep wool insulation (eco-bouwmaterialen)

Figure Rigid insulation (celotex)

Figure Sprayed insulation (ultimate radiant barrier)

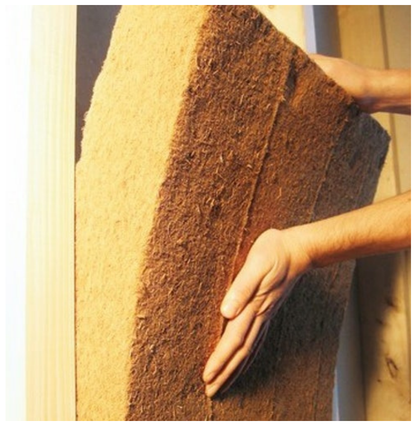


Figure Flexible wood fiber insulation (steico flex)

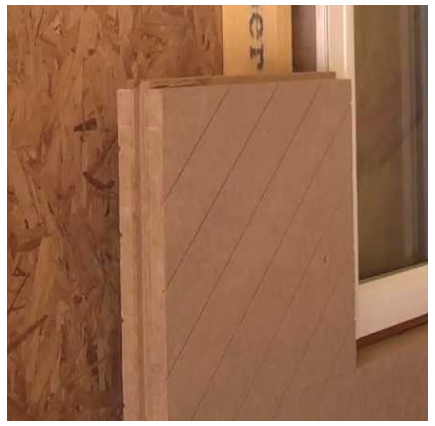


Figure Rigid wood fiber panel (Udifront)

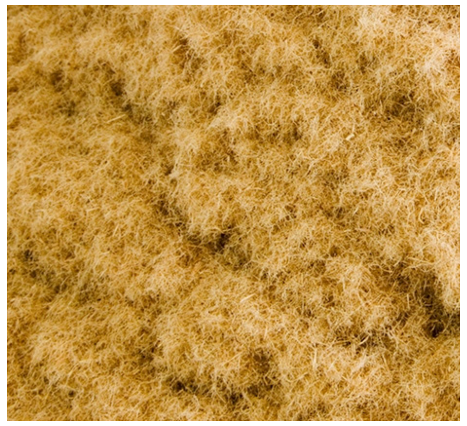


Figure Blown in wood fiber (isofloc)

all kinds of different materials. Some materials can also be applied in different methods. In general, there are three types of insulation.

- Flexible insulation that is delivered in rolls.
- Rigid insulation in the form of boards.
- Spray or blow-in insulation.

The insulation can be divided into insulation based on natural products or insulation based on fabricated materials. Insulation based on natural products is more environmentally friendly. However, more insulation material is needed to achieve the same effect as a fabricated material such as rockwool. For this thesis, an environmentally friendly solution is used. Using a fabricated material that possibly harms the climate is contradictory to the goals of this thesis. The insulation material can be applied within the three types. For example, wood fiber insulation can be flexible, rigid, or blown in.

EXTERIOR PART

The exterior part consists of the cladding and, if required, additional substructure. The cladding protects the structural middle part against the elements and gives the wall an aesthetic appearance. Different types of cladding with different materials can be used, allowing buildings to differ from each other. This thesis focuses on the reuse of waste wood and will research of waste wood can be applied as a cladding material. Therefore, the following references only consist of wood.

All these examples can be divided into two façade categories: an open façade or a closed façade. A closed façade is a façade where the cladding will act as a water-resistant barrier. For the façade to be closed, a specific building method is used. This building method is called clapwood, potdekselen, or rabat in Dutch.

An open façade is not entirely watertight.

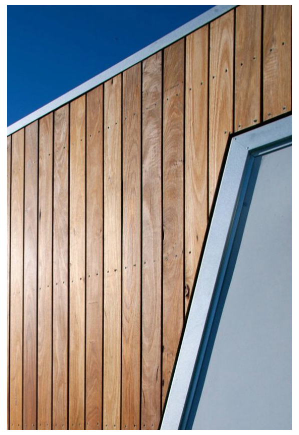


Figure Vertical open facade cladding (pinimg)

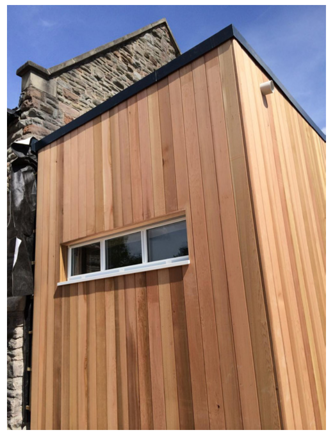


Figure Vertical closed facade cladding (Spahaus timber)

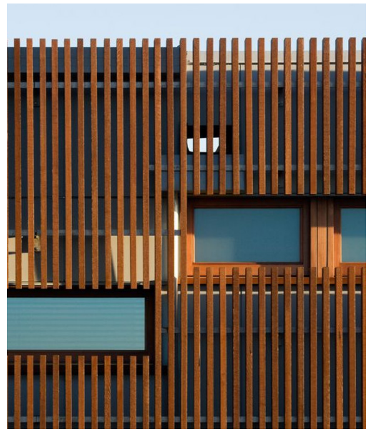


Figure Vertical open facade cladding (pinimg)

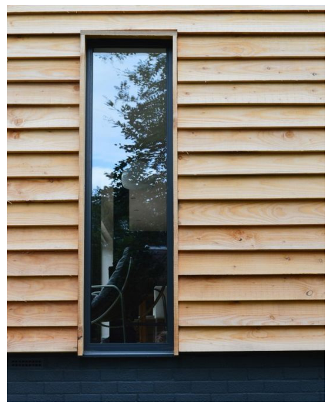


Figure Horizontal closed facade cladding (pinterest)

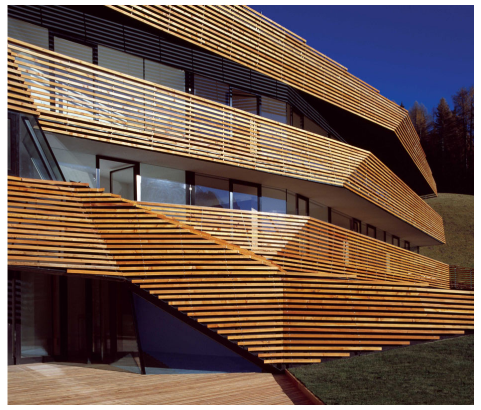


Figure Horizontal open facade cladding (pinterest)

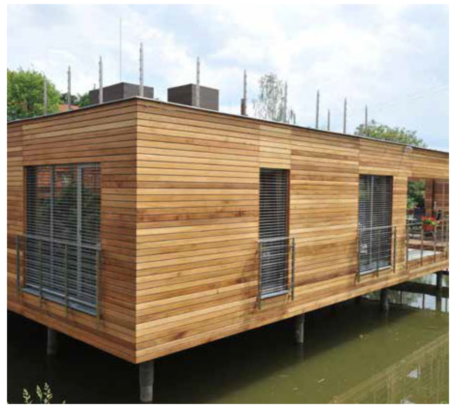


Figure Horizontal closed facade cladding (au-mex)

Therefore, an additional foil or layer is required to act as a water barrier. When the foil is installed, a cavity is required for ventilation and removing moisture in the construction. Depending on the chosen façade type and the exterior cladding, an additional substructure may be required.

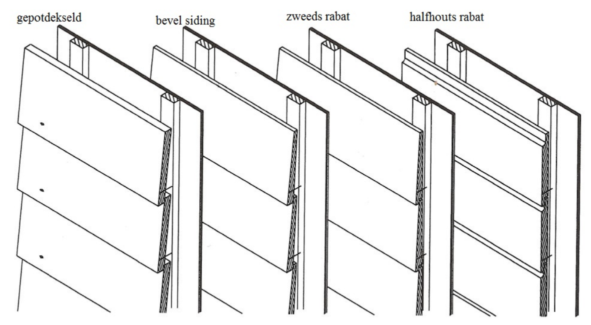


Figure Clapwood (Verdouw)

INTERIOR PART

The interior part consists of panels attached to the middle part that allows for a finishing. Based on the wishes of the user, this finishing can be paint or plaster. The interior panels can have additional fire resistance properties to prevent the fire from directly spreading to the construction's structural middle part.

FOIL

Depending on the construction method and the chosen façade type, foils are required to make the construction water resistant and prevent mold. An open façade requires two foils. The first foil is a water resistance and vapor permeable foil between the exterior part and the middle part. This foil will act as a water barrier while moisture inside the construction can still leave due to the vapor-permeable property. The second foil is placed between the middle part and the interior part and is a non-vapor permeable foil. This foil is set to prevent any moisture from inside the house from penetrating the construction. It is possible to have a vapor permeable construction, but this also requires the right insulation material. With a permeable construction, the foil between the middle and interior part is not required.

Figure Gypsum board (Home Statosphere)

Figure Water resistance foil (Gerbri)

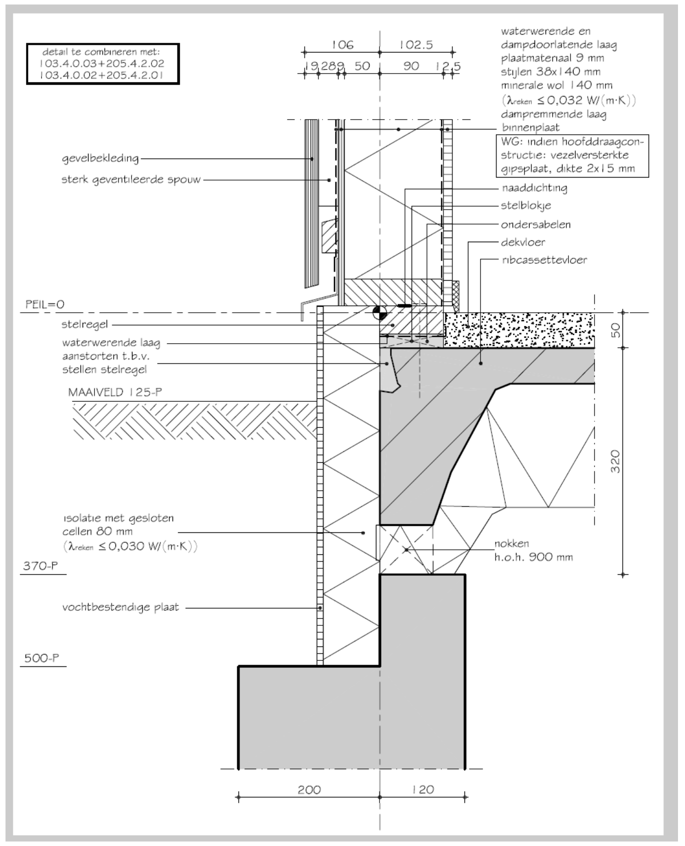
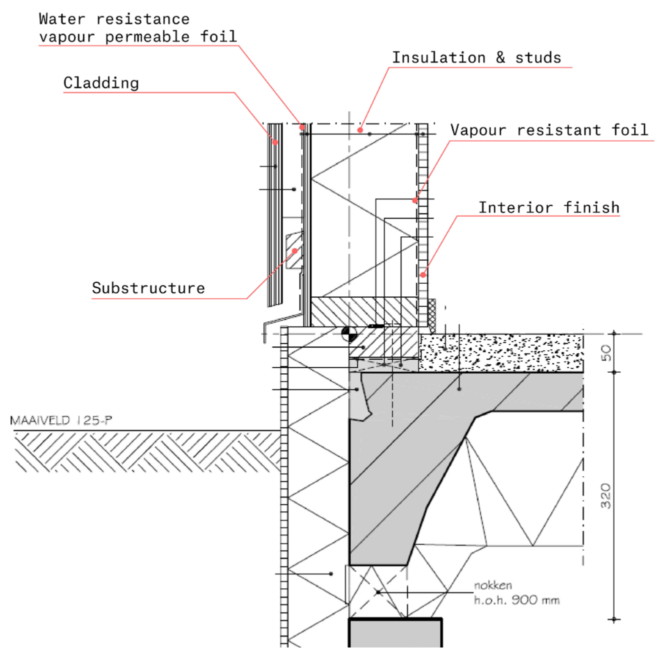
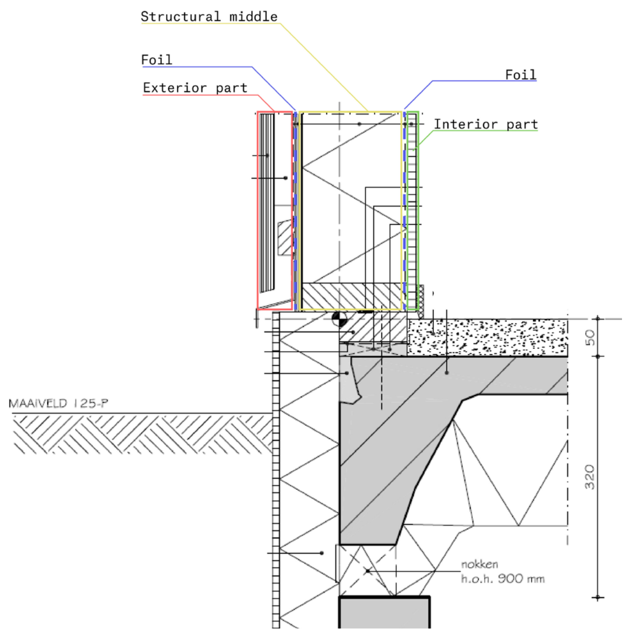


Figure TFC detail (SBR)

EXAMPLE

In the Netherlands, a reference guide with standard construction details that follow the building code is written by SBR (Nieman, 2012). The image below is TFC detail with an open façade. In the images below, the different parts that were explained in this chapter are highlighted.





PREFABRICATION

In the Netherlands, one of the most common ways to build a TFC element is prefabricated (Prefab). This means that at least the structural framing is assembled within the factory's controlled environment instead on the construction site. On the construction site, the conditions such as the weather can change regularly and negatively impact the quality of the construction.

Working with waste wood will result in additional challenges. Due to the constantly changing sizes of the waste wood, it is not recommended to exterior wall elements on the building site. On the building site and during transportation, the waste wood can get mixed up and. This will increase the building process's complexity because with waste wood every piece had a unique location where it needs to be assembled within the wall element. It is recommended to construct the exterior wall element from waste wood in a controlled environment, just like the prefabricated wall elements.

Prefabricated wall elements can differ from each other in the way they are designed and assembled. The assembly can be done manually, completely automatic, or a combination of both. How much of the wall is assembled in the factory can also vary between the prefabricated elements. It can range from just the structural frame to a complete wall element, including insulation, window frames, and cladding.

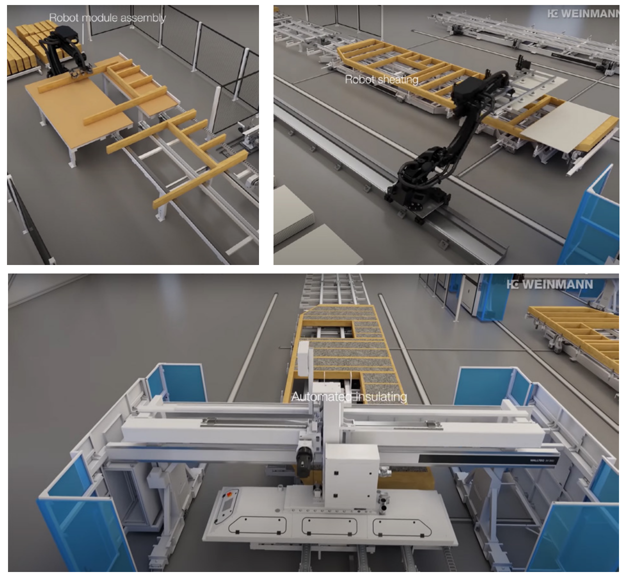


Figure Robotic framing, sheeting & insulating (Weinmann)

EXAMPLES

Timbeco woodhouse

At Timbeco woodhouse the complete exterior wall element gets assembled in a factory. The studs are placed on a horizontal workstation that can rotate to a vertical position. A pre-cut shape in the studs allows for the bracing to be placed. Gyproc gypsum board is placed between the cladding and structural middle. After applying the cladding to the wall element, the workstation is standing up so that the wall element can be flipped. Flexible insulation is added.

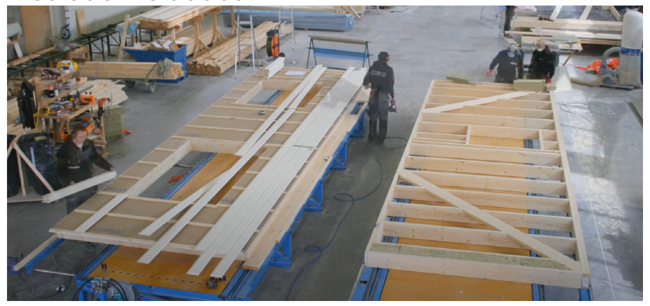


Figure 3.1 Prefab (Timbeco woodhouse)

Weinman

In this video, Weinman suggests a fully automatic assembly line without any manual labor. Sheeting is used to make the construction stiff, and the insulation is blow in by robots. This process requires a large starting investment.

Living wood

Living wood delivers only prefabricated structural frames. The insulation, cladding, and interior are assembled on the building sit.

Figure Prefab frame (Livingwood)

BUILDING REGULATIONS

Every building in the Netherlands needs to fulfill the building code (Rijksoverheid, 2021). The most essential codes that have a direct impact on the exterior wall element are:

- Insulation
- Airtightness
- Soundproofing
- Fireproofing

11.C.1 INSULATION

Since the oil crisis of 1973, the Dutch government made it required by law to insulate houses. How much a house should insulate increased over the years due to climate change and for saving gas. The value in which insulation is expressed is in the R-value (Rc = warmteweerstand). The unit of the R-value is kelvin square-meter per watt (m2K/W). Since the first of January 2021, the building code requires a minim RC value of 4,7 m2K/W for an exterior wall element.

Due to wood’s low thermal conductivity, this requirement can be met without problems for Timber frame construction methods.

AIRTIGHTNESS

Airtightness is a building property that shows how much air it is losing with a certain pressure difference. When air unknowingly enters the building, it is called infiltration, and when it leaves the building unknowingly, it is called exfiltration. If air enters or leaves the building depends on the pressure difference caused by wind, temperature, and residential behavior. The regulations for airtightness are implemented in the building code because of energy-saving and health reasons. The airtightness is measured with the airflow rate (luchtvolumestroom). The maximum allowable amount in the building code is 0,2 m3/s (qv10 ≤ 200 dm3/s). Infiltration and exfiltration can be prevented by securely closing gaps and seams with special tape. In a wall, the essential locations are around the window frames and between the interior finishing. For example, this can be the seam between two gypsum boards.

SOUNDPROOFING

In the building code, there are also regulations for soundproofing. These regulations make sure you can live comfortably in every house regardless of the location. The building code states the minimum amount of decibel (dB) that a wall must reduce and the maximum amount of dB allowed inside the building. Inside the building, the sound level cannot exceed 33db, and the façade must reduce sound by at least 20db. The most prominent sound leak is the ventilation grill. Gaps and seams in the construction can also have a significant influence. The structure itself can also reduce sound when materials are chosen for the interior and insulation with sound insulating properties.

FIRE RESISTANCE

It is vital to make sure your house does not burn instantly if something catches fire. The building code works with different fire compartments in a building. For a standard house, this is just one compartment. The compartment needs to survive at least 60 minutes before the construction can collapse.

CONCLUSION

In the Netherlands, the most used building method with wood is prefabricated timber frame construction. The developed exterior wall element for this thesis will also be constructed in a prefabricated method. The complexity of working with waste wood does not allow for in-situ construction. To keep everything more realistic, the decision is made to assemble the exterior wall element manually. The building codes in the Netherlands require specific demands for fire resistance, soundproofing, insulation, and airtightness. Due to the time frame of this graduation, it is not possible to research if waste wood alone can fulfill these demands. The choice is made to meet the building code requirements with already proven materials and elements. The goal is to select materials that have a positive impact on the environment where possible.

E PYTHON CODE

1. Classes

```python
print('import succes')

class wood_structural():
    def __init__(self, length, width, depth, db_id):
        self.length = length
        self.width = width
        self.depth = depth
        self.id = db_id

    def print_data(self):
        print(self.length, self.width, self.depth, self.id)


class wood_substructure(wood_structural):
    pass


class wood_cladding():
    def __init__(self, length, height, depth, db_id):
        self.length = length
        self.height = height
        self.depth = depth
        self.id = db_id


class coordinates_segments():
    def __init__(self, x_cor_start, z_cor_start, x_cor_end, z_cor_end):
        self.x_cor_start = x_cor_start
        self.z_cor_start = z_cor_start
        self.x_cor_end = x_cor_end
        self.z_cor_end = z_cor_end

    def print_cor(self):
        print(self.x_cor_start, self.z_cor_start, self.x_cor_end, self.z_cor_end)


class coordinates_segments_cladding(coordinates_segments):
    pass
```

2. Structural infill dataset

```python
"""Generates the structural infill data voor the database.
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Database_amount: number of wood in database for structural infill
        _Min_length: minimal length in mm of wood for structural infill
        _Max_length: maximum length in mm of wood for structural infill
        _Min_width: minimal width in mm of wood for structural infill
        _Max_width: maximum width in mm of wood for structural infill
        _Min_depth: minimal depth in mm of wood for structural infill
        _Max_depth: maximum depth in mm of wood for structural infill
    Output:
        structural_database: list with wood dimensions"""

import random

structural_database = []

for i in range(0,_Database_amount):
    length = random.randrange(_Min_length, _Max_length + 1, 10)
    width = random.randrange(_Min_width, _Max_width + 1, 10)
    depth = random.randrange(_Min_depth, _Max_depth + 1, 10)
    piece_of_wood =(length, width, depth)
    structural_database.append(piece_of_wood)
```

3. Substructure dataset

```
"""Generates the substructure data voor the database.
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Database_amount: number of wood in database for substructure
        _Min_length: minimal length in mm of wood for substructure
        _Max_length: maximum length in mm of wood for substructure
        _Min_width: minimal width in mm of wood for substructure
        _Max_width: maximum width in mm of wood for substructure
        _Min_depth: minimal depth in mm of wood for substructure
        _Max_depth: maximum depth in mm of wood for substructure
    Output:
        database_substructure: list with wood dimensions for substructure"""

import random

database_substructure = []

for i in range(0,_Database_amount):
    length = random.randrange(_Min_length, _Max_length + 1 ,10)
    width = random.randrange(_Min_width, _Max_width + 1 ,10)
    depth = random.randrange(_Min_depth, _Max_depth + 1 ,10)
    piece_of_wood = (length, width, depth)
    database_substructure.append(piece_of_wood)
```

4. Cladding dataset

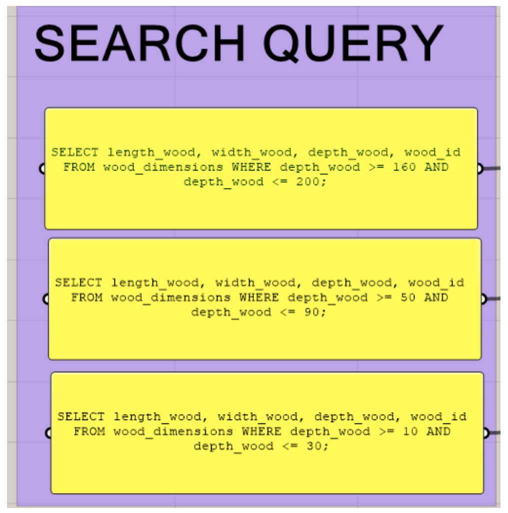
```
"""Generates the cladding data voor the database.
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Database_amount: number of wood in database for cladding
        _Min_length: minimal length in mm of wood for cladding
        _Max_length: maximum length in mm of wood for cladding
        _Min_width: minimal width in mm of wood for cladding
        _Max_width: maximum width in mm of wood for cladding
        _Min_depth: minimal depth in mm of wood for cladding
        _Max_depth: maximum depth in mm of wood for cladding
    Output:
        database_cladding: list with wood dimensions for cladding"""

import random

database_cladding = []

for i in range(0, _Database_amount):
    length = random.randrange(_Min_length, _Max_length + 1, 10)
    height = random.randrange(_Min_height, _Max_height + 1, 10)
    depth = random.randrange(_Min_depth, _Max_depth + 1, 1)
    piece_of_cladding = (length, height, depth)
    database_cladding.append(piece_of_cladding)
```

5. Sent datasets to database

```python
""" Combines data and sents it to PostgreSQL database.
    Inputs:
        _Apply: boolean toggle to sent data to database
    Output:
        output: conformation when data is sent.
"""


def clear_data():
    #deletes existing data in database
    import psycopg2 as pg2
    conn = pg2.connect(database='hout', user='postgres', password='----')
    cur = conn.cursor()
    cur.execute("TRUNCATE wood_dimensions RESTART IDENTITY ")
    conn.commit()
    print('data removed')

def apply_data():
    #applies data to database
    for piece_of_wood in _Waste_wood_data:
        length = piece_of_wood[0]
        width = piece_of_wood[1]
        depth = piece_of_wood[2]
        strength_class = 'C18'
        collector_id = random.randint(1,6)

        cur.execute("INSERT INTO
        wood_dimensions(length_wood,width_wood,depth_wood,strength_class,collector_id)\
        VALUES (%s,%s,%s,%s,%s)",(length,width,depth,strength_class,collector_id));
        conn.commit()

    print('data import succes')
    return 'data import succes'


import psycopg2 as pg2
import random

conn = pg2.connect(database='---', user='---', password='----')
cur = conn.cursor()

if _Apply == True:
    clear_data()
    output = apply_data()
```

6. Extract data from database

```python
""" Extracts data from PostgreSQL database.
    Inputs:
        x: reset toggle to extract new data
        _Structural_query: query to select wood suitable for structural infill
        _Substructure_query: query to select wood suitable for the substructure
        _Cladding_query: query to select wood suitable for the cladding
    Output:
        database: list of extracted data
        check: first line of data to check if data was extracted

"""
import psycopg2 as pg2
import ctypes.wintypes
import pickle


def import_data(query, database):
    cur.execute(query)
    conn.commit()
    data_set = cur.fetchall()
    print('data imported')
    for data in data_set:
        database.append(data)


database_infill = []
database_substructure = []
database_cladding = []

conn = pg2.connect(
    database='hout',
    user = 'postgres',
    password = '----')
cur = conn.cursor()

import_data(_Structural_query, database_infill)
import_data(_Substructure_query, database_substructure)
import_data(_Cladding_query, database_cladding)

total_database = (database_infill, database_substructure, database_cladding)

CSIDL_PERSONAL = 5       # My Documents
SHGFP_TYPE_CURRENT = 0   # Get current, not default value
buf = ctypes.create_unicode_buffer(ctypes.wintypes.MAX_PATH)
ctypes.windll.shell32.SHGetFolderPathW(None, CSIDL_PERSONAL, None, SHGFP_TYPE_CURRENT, buf)


# filename
fname = buf.value + "\Exterior wall element tool\database_pickle"
print(fname)
#fname = "C:/Users/jvane/OneDrive/Building Technology/GRADUATION/GRASSHOPPER/database_pickle"
# filehandle
fhandle = open(fname, 'wb')
# dump data to file
pickle.dump(total_database, fhandle, protocol=2)
# close file
fhandle.close()
# output file
database = fname

check = database_infill[0]
print(check)
check = database_substructure[0]
print(check)
check = database_cladding[0]
print(check)
```



SEARCH QUERY

```sql
SELECT length_wood, width_wood, depth_wood, wood_id FROM wood_dimensions WHERE depth_wood >= 160 AND depth_wood <= 200;
```

```sql
SELECT length_wood, width_wood, depth_wood, wood_id FROM wood_dimensions WHERE depth_wood >= 50 AND depth_wood <= 90;
```

```sql
SELECT length_wood, width_wood, depth_wood, wood_id FROM wood_dimensions WHERE depth_wood >= 10 AND depth_wood <= 30;
```

7. GH_CPython to GH_Python

```python
"""distributes the data from the database in 3 classes and designated list.
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Database: list with data
    Output:
        structural_database: list with wood in structural class
        database_substructure: list with wood in substructure class
        database_cladding: list with wood in cladding class"""


import ctypes.wintypes
import sys

CSIDL_PERSONAL = 5       # My Documents
SHGFP_TYPE_CURRENT = 0   # Get current, not default value
buf = ctypes.create_unicode_buffer(ctypes.wintypes.MAX_PATH)
ctypes.windll.shell32.SHGetFolderPathW(None, CSIDL_PERSONAL, None, SHGFP_TYPE_CURRENT, buf)

install_path = buf.value + "\Exterior wall element tool\wood_classes"
sys.path.append(install_path)

from classes_wood import *
import pickle
import operator

fhandle = open(_Database, 'rb')
total_database = pickle.load(fhandle)
fhandle.close()

database_infill_list = total_database[0]
database_substructure_list = total_database[1]
database_cladding_list = total_database[2]

structural_database = []
database_substructure = []
database_cladding = []

for wood in database_infill_list:
    piece_of_wood = wood_structural(wood[0], wood[1], wood[2], wood[3])
    structural_database.append(piece_of_wood)

structural_database.sort(key = operator.attrgetter('length'))

for wood in database_substructure_list:
    piece_of_wood = wood_substructure(wood[0], wood[1], wood[2], wood[3])
    database_substructure.append(piece_of_wood)

database_substructure.sort(key = operator.attrgetter('length'))
database_substructure.sort(key = operator.attrgetter('depth'))

for wood in database_cladding_list:
    piece_of_cladding = wood_cladding(wood[0], wood[1], wood[2], wood[3])
    database_cladding.append(piece_of_cladding)

database_cladding.sort(key = operator.attrgetter('length'))
database_cladding.sort(key = operator.attrgetter('height'), reverse = True)
```

8. Segment calculation

```python
"""Divides the wall in multiple segments
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Width_wall: width of the wall in mm
        _Height_wall: height of the wall in mm
        _Door: boolean toggle, if True apply door
        _Start_door: x coordinate of start door
        _Height_door: height of the door in mm
        _Start_x_window: x coordinate where window starts
        _Start_z_window: z coordinate where window starts
        _Width_window: width of the window in mm
        _Height_window: height of the window in mm
        _Database_infill: list with wood for structural infill
        _Ctc_distance: centre to centre distance between studs in mm
        _Window: boolean toggle, if True apply window
    Output:
        rectangle: list with rectangle geometry for preview in Rhino
        segments: list with dimensions and positions of segments for structural infill & substructure
        cladding_segments: list with dimensions and positions of segments for cladding
        database_substructure: list with wood in substructure class
        database_cladding: list with wood in cladding class"""

import Rhino.Geometry as rg
import math
from classes_wood import *

rectangles = []
segments = []
pt0 = rg.Point3d(0, 0, 0)
height_point = rg.Point3d(0, 1, 0)
zaxis = height_point-pt0
plane = rg.Plane(pt0, zaxis)

width_segments = []
cladding_segments = []

def number_segments(x_start, z_start, x_end, z_end):
    """Calculate how many segments are required for these dimensions"""
    width_element = x_end - x_start
    height_element = z_end - z_start
    max_width = int((_Database_infill[-1].length**2 - height_element**2)**0.5)

    while max_width < width_element:
        segment_width = math.floor(max_width / _Ctc_distance) * _Ctc_distance
        width_element -= segment_width
        segment = coordinates_segments(x_start, z_start, x_start + segment_width, z_end)
        segments.append(segment)
        x_start = x_start + segment_width

    if max_width > width_element:
        segment = coordinates_segments(x_start, z_start, x_end, z_end)
        segments.append(segment)

def rectangle_wall(_Width_wall, _Height_wall):
    """Generate 2d outline of wall for preview in Rhino """
    pt0 = rg.Point3d(0, 0, 0)
    pt1 = rg.Point3d(_Width_wall, 0, _Height_wall)
    rectangle = rg.Rectangle3d(plane,pt0, pt1)
    rectangles.append(rectangle)


def rectangle_door(_Start_door, _Width_door, _Height_door):
    """Generate 2d outline of door for preview in Rhino """
    pt0 = rg.Point3d(_Start_door, 0, 0)
    pt1 = rg.Point3d(_Start_door + _Width_door, 0, _Height_door)
    rectangle = rg.Rectangle3d(plane,pt0, pt1)
    rectangles.append(rectangle)


def rectangle_window(_Start_x_window, _Start_z_window, _Width_window, _Height_window):
    """Generate 2d outline of window for preview in Rhino """
    pt0 = rg.Point3d(_Start_x_window, 0, _Start_z_window)
    pt1 = rg.Point3d(_Start_x_window + _Width_window, 0, _Start_z_window + _Height_window)
    rectangle = rg.Rectangle3d(plane,pt0, pt1)
    rectangles.append(rectangle)
```

```python
if _Door == False and _Window == False:
    rectangle_wall(_Width_wall, _Height_wall)
    number_segments(0, 0, _Width_wall, _Height_wall)
    cladding_segments.append(coordinates_segments_cladding(0, 0, _Width_wall, _Height_wall))

if _Door == True and _Window == False:
    rectangle_wall(_Width_wall, _Height_wall)
    rectangle_door(_Start_door, _Width_door, _Height_door)

    number_segments(0, 0, _Start_door, _Height_wall)
    number_segments(_Start_door, _Height_door, _Start_door + _Width_door, _Height_wall)
    number_segments(_Start_door + _Width_door, 0, _Width_wall, _Height_wall)

    cladding_segments.append(coordinates_segments_cladding(0, 0, _Start_door, _Height_door))
    cladding_segments.append(coordinates_segments_cladding(_Start_door + _Width_door, 0, _Width_wall, _Height_door))
    cladding_segments.append(coordinates_segments_cladding(0, _Height_door, _Width_wall, _Height_wall))

if _Door == False and _Window == True:
    rectangle_wall(_Width_wall, _Height_wall)
    rectangle_window(_Start_x_window, _Start_z_window, _Width_window, _Height_window)

    number_segments(0, 0, _Start_x_window, _Height_wall)
    number_segments(_Start_x_window, 0, _Start_x_window + _Width_window, _Start_z_window)
    number_segments(_Start_x_window, _Start_z_window + _Height_window, _Start_x_window + _Width_window, _Height_wall)
    number_segments(_Start_x_window + _Width_window, 0, _Width_wall, _Height_wall)

    cladding_segments.append(coordinates_segments_cladding(0, 0, _Width_wall, _Start_z_window))
    cladding_segments.append(coordinates_segments_cladding(0, _Start_z_window + _Height_window, _Width_wall,
_Height_wall))
    cladding_segments.append(coordinates_segments_cladding(0, _Start_z_window, _Start_x_window, _Start_z_window +
_Height_window))
    cladding_segments.append(coordinates_segments_cladding(_Start_x_window + _Width_window, _Start_z_window, _Width_wall,
_Start_z_window + _Height_window))

if _Door == True and _Window == True:
    rectangle_wall(_Width_wall, _Height_wall)
    rectangle_door(_Start_door, _Width_door, _Height_door)
    rectangle_window(_Start_x_window, _Start_z_window, _Width_window, _Height_window)

    if _Start_x_window < _Start_door:
        number_segments(0, 0, _Start_x_window, _Height_wall)
        number_segments(_Start_x_window, 0, _Start_x_window + _Width_window, _Start_z_window)
        number_segments(_Start_x_window, _Start_z_window + _Height_window, _Start_x_window + _Width_window, _Height_wall)
        number_segments(_Start_x_window + _Width_window, 0, _Start_door, _Height_wall)
        number_segments(_Start_door, _Height_door, _Start_door + _Width_door, _Height_wall)
        number_segments(_Start_door + _Width_door, 0, _Width_wall, _Height_wall)

        cladding_segments.append(coordinates_segments_cladding(0, 0, _Start_door, _Start_z_window))
        cladding_segments.append(coordinates_segments_cladding(0, _Start_z_window + _Height_window, _Start_door,
_Height_wall))
        cladding_segments.append(coordinates_segments_cladding(0, _Start_z_window, _Start_x_window, _Start_z_window +
_Height_window))
        cladding_segments.append(coordinates_segments_cladding(_Start_x_window + _Width_window, _Start_z_window,
_Start_door, _Start_z_window + _Height_window))
        cladding_segments.append(coordinates_segments_cladding(_Start_door + _Width_door, 0, _Width_wall, _Height_door))
        cladding_segments.append(coordinates_segments_cladding(_Start_door, _Height_door, _Width_wall, _Height_wall))

    else:
        number_segments(0, 0, _Start_door, _Height_wall)
        number_segments(_Start_door, _Height_door, _Start_door + _Width_door, _Height_wall)
        number_segments(_Start_door + _Width_door, 0, _Start_x_window, _Height_wall)
        number_segments(_Start_x_window, 0, _Start_x_window + _Width_window, _Start_z_window)
        number_segments(_Start_x_window, _Start_z_window + _Height_window, _Start_x_window + _Width_window, _Height_wall)
        number_segments(_Start_x_window + _Width_window, 0, _Width_wall, _Height_wall)

        cladding_segments.append(coordinates_segments_cladding(0, 0, _Start_door, _Height_door))
        cladding_segments.append(coordinates_segments_cladding(0, _Height_door, _Start_door + _Width_door, _Height_wall))
        cladding_segments.append(coordinates_segments_cladding(_Start_door + _Width_door, 0, _Width_wall,
_Start_z_window))
        cladding_segments.append(coordinates_segments_cladding(_Start_door + _Width_door, _Start_z_window,
_Start_x_window, _Start_z_window + _Height_window))
        cladding_segments.append(coordinates_segments_cladding(_Start_door + _Width_door, _Start_z_window +
_Height_window, _Width_wall, _Height_wall))
        cladding_segments.append(coordinates_segments_cladding(_Start_x_window + _Width_window, _Start_z_window,
_Width_wall, _Start_z_window + _Height_window))
```

9. Structural infill & substructure

```python
"""Generates the geometry for the structural infill & substructure and calculates the waste
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Generate: Boolean toggle, if True apply
        _Segments: list with dimensions and positions of segments
        _Ctc_distance: centre to centre distance between studs in mm
        _Min_length_substructure_piece: minimal length that a piece of substructure can have in mm
        _Waste_wood_database: list with wood for structural infill
        _Database_substructure: list with wood for substructure
        _Depth_wall_element: depth of the exterior wall element in mm

    Output:
        geometry_infill_vertical: list with breps for vertical frame
        geometry_infill_horizontal: list with breps for horizontal frame
        geometry_infill_bracing: list with breps for bracing
        geometry_infill_studs: list with brep for studs
        total_infill_used: meter of wood used for structural infill
        total_waste_infill: waste in meter for structural infill
        total_infill_required: meter of wood required for structural infill
        total_substructure_used: meter of wood used for substructure
        total_waste_substructure: waste in meter for substructure
        total_substructure_required: meter of wood required for substructure
        beams_selected_framing_ver: list of wood selected for vertical framing
        beams_selected_framing_hor: list of wood selected for horizontal framing
        beams_selected_bracing: list of wood selected for bracing
        beams_selected_studs: list of wood selected for studs
        beams_selected_substructure: list of wood selected for substructure
        """

import Rhino.Geometry as rg
import random
import math
import operator
import itertools


class select_structural:
    def __init__(self, selected_wood, waste, geometry):
        self.selected_wood = selected_wood
        self.waste = waste
        self.geometry = geometry


def outer_frame_vertical(required_length, _Waste_wood_database, position):
    """Selects wood from db and generates geometry for vertical frame"""
    #pick piece of wood from database for vertical outer frame (left & right)
    chosen_beam = next(beam for beam in _Waste_wood_database if beam.length >= required_length)

    #determine additional waste with picked piece of wood
    waste = chosen_beam.length - required_length

    if position == 'left': #0 = outerleft vertical
        pt0 = rg.Point3d(seg_ver.x_cor_start,0, seg_ver.z_cor_start)
        pt1 = rg.Point3d(seg_ver.x_cor_start + chosen_beam.width, chosen_beam.depth, seg_ver.z_cor_end)
        x_start = seg_ver.x_cor_start
        z_start = seg_ver.z_cor_start
        z_end = seg_ver.z_cor_end

    if position == 'right': #1 = outerright vertical
        pt0 = rg.Point3d(seg_ver.x_cor_end -chosen_beam.width, 0, seg_ver.z_cor_start)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam.depth, seg_ver.z_cor_end)
        x_start = seg_ver.x_cor_end -chosen_beam.width
        z_start = seg_ver.z_cor_start
        z_end = seg_ver.z_cor_end

    # generate geometry in grasshopper
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    #generate substructure for stud
    picked_sub = select_substructure (chosen_beam, required_length, _Database_substructure, x_start, z_start, z_end)
    return select_structural(chosen_beam, waste, brep), picked_sub
```

```python
def outer_frame_horizontal(required_length, _Waste_wood_database, position):
    """Selects wood from db and generates geometry for horizontal frame"""
    #pick piece of wood from database for horizontal outerframe (top & bottom)
    chosen_beam = next(beam for beam in _Waste_wood_database if beam.length >= required_length)

    #determine additional waste with picked piece of wood
    waste = chosen_beam.length - required_length

    if position == 'bottom':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_start)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam.width, seg_ver.z_cor_start + chosen_beam.depth)

    if position == 'top':
        pt0 = rg.Point3d(seg_ver.x_cor_start, 0, seg_ver.z_cor_end -chosen_beam.depth)
        pt1 = rg.Point3d(seg_ver.x_cor_end, chosen_beam.width, seg_ver.z_cor_end)

    # generate geometry in grasshopper
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    return select_structural(chosen_beam, waste, brep)


def inner_studs(required_length, _Waste_wood_database, stud_position):
    """Selects wood from db and generates geometry for vertical frame"""
    #pick piece of wood from database for vertical outer frame (left & right)
    chosen_beam = next(beam for beam in _Waste_wood_database if beam.length >= required_length)
    waste = chosen_beam.length - required_length

    #generate bottom stud geometry
    pt0 = rg.Point3d(seg_ver.x_cor_start + stud_position, 0, seg_ver.z_cor_start)
    pt1 = rg.Point3d(seg_ver.x_cor_start + stud_position + chosen_beam.width, chosen_beam.depth, seg_ver.z_cor_start +
required_length)
    x_start = seg_ver.x_cor_start + stud_position
    z_start = seg_ver.z_cor_start
    z_end = seg_ver.z_cor_start + required_length
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    picked_sub = select_substructure (chosen_beam, required_length, _Database_substructure, x_start, z_start, z_end)
    return select_structural(chosen_beam, waste, brep), picked_sub


def bracing (height_segment, width_segment, beams_selected_bracing, _Waste_wood_database):
    """Selects wood from db and generates geometry for bracing"""
    #determine dimensions right triangle for pythagoras calculation
    bracing_width = width_segment - beams_selected_framing[0+fr_counter].width -
beams_selected_framing[1+fr_counter].width
    bracing_height = height_segment - beams_selected_framing[2+fr_counter].depth -
beams_selected_framing[3+fr_counter].depth
    bracing_length = round((bracing_width**2 + bracing_height**2)**0.5)

    #pick piece of waste wood from database for bracing
    chosen_beam = next(beam for beam in _Waste_wood_database if beam.length >= bracing_length)
    waste = chosen_beam.length - bracing_length

    #calculation
    tan = bracing_width / bracing_height
    y = math.degrees(math.atan(tan))
    corner = 90 - y
    extra_x = math.sin(math.radians(corner)) * (0.5 * chosen_beam.depth)
    extra_z = math.cos(math.radians(corner)) * (0.5 * chosen_beam.depth)

    pt1 = rg.Point3d(beams_selected_framing[0+fr_counter].width + seg_ver.x_cor_start - extra_x , 0, seg_ver.z_cor_end -
beams_selected_framing[3+fr_counter].depth - extra_z)
    pt2 = rg.Point3d(seg_ver.x_cor_end-beams_selected_framing[1+fr_counter].width - extra_x , 0, seg_ver.z_cor_start +
beams_selected_framing[2+fr_counter].depth - extra_z)

    if tan < 1:
        box_1 = rg.Point3d(0,0,0)
        box_2 = rg.Point3d(chosen_beam.depth, chosen_beam.width, -bracing_length)
        zaxis = pt1-pt2

    if tan >= 1:
        box_1 = rg.Point3d(0,0,0)
        box_2 = rg.Point3d(chosen_beam.depth, -chosen_beam.width, bracing_length)
        zaxis = pt2-pt1
```

```python
    plane = rg.Plane(pt1, zaxis)
    box = rg.BoundingBox(box_1,box_2)
    bracing_geo = rg.Box(plane, box)
    return select_structural(chosen_beam, waste, bracing_geo)


def intersection_formula (x_position, height_segment):
    """Formula to determine height of the studs"""
    #determine the formula of the bracing so the length of the bottom and top stud can be termined on every position
along the slope
    a = (height_segment - beams_selected_framing[2+fr_counter].depth - beams_selected_framing[3+fr_counter].depth) /
(width_segment- beams_selected_framing[0+fr_counter].width - beams_selected_framing[1+fr_counter].width)
    b = height_segment - beams_selected_framing[2+fr_counter].depth - beams_selected_framing[3+fr_counter].depth
    y = -a * x_position + b
    length_bottom_stud = round(y + beams_selected_framing[2+fr_counter].depth)
    return length_bottom_stud


def geo_bottom_stud (chosen_bottom_stud, bottom_stud_length, stud_position):

    #generate bottom stud geometry
    pt0 = rg.Point3d(seg_ver.x_cor_start + stud_position, 0, seg_ver.z_cor_start)
    pt1 = rg.Point3d(seg_ver.x_cor_start + stud_position + chosen_bottom_stud.width, chosen_bottom_stud.depth,
seg_ver.z_cor_start + bottom_stud_length)
    x_start = seg_ver.x_cor_start + stud_position
    z_start = seg_ver.z_cor_start
    z_end = seg_ver.z_cor_start + bottom_stud_length
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    picked_bot_sub = select_substructure (chosen_bottom_stud, bottom_stud_length, _Database_substructure, x_start,
z_start, z_end)
    return brep, picked_bot_sub


def geo_top_stud (chosen_top_stud, top_stud_length, stud_positiont):

    #generate top stud geometry
    center_align = (chosen_bottom_stud.width - chosen_top_stud.width) / 2
    pt0 = rg.Point3d(seg_ver.x_cor_start + stud_position + center_align, 0, seg_ver.z_cor_start + bottom_stud_length)
    pt1 = rg.Point3d(seg_ver.x_cor_start + stud_position + center_align + chosen_top_stud.width, chosen_top_stud.depth,
seg_ver.z_cor_end)
    x_start = seg_ver.x_cor_start + stud_position + center_align
    z_start = seg_ver.z_cor_start + bottom_stud_length
    z_end = seg_ver.z_cor_end
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    picked_top_sub = select_substructure (chosen_top_stud, top_stud_length, _Database_substructure, x_start, z_start,
z_end)
    return brep, picked_top_sub


def select_substructure (beam_infill, required_length, _Database_substructure, x_start, z_start, z_end):
    """Selects wood from db and generates geometry for substructure"""
    beam_pair = []

    #determine required depth for the substructure piece so that the complete substructre is alligned
    depth_required = _Depth_wall_element - beam_infill.depth
    length_required = required_length

    #picks the option to complete the substructure with 1 piece
    suitable_beams = iter([beam_sub for beam_sub in _Database_substructure if beam_sub.depth == depth_required and
beam_sub.length >= length_required])
    chosen_beam = next(suitable_beams)

    #temp list with all the possible pieces that are longer then min required length and smaller then the required
length.
    beams_selected_temp = [beam.length for beam in _Database_substructure if
    beam.length >= _Min_length_substructure_piece and beam.length < length_required and beam.depth == depth_required]

    #looks for all the options where the total length of two pieces is smaller then thet total length of 1 piece.
    for numbers in itertools.combinations(beams_selected_temp,2):
        if sum(numbers) >= length_required and sum(numbers) < chosen_beam.length:
            beam_pair.append(numbers)

    #if there is no combination of two pieces, the single piece is picked
    if beam_pair == []:
        waste = chosen_beam.length - length_required
        pt0 = rg.Point3d(x_start, beam_infill.depth, z_start)
        pt1 = rg.Point3d(x_start+chosen_beam.width, beam_infill.depth+chosen_beam.depth, z_end)
        box = rg.BoundingBox(pt0, pt1)
        brep = box.ToBrep()

        return [select_structural(chosen_beam, waste, brep)]

    #otherwise the combination that generates the least waste is selected
    else:
        beam_pair.sort(key=sum)
        beam_pair_1 = next(beam for beam in _Database_substructure if beam.length == beam_pair[0][0] and beam.depth ==
depth_required)
        beam_pair_2 = next(beam for beam in _Database_substructure if beam.length == beam_pair[0][1] and beam.depth ==
depth_required and beam.id != beam_pair_1.id)

        pt0 = rg.Point3d(x_start, beam_infill.depth, z_start)
        pt1 = rg.Point3d(x_start + beam_pair_1.width, beam_infill.depth+beam_pair_1.depth, z_start + beam_pair_1.length)
        box = rg.BoundingBox(pt0, pt1)
        brep_1 = box.ToBrep()

        pt0 = rg.Point3d(x_start, beam_infill.depth, z_start + beam_pair_1.length)
        pt1 = rg.Point3d(x_start +beam_pair_2.width, beam_infill.depth+beam_pair_2.depth, z_end)
        box = rg.BoundingBox(pt0, pt1)
        brep_2 = box.ToBrep()

        waste = (sum(beam_pair[0]) - length_required) / 2

        return select_structural(beam_pair_1, waste, brep_1), select_structural(beam_pair_2, waste, brep_2)


structural_infill_waste = []
substructure_waste = []
structural_infill_used = []
end_of_life = []

beams_selected_bracing = []
beams_selected_framing_hor = []
beams_selected_framing_ver = []
beams_selected_framing = []
beams_selected_studs = []

beams_selected_substructure = []

geometry_infill_booldifference = []
geometry_infill_horizontal = []
geometry_infill_vertical = []
geometry_infill_studs = []
geometry_infill_bracing = []
geometry_substructure = []

fr_counter = 0
br_counter = 0

if _Generate:
    for seg_ver in _Segments:
        width_segment = seg_ver.x_cor_end - seg_ver.x_cor_start
        height_segment = seg_ver.z_cor_end - seg_ver.z_cor_start

        picked_ver_frame = outer_frame_vertical(height_segment, _Waste_wood_database, 'left')
        _Waste_wood_database.remove(picked_ver_frame[0].selected_wood)
        structural_infill_waste.append(picked_ver_frame[0].waste)
        beams_selected_framing_ver.append(picked_ver_frame[0].selected_wood)
        beams_selected_framing.append(picked_ver_frame[0].selected_wood)
        structural_infill_used.append(picked_ver_frame[0].selected_wood.length)
        geometry_infill_vertical.append(picked_ver_frame[0].geometry)
        if height_segment > 1500:
            geometry_infill_booldifference.append(picked_ver_frame[0].geometry)

        for substructure in picked_ver_frame[1]:
            beams_selected_substructure.append(substructure.selected_wood)
            _Database_substructure.remove(substructure.selected_wood)
            geometry_substructure.append(substructure.geometry)
            substructure_waste.append(substructure.waste)

        picked_ver_frame_2 = outer_frame_vertical(height_segment, _Waste_wood_database, 'right')
        _Waste_wood_database.remove(picked_ver_frame_2[0].selected_wood)
        structural_infill_waste.append(picked_ver_frame_2[0].waste)
        beams_selected_framing_ver.append(picked_ver_frame_2[0].selected_wood)
```

```
        beams_selected_framing.append(picked_ver_frame_2[0].selected_wood)
        structural_infill_used.append(picked_ver_frame_2[0].selected_wood.length)
        geometry_infill_vertical.append(picked_ver_frame_2[0].geometry)
        if height_segment > 1500:
            geometry_infill_booldifference.append(picked_ver_frame_2[0].geometry)

        for substructure in picked_ver_frame_2[1]:
            beams_selected_substructure.append(substructure.selected_wood)
            _Database_substructure.remove(substructure.selected_wood)
            geometry_substructure.append(substructure.geometry)
            substructure_waste.append(substructure.waste)

        picked_hor = outer_frame_horizontal(width_segment, _Waste_wood_database, 'bottom')
        _Waste_wood_database.remove(picked_hor.selected_wood)
        structural_infill_waste.append(picked_hor.waste)
        beams_selected_framing_hor.append(picked_hor.selected_wood)
        beams_selected_framing.append(picked_hor.selected_wood)
        structural_infill_used.append(picked_hor.selected_wood.length)
        geometry_infill_horizontal.append(picked_hor.geometry)

        picked_hor2 = outer_frame_horizontal(width_segment, _Waste_wood_database, 'top')
        structural_infill_waste.append(picked_hor2.waste)
        beams_selected_framing_hor.append(picked_hor2.selected_wood)
        beams_selected_framing.append(picked_hor2.selected_wood)
        structural_infill_used.append(picked_hor2.selected_wood.length)
        geometry_infill_horizontal.append(picked_hor2.geometry)


        reuse = (height_segment - beams_selected_framing_hor[-1].depth - beams_selected_framing_hor[-2].depth +
width_segment) * 2
        end_of_life.append(reuse)

        if height_segment < 1500 or width_segment < _Ctc_distance:
            stud_step = 450
            stud_position = stud_step
            while stud_position < width_segment:
                #determine required length for bottom & top stud
                stud_length = height_segment
                picked_inner = inner_studs(stud_length, _Waste_wood_database, stud_position)

                _Waste_wood_database.remove(picked_inner[0].selected_wood)
                structural_infill_waste.append(picked_inner[0].waste)
                beams_selected_studs.append(picked_inner[0].selected_wood)
                structural_infill_used.append(picked_inner[0].selected_wood.length)
                geometry_infill_studs.append(picked_inner[0].geometry)

                for substructure in picked_inner[1]:
                    beams_selected_substructure.append(substructure.selected_wood)
                    _Database_substructure.remove(substructure.selected_wood)
                    geometry_substructure.append(substructure.geometry)
                    substructure_waste.append(substructure.waste)

                stud_position = stud_position + stud_step
                reuse = stud_length - beams_selected_framing_hor[-1].depth - beams_selected_framing_hor[-2].depth
                end_of_life.append(reuse)

            fr_counter += 4
            #inner_studs(stud_length, _Waste_wood_database, width_segment)

        else:
            picked_bracing = bracing(height_segment, width_segment, beams_selected_bracing, _Waste_wood_database)
            _Waste_wood_database.remove(picked_bracing.selected_wood)
            structural_infill_waste.append(picked_bracing.waste)
            beams_selected_bracing.append(picked_bracing.selected_wood)
            structural_infill_used.append(picked_bracing.selected_wood.length)
            geometry_infill_bracing.append(picked_bracing.geometry)

            stud_position = _Ctc_distance

            while stud_position < width_segment:
                #determine required length for bottom & top stud
                bottom_stud_length = intersection_formula (stud_position, height_segment)
                top_stud_length = height_segment - bottom_stud_length
                margin = 0
                controle = False

                while controle == False:
                    suitable_bottom_studs = iter([beam for beam in _Waste_wood_database if beam.length >=
bottom_stud_length + margin])
```

```
                    chosen_bottom_stud = next(suitable_bottom_studs)

                    #restrictions top studs based on bottom stud, top stud cannot be wider or deeper then bottom stud
                    suitable_top_studs = iter([beam for beam in _Waste_wood_database if beam.length >= top_stud_length
                    and beam.width <= chosen_bottom_stud.width
                    and beam.depth == chosen_bottom_stud.depth
                    and beam.length < top_stud_length + 150
                    and beam.id != chosen_bottom_stud.id])

                    chosen_top_stud = next(suitable_top_studs, 0)

                    #if no top stud is suitable, a different bottom stud is selected
                    if chosen_top_stud == 0:
                        margin +=1

                    else:
                        controle = True

                        #determine waste bottom stud & remove bottom studs from database
                        waste = chosen_bottom_stud.length - bottom_stud_length
                        _Waste_wood_database.remove(chosen_bottom_stud)
                        structural_infill_waste.append(waste)
                        beams_selected_studs.append(chosen_bottom_stud)
                        structural_infill_used.append(chosen_bottom_stud.length)

                        picked_bot_stud = geo_bottom_stud (chosen_bottom_stud, bottom_stud_length, stud_position)
                        geometry_infill_studs.append(picked_bot_stud[0])
                        for substructure in picked_bot_stud[1]:
                            beams_selected_substructure.append(substructure.selected_wood)
                            _Database_substructure.remove(substructure.selected_wood)
                            geometry_substructure.append(substructure.geometry)
                            substructure_waste.append(substructure.waste)


                        #determine waste top stud & remove top studs from database
                        waste = chosen_top_stud.length - top_stud_length
                        _Waste_wood_database.remove(chosen_top_stud)
                        structural_infill_waste.append(waste)
                        beams_selected_studs.append(chosen_top_stud)
                        structural_infill_used.append(chosen_top_stud.length)

                        picked_top_stud = geo_top_stud(chosen_top_stud, top_stud_length, stud_position)
                        geometry_infill_studs.append(picked_top_stud[0])
                        for substructure in picked_top_stud[1]:
                            beams_selected_substructure.append(substructure.selected_wood)
                            _Database_substructure.remove(substructure.selected_wood)
                            geometry_substructure.append(substructure.geometry)
                            substructure_waste.append(substructure.waste)


                        stud_position = stud_position + _Ctc_distance
                        reuse = height_segment - beams_selected_framing_hor[-1].depth - beams_selected_framing_hor[-
2].depth - beams_selected_bracing[-1].depth
                        end_of_life.append(reuse)
            fr_counter += 4
            br_counter += 1

    #calculate the total waste of the structural infill + substructure
    total_waste = sum(structural_infill_waste) + sum(substructure_waste)

    total_infill = beams_selected_bracing + beams_selected_framing + beams_selected_studs
    total_infill_used = round(sum(structural_infill_used)/1000,2)
    total_waste_infill = round(sum(structural_infill_waste)/1000,2)
    total_infill_required =  round((sum(structural_infill_used)/1000)-(sum(structural_infill_waste)/1000),2)

    substructure_used = [wood.length for wood in beams_selected_substructure]
    total_substructure_used = round(sum(substructure_used)/1000,2)
    total_waste_substructure = round(sum(substructure_waste)/1000,2)
    total_substructure_required = round(sum(substructure_used)/1000 - sum(substructure_waste)/1000,2)
    total_infill_used_2 = sum(structural_infill_used)/1000
    total_substructure_used_2 = sum(substructure_used)/1000
    reuse_end_of_life = round(sum(end_of_life)/1000,2)

    total_volume = []

    #check to see if no piece of wood was selected twice from the database
    print('\n')
    print('structural_infill')
    total_infill.sort(key = operator.attrgetter('id'))
```

```
    for balk in total_infill:
        balk.print_data()
        volume = (balk.length / 1000) * (balk.width / 1000) * (balk.depth / 1000)
        total_volume.append(volume)

    #check to see if no piece of wood was selected twice from the database
    print('\n')
    print('substructure')
    beams_selected_substructure.sort(key = operator.attrgetter('id'))
    for balk in beams_selected_substructure:
        balk.print_data()
        volume = (balk.length / 1000) * (balk.width / 1000) * (balk.depth / 1000)
        total_volume.append(volume)

    total_volume_def = sum(total_volume)
    print(total_volume_def)
```

```python
"""Generates the geometry for the cladding and calculates the waste
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Generate: Boolean toggle, if True apply
        _Database_cladding: list with wood for cladding
        _Ctc_distance: centre to centre distance between studs in mm
        _Cladding_segments: list with dimensions and positions of segments for cladding

    Output:
        geometry_claddingl: list with breps for cladding
        total_cladding_waste: waste in meter for cladding
        total_claddding_used: meter of wood used for cladding
        total_cladding_required: meter of wood required for cladding
        wood_selected_cladding: list of wood selected for cladding
        """

import Rhino.Geometry as rg
import itertools
import math

class cladding:
    def __init__(self, selected_wood_clad, geometry_clad):
        self.selected_wood_clad = selected_wood_clad
        self.geometry_clad = geometry_clad

def cladding_geometry (required_length, current_height, piece_height, x_start, x_end, z_start, z_end, height_segment):
    """Selects wood from db and generates geometry for cladding"""
    picked_cladding = next(clad for clad in _Database_cladding if clad.height == piece_height and clad.length ==
required_length)

    pt0 = rg.Point3d(x_start, 250, z_start + current_height)
    pt1 = rg.Point3d(x_end, 250 + picked_cladding.depth, z_start + current_height + piece_height)
    if current_height + piece_height > height_segment:
        pt1 = rg.Point3d(x_end, 250 + picked_cladding.depth, z_end)
    box = rg.BoundingBox(pt0, pt1)
    brep = box.ToBrep()

    return cladding(picked_cladding, brep)


cladding_waste = []
geometry_cladding = []
wood_selected_cladding = []

if _Generate:
    for segment in _Cladding_segments:
        height_segment = segment.z_cor_end - segment.z_cor_start
        width_segment = segment.x_cor_end - segment.x_cor_start
        x_start = segment.x_cor_start

        cladding_selected_temp = []
        current_height = 0
        marge = 0

        #calculate the max allowed cladding pieces in x-direction
        max_number_pieces = int(math.floor(width_segment / (_Ctc_distance*2)))

        min_length_clad = 2*_Ctc_distance
        if width_segment < 2*_Ctc_distance:
            min_length_clad = width_segment

        #marge prevents the happening of infinite while loop
        while marge < 1000:
            if current_height > height_segment:
                break

            #start finding cladding with the highest height cladding
            piece_height = _Database_cladding[0].height

            #continue until it is tried with the lowest height cladding
            while piece_height >= _Database_cladding[-1].height:
                #generate list with all the cladding with the same height
                cladding_same_height = [clad.length for clad in _Database_cladding if clad.height == piece_height and
clad.length >= min_length_clad]
```

```python
                if current_height >= height_segment:
                    break

                #see if the total width of the wall can be covered with one piece of cladding
                for clad_len in cladding_same_height[:]:
                     if clad_len >= width_segment and clad_len <= width_segment + marge:
                        cladding_same_height.remove(clad_len)
                        waste = clad_len - width_segment
                        cladding_waste.append(waste)
                        picked = cladding_geometry (clad_len, current_height, piece_height, segment.x_cor_start,
segment.x_cor_end, segment.z_cor_start, segment.z_cor_end, height_segment)

                        wood_selected_cladding.append(picked.selected_wood_clad)
                        _Database_cladding.remove(picked.selected_wood_clad)
                        geometry_cladding.append(picked.geometry_clad)

                        current_height += piece_height

                if current_height >= height_segment:
                            break

                #see if the width of the wall can be covered with n pieces of cladding, to a maximum of calculated
max_number_pieces
                for x_pieces in range(2,max_number_pieces+1):
                    for numbers in itertools.combinations(cladding_same_height,x_pieces):
                        x_start = segment.x_cor_start
                        if current_height >= height_segment:
                            break
                        if sum(numbers) >= width_segment and sum(numbers) <= width_segment + marge:
                            check =  all(elem in cladding_same_height for elem in numbers)
                            if check:
                                for counter in range(0,x_pieces):
                                    if counter == x_pieces - 1:
                                        picked_1 = cladding_geometry (numbers[counter], current_height, piece_height,
x_start, segment.x_cor_end, segment.z_cor_start, segment.z_cor_end, height_segment)
                                        wood_selected_cladding.append(picked_1.selected_wood_clad)
                                        _Database_cladding.remove(picked_1.selected_wood_clad)
                                        geometry_cladding.append(picked_1.geometry_clad)
                                    else:
                                        picked_2 = cladding_geometry (numbers[counter], current_height, piece_height,
x_start, x_start + numbers[counter], segment.z_cor_start, segment.z_cor_end, height_segment)
                                        wood_selected_cladding.append(picked_2.selected_wood_clad)
                                        _Database_cladding.remove(picked_2.selected_wood_clad)
                                        geometry_cladding.append(picked_2.geometry_clad)
                                    x_start += numbers[counter]
                                    cladding_same_height.remove(numbers[counter])

                                waste = sum(numbers) - width_segment
                                cladding_waste.append(waste)
                                current_height += piece_height

                piece_height -= 10
            marge += 10

        #calculate the amount of waste
        cladding_used = [length.length for length in wood_selected_cladding]
        total_cladding_used = round(sum(cladding_used)/1000,2)
        total_cladding_waste = round(sum(cladding_waste)/1000,2)
        total_cladding_required = round((sum(cladding_used)/1000)-(sum(cladding_waste)/1000),2)
```

11. Statistics calculation for interface

```python
"""Transforms the waste wood data in a string for the exterior wall element tool
All the variables that start with an underscore and capital letter are direct inputs"""

total_infill_used = 'strucutral infill used: '+ str(_Total_infill_used) + 'm'
total_waste_infill = 'strucutral infill waste: '+ str(_Total_waste_infill) + 'm'
total_infill_required =  'structural infill required: ' + str(_Total_infill_required) + 'm'

total_substructure_used = 'substructure used: '+ str(_Total_substructure_used) + 'm'
total_waste_substructure = 'substructure waste: '+ str(_Total_waste_substructure) + 'm'
total_substructure_required =  'substructure required: ' + str(_Total_substructure_required) + 'm'

total_cladding_used = 'total cladding used: ' + str(_Total_cladding_used) + 'm'
total_cladding_waste = 'total cladding waste: ' + str(_Total_cladding_waste) + 'm'
total_cladding_required = 'total cladding required: ' + str(_Total_cladding_required) + 'm'

total_wood_used = 'Total wood used: ' + str(_Total_infill_used + _Total_substructure_used + _Total_cladding_used) + 'm'
total_waste = 'Total waste: ' + str(_Total_waste_infill + _Total_waste_substructure + _Total_cladding_waste) + 'm'

end_of_life = _End_of_life + _Total_substructure_used + _Total_cladding_used
end_of_life_txt = 'End of life reusable: ' + str(end_of_life) + 'm'
end_of_life_per = round(end_of_life / (_Total_infill_used + _Total_substructure_used + _Total_cladding_used) * 100)
end_of_life_waste = 100 - end_of_life_per

volume = bracing_geo = [piece_wood.GetVolume() for piece_wood in _Breps]
total_volume = round(sum(volume)/1000000)
total_volume_dm3 = 'Total volume: ' + str(total_volume) + ' dm3'
weight = round(500 * (total_volume/1000))
weight_kg = 'Total weight: ' + str(weight) + ' kg'
```

12. BooleanDifference

```python
"""Peformes a BooleanDifference operation on the breps
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Generate: Boolean toggle, if True apply
        _Bracing: list with breps used for bracing
        _Studs: list with breps used for the studs
        _Framing: list with breps used with vertical frame that also have bracing
        _Horizontal_frame: list with breps used for the horizontal frame
        _Vertical_frame: list with breps used for the vertical frame

    Output:
        bracing_geo: list with modified breps for bracing
        vertical_frame_geo: list with modified breps for the vertical frame
        studs_geo: list with modified breps for the studs
        next_step: Boolean toggle, allow next step to happen when boolean difference is complete
"""

import Rhino.Geometry as rg

if _Generate:
    bracing_geo = rg.Brep.CreateBooleanDifference(_Bracing, _Horizontal_frame, 0.01)
    bracing_geo = rg.Brep.CreateBooleanDifference(bracing_geo, _Framing, 0.01)
    vertical_frame_diff = rg.Brep.CreateBooleanDifference(_Vertical_frame, _Horizontal_frame, 0.01)
    vertical_frame_geo = rg.Brep.CreateBooleanDifference(vertical_frame_diff, bracing_geo, 0.01)
    studs_diff = rg.Brep.CreateBooleanDifference(_Studs, _Horizontal_frame, 0.01)
    studs_geo = rg.Brep.CreateBooleanDifference(studs_diff, bracing_geo, 0.01)

    #removes any unwanted geometry that is generated when the bracing sticks out of the wall element.
    bracing_geo = [brep for brep in bracing_geo if brep.GetVolume() > 100000]

    #This assures that the export or bake can only happens when the Boolean difference is complete
    next_step = True


else:
    next_step = False
```

13. Bake geometry in Rhino

```
"""Bakes the geometry to a rhino file
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Active: Boolean toggle, if True apply
        _Geometry: list with breps
        _Layer: layer where geometry needs to be baked in
"""
import rhinoscriptsyntax as rs
import scriptcontext as sc
import Rhino

if _Active:
    for brep in _Geometry:
        brep_id = brep
        sc.doc = ghdoc
        doc_object = rs.coercerhinoobject(brep_id)
        baked_geo = doc_object.Geometry
        attributes = doc_object.Attributes
        sc.doc = Rhino.RhinoDoc.ActiveDoc
        baked_geo = sc.doc.Objects.Add(baked_geo, attributes)

#remove the baked geometry when the toggle is off
else:
    sc.doc = Rhino.RhinoDoc.ActiveDoc
    objs = rs.ObjectsByLayer(_Layer)
    rs.DeleteObjects(objs)
```

14. Export 3dm files

```
"""exports each baked brep to a seperate 3dm file
    All the variables that start with an underscore and capital letter are direct inputs
    Inputs:
        _Active: Boolean toggle, if True apply
        _Bracing_geo: list with breps for bracing
        _Vertical_frame_geo: list with breps for the vertical frame
        _Horizontal_frame_geo: list with breps for the vertical frame
        _Studs_geo: list with breps for the studs
        _Substructure_geo: list with breps for the substructure
        _Cladding_geo: list with breps for the cladding
        _Beams_selected_framing_ver: list of wood selected for vertical framing
        _Beams_selected_framing_hor: list of wood selected for horizontal framing
        _Beams_selected_bracing: list of wood selected for bracing
        _Beams_selected_studs: list of wood selected for studs
        _Beams_selected_substructure: list of wood selected for substructure
        _Wood_selected_cladding: list of wood selected for cladding
        _Layer: layer where geometry needs to be baked in

    Output:
        id_list: list with all the id numbers of the exported wood
        toggle: Boolean toggle, allow next step to happen when export is complete
"""
import ctypes.wintypes
import rhinoscriptsyntax as rs
import scriptcontext as sc
import Rhino

def export_file (geometry, wood_list):
    """Bakes brep to Rhino and exports it to 3dm file"""
    counter = 0
    for brep in geometry:
        id_list.append(wood_list[counter].id)
        brep_id = brep
        sc.doc = ghdoc
        doc_object = rs.coercerhinoobject(brep_id)
        baked_geo = doc_object.Geometry
        attributes = doc_object.Attributes
        sc.doc = Rhino.RhinoDoc.ActiveDoc
        rs.UnselectAllObjects()
        baked_obj = sc.doc.Objects.Add(baked_geo, attributes)
        rs.ObjectLayer(baked_obj, _Layer)
        rs.SelectObject(baked_obj)
        filepath = save_dir + 'id_' + str(wood_list[counter].id)
        rs.Command('_-Export "'+ filepath + '.3dm" _Enter _Enter')
        counter += 1

id_list = []
toggle = False

CSIDL_PERSONAL = 5       # My Documents
SHGFP_TYPE_CURRENT = 0   # Get current, not default value
buf = ctypes.create_unicode_buffer(ctypes.wintypes.MAX_PATH)
ctypes.windll.shell32.SHGetFolderPathW(None, CSIDL_PERSONAL, None, SHGFP_TYPE_CURRENT, buf)

save_dir = buf.value + "\\Exterior wall element tool\\Exported 3dm files\\"

#export every geometry to a folder with the correlating ID number as file name
if _Export and _Active:
    rs.UnselectAllObjects()
    export_file(_Vertical_frame_geo, _Beams_selected_framing_ver)
    export_file(_Horizontal_frame_geo, _Beams_selected_framing_hor)
    export_file(_Studs_geo, _Beams_selected_studs)
    export_file(_Bracing_geo, _Beams_selected_bracing)
    export_file(_Substructure_geo, _Beams_selected_substructure)
    export_file(_Cladding_geo, _Wood_selected_cladding)

    sc.doc = Rhino.RhinoDoc.ActiveDoc
    objs = rs.ObjectsByLayer(_Layer)
    rs.DeleteObjects(objs)
    toggle = True
```

15. Remove selected pieces of wood from database

```python
""" Deletes used wood from SQL database.
    Inputs:
        _Apply: boolean toggle to start data the sciprt
        _Id_list: List with id numbers of the wood that needs to be deleted
    Output:
        output: conformation when data is removed.

"""


def delete_id(id_number):
    #deletes existing data in database
    cur.execute("DELETE FROM wood_dimensions WHERE wood_id = (%s)", (id_number,));
    conn.commit()
    print(str(id_number) + ' removed')

import psycopg2 as pg2
import random

if _Apply == True:
    conn = pg2.connect(database='hout', user='postgres', password='----')
    cur = conn.cursor()
    for id_number in _Id_list:
        delete_id(id_number)
```