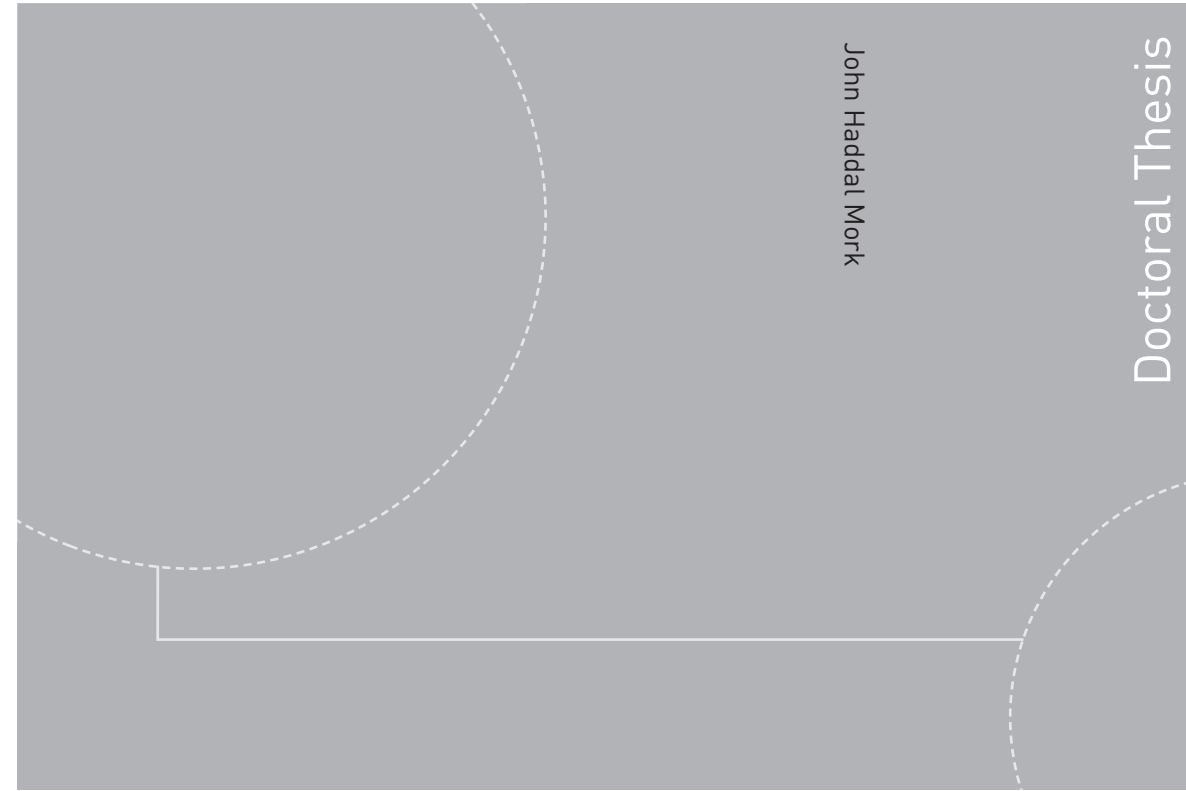


ISBN 978-82-326-4824-5 (printed version)
ISBN 978-82-326-4825-2 (electronic version)
ISSN 1503-8181



Doctoral theses at NTNU, 2020:238

John Haddal Mork

Parametric Timber Detailing

A parametric toolkit customized for detailing fabrication-ready timber structures

Doctoral theses at NTNU, 2020:238

NTNU
Norwegian University of
Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology

 **NTNU**
Norwegian University of
Science and Technology

 NTNU

 **NTNU**
Norwegian University of
Science and Technology

John Haddal Mork

Parametric Timber Detailing

A parametric toolkit customized for detailing fabrication-ready timber structures

Thesis for the degree of Philosophiae Doctor

Trondheim, August 2020

Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Architecture and Design
Department of Architecture and Technology

© John Haddal Mork

ISBN 978-82-326-4824-5 (printed version)

ISBN 978-82-326-4825-2 (electronic version)

ISSN 1503-8181

Doctoral theses at NTNU, 2020:238



Printed by Skipnes Kommunikasjon as

Parametric Timber Detailing

A parametric toolkit customized for detailing fabrication-ready timber structures

Author: John Haddal Mork
Supervisor: Bendik Manum
Co-Supervisor: Anders Rønnquist

Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology

Trondheim, August 2020

This thesis is dedicated to my Morfar, Johan Paul Haddal (1929-2016), and my Tante, Palma Irene Sandvik (1935-2016). They were both immensely supportive and loved to hear about my life and education. I am very sad that I did not have the chance to see their happy faces after telling them that I obtained my PhD.

Publications

Included publications

The thesis includes the following papers:

- I. Mork JH, Luczkowski M, Manum B, Rønnquist A. Toward Mass Customized Architecture. Applying Principles of Mass Customization While Designing Site-Specific, Customer-Inclusive and Bespoke Timber Structures. In: Digital Wood Design. Springer; 2019. p. 221–249. doi: 10.1007/978-3-030-03676-8
- II. Mork JH, Luczkowski M, JointSearch: Efficient parametric detailing preparation through user-defined and property based joint type filtering
- III. Mork JH, Luczkowski M, Manum B, Rønnquist A. Parametric timber toolkit: A timber tailored approach. In: Again, Golden Era of timber. Seoul: World Conference On Timber Engineering; 2018.

The thesis includes the following source code:

- IV. Mork JH, Toda Y, Izumi B, Luczkowski M, Hillersøy Dyvik S. CSDG-DDL/PTK: Reindeer 0.5 (Version v0.5). Zenodo. <http://doi.org/10.5281/zenodo.3567051>

Other scientific contributions

- V. Mork JH, Hillersøy Dyvik S, Manum B, Rønnquist A. Introducing the segment lath—a simplified modular timber gridshell built in Trondheim Norway. In World Conference on Timber Engineering; 2016.
- VI. Mork JH, Luczkowski M, Dyvik SH, Manum B, Rønnquist A. Generating timber truss bridges—examining the potential of an interdisciplinary parametric framework for architectural engineering. In IABSE Stockholm; 2016
- VII. Mork JH, Luczkowski M, Manum B, Rønnquist A. Conceptual structural design of shell structures in virtual reality. In: IABSE Symposium Report. International Association for Bridge and Structural Engineering; 2017. p. 157–158.
- VIII. Mork JH, Luczkowski M, Manum B, Rønnquist A. One algorithm, two timber bridges built in Orkdal, Norway. In: Forum Wood Building/Nordic Trondheim 17. Faculty of Architecture and Design Trondheim; 2017.

Abstract

This thesis develops a parametric toolkit for visual programming that is customised for detailing fabrication-ready timber structures.

Digital fabrication increases the domain of what is rational to manufacture, and algorithm aided design expands the domain of what is rational to design. Visual programming has made both technologies more accessible.

However, digital fabrication and algorithmic aided design remain technically demanding. By reviewing existing methods and tools related to parametric timber detailing, two main challenges are identified and chosen for further investigation. The first challenge concerns parametric detailing preparation in general and joint type identification in particular. The second challenge concerns parametric timber and the transfer from a parametric model to digital fabrication.

Within the framework of this thesis a parametric toolkit is developed for visual programming and customised for detailing fabrication-ready timber structures. The toolkit aims to reduce the time and algorithmic knowledge that are required to prepare a model for timber detailing and to transfer a detailed timber structure to digital fabrication. The toolkit is named Reindeer and is an open source plugin for Grasshopper3D

The thesis has special focus on two tools within the toolkit: JointSearch is an approach that enables the user to define the solution space of a joint type via one or multiple search criteria. TimberProcessingTools is an approach that enables the user to apply tools that mimic the physical processing of timber structures. The tool outputs a BTLx file, which is readable by most manufacturers.

The toolkit is developed by designing, manufacturing and building a series of case structures. The investigation chapter includes a presentation of Reindeer and the case structures. Three papers that discuss the framework, JointSearch and TimberProcessingTools are summarised.

The contribution of this thesis is two-fold. The main contribution this thesis makes to the field of practitioners is that the toolkit is open source and makes partly existing methods more accessible. The main academic contribution this thesis makes to the field of parametric detailing is the JointSearch methodology. First, JointSearch is independent of the global topology and reduces the complexity of a parametric model. Second, the method has proven to be both efficient, flexible and precise while filtering. Last, the method is independent of materials and applications.

By conducting this study, I am convinced that AAD and digital fabrication is a key to making our building industry sustainable while maintaining the flexibility of design. Reindeer is a small step towards further democratising the use of AAD and digital fabrication while designing and detailing timber structures.

Samandrag

Digital fabrikasjon utvidar domenet for kva som er rasjonelt å byggje, og Algoritme Assistert Design (AAD) utvidar domenet for kva som er rasjonelt å designe. Parallelt sørgjer visuell programmering for at teknologien vert stadig meir tilgjengelig.

Når det er sagt, er teknologien fortsatt teknisk krevjande. Gjennom ei analyse av ledande metodar og verkty tilknytt parametrisk detaljering av trekonstruksjonar, har det vorte identifisert to hovudutfordringar for vidare utforskning. Den fyrste utfordringa handlar om førebuing til parametrisk detaljering generelt, men spesielt omhandlar det å enkelt kunne identifisere ulike type knutepunkt i ein konstruksjon. Den andre utfordringa handlar om overgangen frå ein parametrisk modell til digital fabrikasjon.

Denne avhandlinga utviklar ei parametrisk verktøykasse for visuell programmering som er skreddarsydd for å detaljere fabrikkasjonsklare trekonstruksjonar. Målet med verktøykassa er å forenkle førebuing av ein parametrisk modell, samt overgangen frå ein parametrisk modell til digital fabrikasjon.

Den utvikla JointSearch-metodikken gjer at brukaren kan definere eit knutepunkt sitt mogleighetsrom gjennom eit eller fleire søjekriterier. TimberProcessingTools-metodikken gjer at brukaren kan bruke digitale designverktøy som imiterer korleis ein fabrikkere trekonstruksjonar. Desse verktøya produserer også ei BTLx-fil – eit filformat som dei fleste produsentar klarer å lese. Verktøyet er døypt Reindeer og er ein Open-Source plugin for Grasshopper3d.

Verktøykassa er utvikla gjennom å designe, fabrikkere og byggje ei rekkje konstruksjonar. Ein presentasjon av desse, saman med tre artiklar og ein eigen presentasjon av Reindeer, utgjer Utforskningskapittelet.

Bidraget frå dette studiet er todelt. Bidraget til profesjonen er eit fritt tilgjengelig, gratis verkty for å detaljere trekonstruksjonar parametrisk. Det akademiske bidraget handlar primært om JointSearch-metodikken. For det fyrste: JointSearch er uavhengig av global topologi og gjer den parametriske modellen meir robust. For det andre: filtreringa av knutepunkt er både fleksibel og presis. Til slutt: Denne metoda er materialuavhengig og kan ha andre bruksområder.

Etter å ha gjort denne studia, er eg viss på at AAD og digital fabrikasjon er ein viktig nykjel for å gjere byggeindustrien vår meir berekraftig. Dette samtidig som me beheld formgjevingsfleksibiliteten. Reindeer er eit lite steg i retning å gjere digital fabrikasjon og AAD meir tilgjengelig.

Acknowledgements

I have tears in my eyes while writing this section (true). I may also have had tears in my eyes while writing other sections for reasons other than the notion that the content touched me. This study would not have been possible without comprehensive support and active collaboration from friends, family, and colleagues. This acknowledgement is relatively comprehensive and personal, but sharing my gratitude is important.

From the day I started my Architectural education, the people at NTNU have been remarkably supportive. I have always been allowed to create a unique path for myself. Thank you Steffen Wellinger for helping me become a reflective student, asking “Hva vil dere?”, saying “Don’t let the school get in the way of your education” and always believing in us. Thank you Finn Hakonsen for teaching me about architecture and for our conversations about values, work-life balance, and parametric tectonics (!). Although I have always had a technological interest and perspective, Hans Skotte and Elena Archipovaite have fed me with important knowledge and values from other branches of architecture. Pasi Aalto, thank you for being the leading commander in our underground parametric revolution. For not giving up when the system is slow, buying nice CNC equipment at the workshop, having in-depth discussions and involving me in exciting research projects. Jörg Siegfried Schauer, Pete Rose and not least Sverre Halvorsen at the Lukas-Workshop: You are amazing. Thank you for illegally letting us in at the workshop during weekends, for showing up on a Saturday teaching us how to weld, for forgiving me when I forgot (still forget) to clean up properly. Thank you for sharing valuable insights about woodworking. I am also grateful for the administrative staff and the PhD-board. I have always wanted to do my own stuff, and I know that it has been challenging from time to time. Special mention to Jorun Schanke Olsen and Brit Gulvåg.

Thank you, Rallar. Silje Ruud, Kristine Øvstebø, Eiliv Andreas Myren Ribe, Steinar Hillersøy Dyvik, Anders Gunleiksrud, Kristin Solhaug Næss, Sebastian Østlie and all my friends joining our student workshops: I am immensely grateful for the way we created our education. My partner in crime, Anders Gunleiksrud you have been essential for this PhD. Your carpentry skills, execution capabilities, design skills, friendly attitude, and reliability have made our naive projects realized. Thea Hougsrud Andreassen: I am incredibly grateful that you joined Rallar. For turning the then amateurish Rallar into a professional company. Thank you for adding remarkable stubbornness and design skills to our team. To both of you: Thank you for being stupidly patient, working 24/7, and allowing me to be rude when I am tired: just to let me finish this PhD.

None of these Rallar projects that have been important for this study would have been possible without three brave persons. Firstly, my fantastic Tante Ann: for naively saying yes when I asked her if we could draw her boathouse and afterwards build it together with 21 inexperienced architecture students. Secondly, a great thanks to Susanne Saue and Nadja Sahbegovic, Trondheim Kommune, for letting us design and build Piren and the Gridshell.

Collaborations have been vital in this study. Three external collaborations have been essential. Firstly, a big thanks to Moelven Limtre for sharing drawings, machine data, schedule, and not least knowledge. Special thanks to Åge Holmestad, Daniel Barosso, Trond Egil Nyløkken, and Rune Abrahamsen. Secondly, Johannes Lipphart, Ivar Lillery, and Orkdal Kommune were exemplary when they wanted to implement their bridge project in my PhD-study. Thank you for being patient. Thirdly, Nikken Sekkei has been essential contributors to developing the plugin. Bunji Izumi, Toda Yuto, Anders Rod and Xuhao Lin: thank you for your hospitality and great willingness of sharing and developing knowledge. Additionally, I would like to thank Martin Antemann at Blumer Lehman for two times, spending a whole

day sharing his knowledge and showing me their factory facilities. I would also like to share my gratitude for the support given by Innovation Norway while developing the Orkla Bridge Project.

My beautiful and geeky family. I want to thank my parents for raising me to become stubborn, for always supporting me, listening to my troubles and multiple times stop me when I was about to quit my PhD. I want to thank my mom, Elin-Grethe Haddal, for sharing Creativity-Optimization-Math-Genes and, for always liking Rallar's Facebook-posts, for being patient and educational and for sharing many sorts of crafts. I want to thank my dad, Ivar Mork. I am sorry for throwing away and destroying all your carpentry tools. But I am most of all, grateful for our projects, visionary discussion and our visits to the furniture factory. Thank you for letting me believe that we were actually going to build the tower we planned on top of our house.

I want to thank my sisters. For being role models, supportive, creative, and geeky. Thank you, Silja: for showing that there are better ways than the conventional way. For chairing me up and always wanting to know how my PhD was going. Thank you Ingvild: for our inspiring discussions about creativity and work habits. And for telling me that if I chose to specialize in 3D-modelling, "I would laugh all the way to the bank" (I still do not laugh at a bank, and laughing at the bank was never a motivation.). Dear Tante Palma: You were the first one to tell me that I was probably becoming an architect. You inspired me.

I would like to thank my Mormor and Morfar. I am infinitely thankful that I was able to grow up on a farm, ten meters away from your home. You have always been there for me, supported me and given me values. Mormor: you were my first partner in crime: together we built tunnels, moved snowballs uphill, we constructed railways and designed lego houses. Morfar: You always loved to share your knowledge and wisdom. I will always try to live up to your integrity and work ethics.

I also want to thank a couple of geeky, but non-architecture, engineering friends: John Martin Kleven Godø, Thomas Gjerde, Jostein Furseth and Erlend Erdal Christiansen. They have all been supportive and interested in my project. At the same time, they have been sharing impressive projects they are developing. The conversations have given me essential perspectives from other professions.

I have been lucky to be part of a young, growing research group called Conceptual Structural Design Group (CSDG). In this group, Marcin Luczkowski, Steinar Hillersøy Dyvik, and I represented the young, novice generation. Our discussions, fights, inspirational travels to conferences, jogs, Petters kebab-pizzas, and sushi-dinners have a fundamental piece in my study. Intellectually, but not least, helping me get through four hard years of becoming a researcher. Marcin: It has been a great journey. From originally you believing that architects only deal with colours, and from me believing that engineers only understand equations, we have both changed a lot. Now, I sometimes feel that you think too much like an architect, and I think too much like an engineer – But I do hope it is a sign of improved understanding. Steinar: there are many aspects of this PhD were I owe you a great thank. First of all, you are my best friend. You know what strings to pull when I am depressed. Equally important, you have understood what strings to pull when I am too confident. You have encouraged me to quit my PhD (because you knew that would make me not quit my PhD). You have been the one person I can discuss parametric modelling, architecture, and PhD-life crises. You have been an essential part of many of the structures presented. Lastly, you have a father-in-law, Tore Johan Øvstebø, that (again) stopped me from quitting my PhD. He asked: "John, are you not being a coward when quitting your PhD?". Thank you, Tore Johan!

Ever since the first years of the architecture study, I have been having interesting discussions with Bendik Manum. Designing imaginary gridhells, Piren, and a real-scale gridshell would not be possible

without intellectual, financial, and lobbying support from Bendik. From there, I was lucky to be picked out as his PhD candidate. Bendik: we have had our fights, but I am glad that you have always been direct, precise, and honest. You have never chosen any short cuts, never telling me the answer, even though you knew it. Instead, you have left me frustrated, waiting for me to realize a solution myself. At the same time, you manage to share your knowledge. The way you teach is demanding but still excellent. Your educational style has turned me into what I hope is an independent and robust researcher. I also want to thank my Co-Supervisor Anders Rønquist: I have had great use of your positive attitude and intelligent feedback. My sprint towards finalizing this study has been dependent on your pep-talks. Further, Gunnstein Thomas Frøseth helped me raise my research to a new level. Lastly, I am thankful for the support I have gotten from Nathalie Labonnote. All though you are not paid to be my supervisor, you have still invested time in listening, supporting and giving highly intellectual guidance on my PhD's direction

Contents

Publications	3
Included publications.....	3
Other scientific contributions	3
Abstract.....	4
Samandrag	5
Acknowledgements	6
Contents.....	9
I. INTRODUCTION AND BACKGROUND.....	12
1. Introduction	13
1.1. Parametric timber detailing	15
1.2. Objective	20
2. Content and structure of the thesis	21
3. Scope	22
4. Collaborations	23
5. Software Platform	25
5.1. Parametric platform.....	25
5.2. An Open Source Plugin for Grasshopper 3D.....	27
6. State-of-The-Art: Relevant tools and methods.....	29
6.1. DesignToProduction: Modelling strategy	29
6.2. Front Inc.: Parametric Building Information Generation.....	29
6.3. Buro Happold: Identify similar connections	30
6.4. CITA and Innochain.....	30
6.5. Topologic: Non-Manifold Topology (NMT).....	31
6.6. Rhino.Inside for Revit.....	32
6.7. Timber Plate Shell Structures.....	32
6.8. Transferring a digital design to fabrication	33
6.9. Summary	36
7. Methodology	37
7.1. Research Activities.....	37
7.2. Activity configurations.....	39
7.3. Summary	41

II.	INVESTIGATIONS	42
8.	Reindeer – Grasshopper-plugin	44
8.1.	Workflow	44
8.2.	Step 1: Defining elements	46
8.3.	Step 2: Assemble the structure, define the geometric relations	50
8.4.	Step 3: Structural analysis	51
8.5.	Step 4: JointSearch and JointOutput	52
8.6.	Step 5: Detailing the elements (and nodes)	59
8.7.	Step 6: Processing assembly (Generate NURBS or CAM output)	61
8.8.	Step 7: Manufacture and build	62
9.	Case-Structures	63
9.1.	Orkla Bridges	65
9.2.	Printshell	79
9.3.	Water Ramp	89
9.4.	Log House	101
9.5.	Smaller tests	108
10.	Papers	111
10.1.	Paper I: Framework and basic explanation of concepts	112
10.2.	Paper II: JointSearch	114
10.3.	Paper III: TimberProcessingTools	115
10.4.	Paper III Addition	Error! Bookmark not defined.
III.	DISCUSSION, CONCLUSION AND FURTHER WORK	117
11.	Discussion	118
11.1.	A specialist’s tool or a mainstream tool?	122
11.2.	Redefining similarity	122
11.3.	Consistent detailing	123
11.4.	Control	123
11.5.	Productivity, Emissions, and Waste	124
12.	Conclusion	126
13.	Further work	127
	References	128
IV.	APPENDICES	138
	Papers	139

I. INTRODUCTION AND BACKGROUND

1. Introduction

This thesis develops a parametric toolkit for visual programming that is customised for detailing fabrication-ready timber structures. The toolkit aims to reduce the time and algorithmic knowledge that are required to prepare a model for timber detailing and transfer a detailed timber structure to digital fabrication.

The global building industry has lower productivity than other industries [1]; a UN report reaffirms that buildings and construction account for more than 35% of the global final energy use and nearly 40% of energy-related CO₂ emissions [2]. The European Union (EU) states that construction and demolition waste (CDW) account for approximately 25-30% of all waste generated in the EU[3]. To prevent a global catastrophe, we need to develop better processes and tools for designing, planning and building our built environment.

Arguably, what we build in the future must be efficiently designed, planned and built. Further, structures must contribute to reducing carbon emissions and CDW. To ensure durable structures, we must also focus on users, functions, climate and context. Thus, our future building industry will be dependent on both flexible and efficient methods and tools.

Digital fabrication and algorithmic aided design (AAD) has the potential to revolutionise how we design, plan and build structures. The age of hand-making offered variations and flexibility, and the age of mass-production offered efficiency. Conversely, the digital age is the first era that offers both flexibility and efficiency while sustaining the precision introduced by mechanical manufacturing. Mario Carpo states *“Hand-making begets variations, and so does digital making; but the capacity to design and mass-produce serial variations (or differentially) is specific to the present digital environment.”*[4]

Digital fabrication and AAD introduce opportunities that the building industry cannot resist. However, current limitations hinder the parametric revolution. The next sections highlight both the opportunities and the limitations related to digital fabrication, AAD and parametric thinking.

Digital fabrication

Digital fabrication and computer numerical control (CNC) machines expand the domains of rational manufacturing. Not dependent on jigs and manual labour, CNC machines are universal[5] and can produce variations at no extra cost[6]. This mode of production is referred to as nonstandard seriality[4] and substantiates a mass-customised building industry[7].

All variations of a building element need to be accurately modelled when aided by digital fabrication. A digitally fabricated structure requires a digital twin. Planning for digital fabrication contrasts notational drawings that are aimed at manual labour, where principal sections, plans and 2D details are sufficient.

Algorithmic Aided Design

The power of digital fabrication is limited if it is not combined with AAD. Mario Carpo problematises the first digital turn: *“As digital fabrication processes invite endless design variation (within technical limits), and promise to deliver them at no extra cost, the question inevitably arises as to who is going to design them all.”*

The second digital turn integrates AAD. Robert Aish refers to the same tendencies as “the design computation era”[8]. Traditional CAD and building information modelling are digital emulations of traditional drawing boards[9]. By applying traditional CAD, each building element must be manually modelled, and the flexibility of digital fabrication is not easily exploited. Conversely, AAD better exploits the natural methods of computers and is crucial to digitally expand the domain of what is rational to model digitally. The digital aspect of nonstandard seriality is achieved by building algorithms that automatically generate geometry based on its custom inputs.

Visual programming

AAD emerged when architects and other designers started to hack the programming interfaces (APIs) of the CAD application[10]. These APIs were script-based. Thus, the flexibility of utilising parametric modelling was limited to technically talented designers. The threshold of writing codes to make architecture was too vast, and therefore, in the early 2000s, several visual programming software were developed. Generative Components (2003)[11] and Grasshopper 3D (2007)[12] have been significant. Autodesk subsequently developed Dynamo Studio[13]. Visual programming enables the user to build complex algorithms by using schema-based interfaces and pre-built functions and removes the need to know a programming language.

Parametric design thinking

Robert Aish, who is the inventor of Generative Components, claims that computational design differs from conventional design as the designer is not drawing the geometry but rather models a system that generates geometry. He exemplifies this claim by explaining how geometry is constructed[14]: *“With this understanding, we will have the opportunity to build “long chain” dependencies which will create interesting geometric configurations. [...] What becomes apparent is that we are not designing the geometry of the artefact, but rather we are constructing a “control rig”, some geometry that will never be built or seen, but which indirectly controls what will be constructed and experienced. “*

AAD requires not only a new skill set but also a new mindset. This mindset is referred to as Parametric Design Thinking (PDT)[15] and is how we design. Regardless of visual programming, if applying parametric modelling, the user is required to abstract his or her design into a logic that is implemented in an algorithm.

Complexity of parametric detailing

AAD and digital fabrication represents an opportunity to improve how we design and detail structures. These technologies are likely to be an important answer to make our building industry sustainable while maintaining the flexibility of what we design. However, AAD and digital fabrication has not been fully democratised and is too algorithmically demanding for most designers.

Setting up a parametric model, the “control jig” is a complex task and is one of the reasons that a parametric modeller is required *“to be part designer, part computer scientist, and part mathematician”*[9]. DesignToProduction claims that *“The modelling and detailing process is about creating and maintaining relations as much as geometry”*[16]. Further, novice parametric modellers often make the *“mistake of creating the thing itself, skipping the importance of creating the conceptual framework for their design”*[17]. These quotes relate to the complexity

of implementing topological relations in a parametric model. Topology, when in relation to parametric modelling, can be defined as “*a conceptual regulating skeleton that allows the development of more complex and detailed design solutions*”[17]

Well-crafted details are essential for designing a successful structure. When designing details, topologic relations are extremely important. Detailing involves shaping the relations between two building elements. To detail a joint, precise information about the node and its elements is required. When detailing parametrically, the algorithm needs to know what joint is to be detailed.

As long as parametric detailing is heavily about creating and maintaining relations and requires a designer to have multiple professions, the power of parametrically detailed structures will be restricted to high-end projects. Innovators that push the limit will always exist, but the real change occurs when the majority starts to apply the technology. Thus, a software capability gap exists between conventional CAD/BIM and pioneering AAD.

From these discussions, we can conclude that there is an acute need for a tool of intermediate complexity that utilises the power of AAD and digital fabrication while detailing structures. To limit the scope of this thesis, timber structures have been chosen for further investigation. However, the developed methods are adaptable for other materials.

The use of timber in buildings and structures has radically increased. Timber is a sustainable building material[18], strong compared with its weight[19], and has proven to be a suitable material for industrial and flexible prefabrication[20]. Designing, planning and building timber structures is a complex task that requires a team of multiple disciplines, including architects, structural engineers and manufacturers.

Before presenting the objectives of the tool, the following sections elaborate on the specific challenges related to parametric detailing of fabrication-ready timber structures.

1.1. Parametric timber detailing

The challenges are divided into two topics: Digital Timber Fabrication and Parametric Detailing Preparation.

Digital Timber Fabrication

One aspect of digital timber design is the distinctive characteristics of timber and manufacturing. First, timber is an anisotropic material that substantially varies in strength. Timber is dependent on fibre-direction, available cross-sections, and timber-specific properties. Second, timber is primarily crafted by subtracting material. A craftsman starts with a timber blank and then transforms the blank into a building component using mills, saws, and drills to subtract material. Understanding these constraints and logics are essential to design well-crafted timber structures.

A challenge is that the rationality constraints of digital design are significantly different from the above rationality constraints of fabricating a timber component. Unfortunately, while designing digitally, following the digital design’s rationality constraints is easier. As an example, if modelling a beam with a hole, drawing a rectangle and a circle and then extruding the 2D shape into a 3D object is more convenient. This approach contradicts modelling a box

(which represents the blank) and then using a cylinder to subtract the hole. The latter method is more similar to the reality of timber manufacturing. Refer to Figure 1.

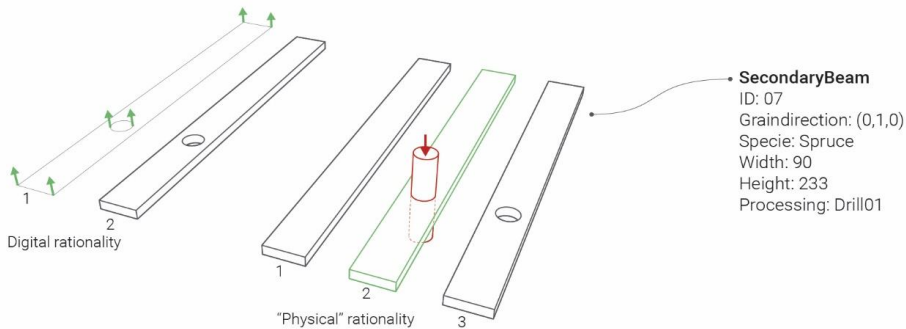


Figure 1 Parametric timber is concerned with implementing timber specific properties and ensuring that a design can be fabricated without remodelling the geometry in the CAM software.

This methodology gap may produce designs that are not buildable, and the gap affects the quality of the geometry transferred from design to manufacturing. Even if a 3D model is accurate and transferred via a precise 3D format, manufacturers are at risk of remodelling the geometry: The CAM software may not be able to identify the original blank and its timber processings. If a component is relatively simple, the CAM software is likely to succeed in translating the geometry into fabrication processings. This method is referred to as feature recognition. However, if the component is more complex, the CAM software may not be able to distinguish the original blank and subtractive processings.

BTL/BTLx is a file format that aims to solve the transfer of timber designs. Instead of storing the 3D geometry of the final component, the file stores both the original blank and the timber processings. Thus, transfer from design to fabrication is simplified. However, a plugin named Woodpecker is the only parametric tool that enables BTL export. This plugin enables drilling, cut, slot, pocket, and free contour[21] but is not integrated with the complete design process.

Although solutions exist, transfer from design to manufacturing is a significant challenge in the timber manufacturing industry. Marika Makkonen states that “compatible information system is lacking between different organisations, particularly in wood construction value chains.” [22] Further, she writes that design is often transferred from architects to manufacturers using 2D drawings. Manufacturers must redraw a design to make it compatible with CAM[23], which leads to redundant work and an increased probability of committing errors.

Tobias Schwinn supports the same perspective. In the book *Advancing Wood Architecture*[20], which presents state-of-the-art timber structures that are algorithmically designed, Schwinn claims that a limiting factor for collaboration is data translation among architectural design, structural engineering and timber fabrication. Further, Schwinn states that re-modelling and additional work dominates the transfer from CAD to CAM[20].

Based on these discussions, the transfer from a parametric model to digital fabrication is the main parametric timber-related challenge chosen for further investigation in this thesis.

Parametric Detailing Preparation

Regardless of the design technology, the following steps constitute a possible workflow when designing a timber-structure:

- 0) model a global structure using the centre lines
- 1) assign structural/geometric properties to the centre lines
- 3) analyse the structure
- 5) detail the structure's joints
- 7) manufacture and build the structure

As the numbers reveal, this workflow requires additional steps if it is parametrically conducted. From abstract steps, the process is converted to actual steps in the “control-rig” from a global model to manufacturing. To make a continuous chain, an algorithm must automate all the transfers and interpretations that are originally made by a human. These steps are 2) Generate joints, 4) Assign joints to joint types and 6) Process the model:

Step 2, Generate joints: In this thesis, a joint is defined as a node and its elements. Such relation can easily be identified by trained humans. However, to ensure a continuous workflow, a parametric model needs a system to identify and generate joints. Figure 2 illustrates the geometric relation.

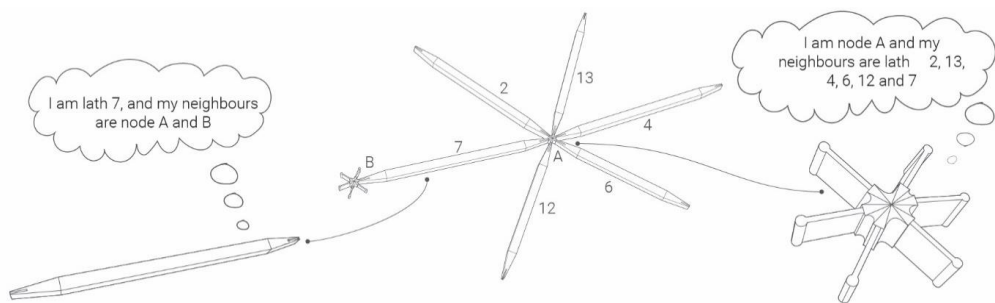


Figure 2: A joint and its relations. A joint is a node and its elements. By creating this relation, the node knows its elements and the element knows its nodes.

Step 4, Assign joints to joint types: A structure seldom contains only one joint type—one type of detailing logic. This step pertains to assigning joints to their types. If manually modelled, this step seems very easy. A trained eye intuitively spots what joints belong to the same joint type. The separation logic varies among projects, but the common defining parameters are the function of the structure, topology of the joint, number of elements, internal angles of elements and structural loads.

Although a structure consists of thousands of unique joints, the number of principally different joint types is likely to be manageable. La Seine Musicale, which is a timber gridshell, had 2798 unique joints but only eight main joint types[16,24]. By developing an algorithm-aided parametric system for each joint type, the types' instances can be automatically generated based on general inputs that are valid for all instances and unique inputs that are based on the shape of the instance. The size of a joint type's possible solution space is dependent on the developer. The parametric system of most joint types will enable some degree of geometric flexibility.

However, the size of the solution space and which parameters are flexible or locked are project dependent.

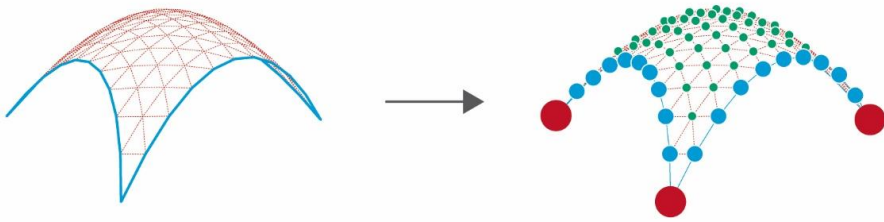


Figure 3: Before detailing, joints must be assigned to their joint type. Interior elements are red, edge elements are blue. The shell in this figure distinguishes its joint types by interior nodes (green), edge nodes (blue) and foundation nodes (red).

Based on these descriptions, a parametric detailing workflow may be described as follows:

- 0) **Global Model:** A global model, which is expressed using centre-curves, is manually or parametrically modelled.
- 1) **Element Generation:** Geometric information attributes the curves and converts them into timber elements. Cross-section, material properties and element names are common required attributes.
- 2) **Assembly:** The assembly generates all joints (all nodes and their elements).
- 3) **Structural analysis:** The analysis determines whether the global model is feasible.
- 4) **Joints type identification:** Project-specific logic assigns the joints to their joint types.
- 5) **Detailing the joints:** The elements are converted into detailed building components.
- 6) **Processing the model:** The model is processed and transferred to the Computer-Aided Manufacturing (CAM) software.
- 7) **Manufacturing:** CAM and CNC machines or manual labour manufacture the structure.

This workflow is not standardised but is employed throughout this thesis and is referred to as the 7-step workflow. The workflow has significant similarities to DesignToProduction's digital planning of La Seine Musicale[16]. The order and appearance of the steps are likely to vary among projects or institutions.

Other researchers and institutions have developed tools and methods related to the workflow. The following section introduces relevant state-of-the-art research.

Generating Elements: Karamba 3D, which is a parametric toolkit for structural analysis, modularly defines elements. An element is built up by multiple components: The element, cross-section, cross-section shape and material[25,26]. Similarly, this element can be deconstructed into its types. Graphisoft's connection to Grasshopper3D has a similar approach to constructing and deconstructing parametric BIM elements[27]. Front Inc.'s Elefront is more customisable and enables the user to attribute a geometry using key-values. Instead of a database, the information is stored in the baked Rhino-geometry.

Generating joints from 1D-elements: Abstract networks and graph theory can be employed when generating joints to establish a relationship between elements and nodes. These two-dimensional networks can be easily analysed[28]. CITA applies abstract networks that are involved in creating multi-scalar models, which enables multiple levels of details in one model, including centre-curves, blanks and timber operations. [28,29] Further, CITA investigates various ways of extracting/visualising data by using Speckle and SchemaBuilder[30]. DesignToProduction uses the intersection between two elements to generate a preliminary joint[16].

Non-Manifold-Topology: Robert Aish and collaborators are investigating Non-Manifold-Topology (NMT) in relation to Spatial Information Modelling of buildings[31]. Their topology class structure defines a relationship among a building's different geometry types. The class structure includes vertices, edges, faces and other types. Their related software is named topologic[32]. In contrast to abstract networks, NMT enables more comprehensive topology relations. If an edge is selected, neighbouring vertices, edges, and cells can be extracted[33].

Joints type identification: In the case of La Seine Musicale, a spreadsheet interface was used to assign the correct type to each joint[16]. A different strategy is to write a custom algorithm that filters joints based on defined properties, which was the case when Buro Happold identified similar joints in the Morpheous Hotel Exoskeleton[34]. Similarly but more flexibly, Rhino.Inside for Revit enables the user to filter geometry using predefined geometric rules[35]. Using Elefront, the user can search for geometry using described key-values[36,37].

These examples illustrates that there are many ways to create relations in a parametric model. However, most of the examples require advanced parametric thinking to grasp the concepts. A substantial challenge of parametric models with poorly defined relations is that joint type identification becomes unintentionally dependent on a fixed topology. Thus, the topology unintentionally narrows the flexibility, and changing the topology of the structure leads to a risk of crashing the logics that filter joints and output the data of the structure. This finding is backed by Wassim Jabi and Robert Aish: “When a design system omits ratiocination and the definition of topological relationships, it risks brittleness and failure in later design stages”[17]

Figure 4 illustrates this challenge. The left side of the figure shows a shell with foundation joints, edge joints and interior joints. If the list of nodes is sorted based on UV coordinates, a formula based on the surface resolution can output the list index of the joint types. If the topology changes, as shown on the right side of the picture, interior, edge and foundation joints will remain. However, as the shape and division is no longer rectangular, the logic has changed and the code will output wrong results.

the presented logic will crash the code.

Logics that are more flexible than the previous example can be created, but the greater the flexibility, the greater the abstract thinking required. Thus, the flexibility and precision of joint type identification is the main workflow-related challenge chosen for further investigation.

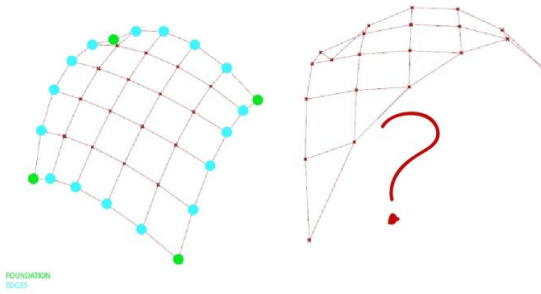


Figure 4: Left side: A gridshell that has Interior nodes, Foundation nodes and Edge Nodes. The previous text explains how sorting rules can identify the different nodes. Right side: If the grid-principle changes, the described sorting rules will not work.

1.2. Objective

There is an acute need for a parametric tool that is customised for detailing structures—a tool that has reduced technical complexity while utilising the power of AAD and digital fabrication—is needed. The last section argued that two main technical challenges of democratising parametric timber detailing are to create a flexible and precise joint type identification and ensure a streamlined transfer from a parametric model to digital fabrication.

Thus, this thesis develops a parametric toolkit for visual programming that is customised for detailing fabrication-ready timber structures. The toolkit aims to reduce the time and algorithmic knowledge required to prepare a model for timber detailing and transfer a detailed timber structure to digital fabrication. The toolkit is built around the described 7-step workflow and is designed to ensure a user-friendly and flexible joint type identification and integrate fabrication-constraints and data throughout the design process.

The developed **JointSearch** approach enables the user to define a joint type’s solution space via one or multiple search criteria. Based on these criteria, JointSearch analyses the joint’s properties and outputs only joints that fulfil the solution space.

The developed **TimberProcessingTools** approach enables the user to use tools that mimic the physical processing of timber structures, e.g., milling, drilling and sawing. In addition to outputting the geometry as a parametric model, the tool also generates a BTLx-file readable by the manufacturer.

The developed toolkit is named Reindeer and is an open source plugin for Grasshopper3D. The 0.5 version of the toolkit was released in November 2019: www.food4rhino.com/app/reindeer.

The toolkit has been developed using a practice-related approach. Two bridges, a gridshell, a log house and a water ramp (ski jump) have been designed, manufactured and built while developing Reindeer. The case structures have evaluated existing versions of the toolkit but have also revealed missing functionality. The key has been to generalise aspects of the concrete projects into concepts that are valid for timber structures in general. These case structures are presented in the Investigation section and explained in the established 7-step workflow.

2. Content and structure of the thesis

This thesis can be understood as practice-related research but has been carried out via the ordinary PhD program at NTNU by the Faculty of Architecture and Design. To share the work in progress and receive responses and critiques throughout the study, a series of papers have been written. Three of these papers have been compiled as a paper-based thesis and are attached as appendices.

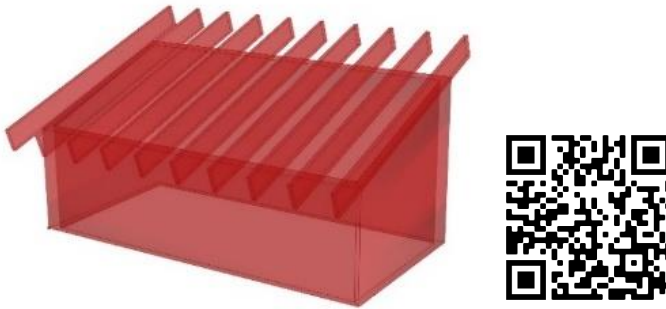


Figure 5: Some figures include QR-codes, which are directed to a YouTube-movie.

This thesis follows NTNU's option that consists of a written component in combination with a product[38]. Reindeer represents this product and is thoroughly explained using text, illustrations and videos in the Investigation chapter. Additionally, the source code is publically available at GitHub. The four parts of the thesis are presented as follows:

I: INTRODUCTION AND BACKGROUND

This chapter provides an introduction to the topic, challenges and objective of this study. Further, the scope, collaborators, chosen software platform State-of-the-Art and methodology are presented.

II: INVESTIGATIONS

The investigation chapter describes the main part of this study. First, the toolkit Reindeer, which is front-end focused, including the intended steps of the workflow, is presented. Second, designed and built case-structures are discussed. In addition to demonstrating the functionality of the toolkit, this chapter demonstrates how the case structure influences toolkit development. Last, a summary of the publications is provided.

III: DISCUSSIONS, CONCLUSION AND FURTHER WORK

First, this chapter discusses the thesis from a broader perspective. Second, a brief conclusion is provided. Last, a suggestion for further work is provided.

IV: APPENDICES

The full version of the papers is printed in the back of the Appendices.

3. Scope

Related to topics within architecture, structural engineering, manufacturing, and computer science, this thesis primarily discusses challenges from an architect's perspective (an architect who is especially interested in digital technology). First-hand knowledge about an architectural process has been beneficial while investigating the objectives. However, the developed tools are probably not optimal from a computer engineer's perspective.

The toolkit is primarily designed for load-bearing structures, such as bridges, gridshells and beam/column systems. However, as the case structures will illustrate, the toolkit is also applicable to other types of structures.

Fabrication techniques and material properties connect this thesis to timber. The thesis does not provide an in-depth analysis of the variations of species, properties, building physics, or other related timber topics.

The process and results of the projects included in the thesis are important aspects of this study. However, an in-depth discussion of how they score as pieces of architecture is not provided.

With the exception of a short description in the background chapter, Building Information Modelling is merely discussed in this thesis. BIM remains extremely important for the construction industry. The interaction between computational models and BIM is an increasingly powerful method. Rhino Inside[35] enables the direct use of Grasshopper inside Revit. Similarly, Graphisoft has developed a Grasshopper plugin for interacting with Archicad[27]. However, to limit the scope of this thesis, the BIM interaction will not be discussed.

Timber specific CAD/BIM software is only briefly described. A series of popular timber design software bridge the gap from CAD to CAM, e.g., CadWork, Sema, and HSB CAD. These software contain predefined advanced parametric models, e.g., timber roof generators, which are efficient and powerful within their solution space. However, these software are not suitable for flexible, user-friendly, parametric modelling. Thus, these programs are beyond the scope of the thesis.

4. Collaborations

This thesis is conducted via a series of collaborations and highly influence the result. Following presents the different collaborator's roles:

Moelven Limtre

Moelven Limtre is the largest glulam manufacturer in Norway and has been a pioneer in developing ambitious timber structures. In the 1990s, they developed the dowels and slotted-in steel plate solution. This solution was integrated into both sports arenas for the 1994 Lillehammer Winter-Olympic[39] and large timber bridges, such as the Evenstad Bridge[40] (refer to Figure 6). They are responsible for Mjøstårnet, the world's tallest (June 2019) timber building [41].

Moelven Limtre was an essential collaborator in the Orkla Bridges Project. They were the manufacturers of the final design and also played an essential role in sharing knowledge about their manufacturing strategies and challenges. In addition to multiple day visits to the factory, we had two extended visits of 3 days. During these visits, we developed prototypes for how to transfer parametric models to fabrication.



Figure 6: Evenstad Bridge.

Marcin Luczkowski

Ph.D.-candidate Marcin Luczkowski is a structural engineer who is employed at NTNU's Faculty of Engineering - Department of Structural Engineering, and has been working with the same topic but from an engineer's perspective.

Luczkowski represents the engineer's voice in the development of the toolkit and has been programming parts of the algorithm. The final workflow and concept of the toolkit is a result of an ongoing debate about how to unify architect's, engineer's and manufacturer's ways of working. Luczkowski also was a member of the design-team of the Orkla Bridges and Printshell projects.

Steinar Hillersøy Dyvik

Steinar Hillersøy Dyvik is an architect who is employed at NTNU as a PhD candidate and has been a member of the design team of the Orkla Bridges and Printshell projects. He has been an essential contributor to regular discussions and criticism about the development of the toolkit and has also been beta-testing the toolkit and scripted a few of the toolkit components.

Rallar AS

Rallar, which was established in March 2017, is a firm that I run with Anders Gunleiksrud and Thea Hougsrud Andreassen. The water ramp and log house were designed and partly built via Rallar AS. www.rallararkitekter.com

Nikken Sekkei

In early 2018, CSDG[42] initialised a collaboration with a Japanese architecture office named Nikken Sekkei, which has a group named Digital Design Lab. With them, we developed the current C# version of Reindeer. Bunji Izumi was their project leader. The shared development primarily occurred online via GitHub. Additionally, I had two working sessions in Tokyo: one week in February/March 2018 and two weeks in August/September 2018.

The initial concepts of the toolkit were developed before the initialisation of the collaboration. However, these concepts have been a great resource for further assessing the relevance of the toolkit and have contributed to project management and programming skills.

www.nikken.co.jp/en - www.openddl.com

SINTEF

SINTEF is a Norwegian, independent research organisation. SINTEF was a research partner in the Orkla Bridge Project and was responsible for organising the research project funded by Innovasjon Norge. Additionally, they functioned in a mentoring role during the design and research project. Nathalie Labonnote and Berit Time were SINTEF's representatives.

Declaration of contributions in developing the Reindeer toolkit

With the exception of modules related to the structural analysis, the first two versions of the toolkit were primarily written by me. Marcin Luczkowski wrote the code related to the structural analysis and parts of the BTL prototype and intellectually contributed to the toolkit.

0.3 PTK and 0.5 Reindeer have been a collaboration project between Nikken Sekkei and Marcin Luczkowski, Steinar Hillersøy Dyvik, and me at CSDG.

In addition to contributing to the concept of the toolkit, I have been responsible for developing and writing the modules that relate to composites, global alignment, node-plane generators, detail generation, DetailSearch, TimberProcessingTools, export to BTLX and NURBS-generation of elements.

The toolkit development has been highly dependent on the programming skills from Nikken Sekkei DDL.

5. Software Platform

This section outlines the chosen software platform, programming language and collaboration tools. Choosing a parametric platform is a major strategic choice. The other aspects are partly subjective or a consequence of the chosen parametric platform. Reindeer is developed as an open source Grasshopper plugin. The employed language is C#, which was developed by applying Microsoft Visual Studio and is shared and documented via GitHub. The following section discusses the choice of software platform.

5.1. Parametric platform

Digital parametric tools are used to create an algorithm that solves a task. Depending on the task's complexity, different tools will be applicable. A large variety of both programming languages and software is available for creating algorithms. In general, the more capable is a programming language, the more difficult is learning the language[9]. Choosing a parametric platform is an informed compromise between capability and usability. BIM, Script-based and Visual Programming are discussed.

BIM

Building information modelling (BIM) is often considered to be a contrast to parametric modelling. The generation of geometry is primarily based on pick and place. However, the objects that are placed are advanced, editable parametric models that are hidden behind a graphical user interface (GUI).

When placing a window, door, or stair, the software enables the user to adjust the model by deciding various parameters. Step height, window width, floor height, door material, and hinge type are a few examples. Figure 7 shows a window-generator interface.

The interface is understandable, and no programming skills are required. These models are relatively flexible. Generating bespoke windows is difficult when applying a standard window generator in BIM software.

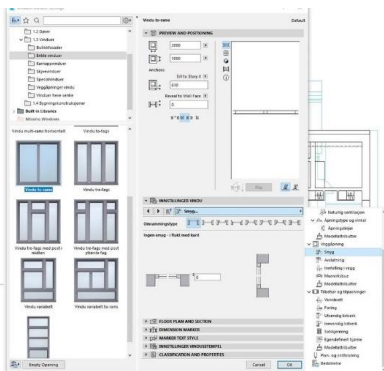


Figure 7: Window-generator in Archicad is a parametric model. The interface is intuitive, but the geometric options are limited.

Scripting

All digital algorithms, including BIM-models and visual programming, are transformed step by step to/from binary code, which is readable/writable by the hardware. Multiple levels wrap binary code into more readable code. C# and Python are examples of highly readable scripting languages that are applicable to creating parametric models.

Scripts are constructed by writing line by line. The correct syntax is required to make the code run, but modern compilers help a developer correctly write code. Writing code increases the flexibility and capability. A few lines of code may rewrite a complicated algorithm that was created using visual programming.

One platform strategy is to develop a Software Developer Kit (SDK), which is familiar to Rhinoceros' Rhinocommon. This kit can reduce the complexity and knowledge that is required for detailing timber structures but would require the user to know a programming language. Figure 8 shows a scripting interface in Microsoft Visual Studio.

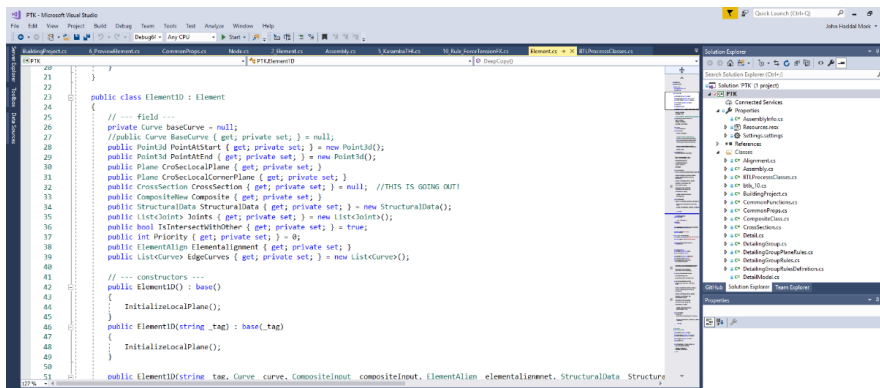


Figure 8 Visual studio interface and C# Scripting.

Visual programming software

When applying visual programming, the user does not need to know any programming syntax but must use prewritten functions wrapped as components. These components have inputs and outputs and are connected by “wires.” By combining a series of components, complex algorithms can be created.

Visual programming is designed to be user-friendly and satisfactorily flexible. If a user inputs a surface component into a curve component, the surface automatically transforms the surface into a boundary curve. One operation is sufficient as most components have a series of overloads (allowing various inputs). The upside of the overload is that the software becomes forgiving and user-friendly. The downside is that the designed algorithms perform slower than a more specific algorithm. Figure 9 shows a Grasshopper code, and Figure 10 shows a dynamo code.

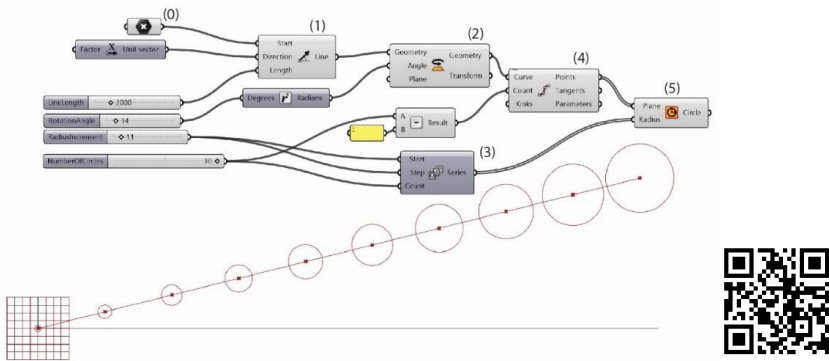


Figure 9: Example of a parametric model that generates linearly distributed circles with increasing radiuses. Adjusting a slider will regenerate a new result. (0) Starting point of the line, (1) constructing a line with step 0's starting point, x-direction and 2000 length, (2) rotating the line 14 degrees around the red plane, (3) making a number list starting at step 11 with 10 steps, (4) the rotated line.



Figure 10: Autodesk Dynamo Studio is a visual programming language. (Photo from Autodesk's webpage. [13])

5.2. An Open Source Plugin for Grasshopper 3D

Based on these aspects, an open source plugin for a visual programming software was chosen as a platform. The plugin for visual programming software provides a desirable balance between capability and usability. As the plugin is open source, more advanced users may want to hack or further develop Reindeer.

Grasshopper 3D and Dynamo are major parametric software for architecture creation. Dynamo is well integrated into Autodesk's ecosystem, including Revit Architecture. Differently, Grasshopper 3D is more general and is not connected to any BIM software; however, benchmark tests indicate that the Grasshopper 3D engine is substantially faster than the Dynamo engine[43]. Further, Rhino.Inside and the Archicad Grasshopper plugin bridge the gap from AAD to Revit and Archicad. Choosing a Visual Programming Software is subjective and based on previous experience. For this software development, developing the code for Grasshopper 3D was chosen, but the principles are transferable to other software. Figure 11 illustrates the chosen software platform.

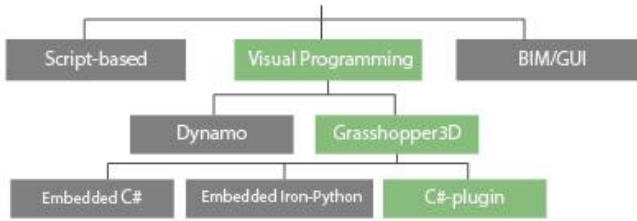


Figure 11: Chosen Software Platform. Visual Programming enables Grasshopper 3D, which enables a C# plugin.

Programming Language

Custom C# or IronPython components can be directly written at the Grasshopper Canvas. However, this method is suitable for prototyping but is not optimal for collaboratively developing a shareable software. The best-practice method of creating plugins for Grasshopper is to apply the C# Visual Studio Grasshopper Template[44]. Thus, C# and Visual Studio were chosen as the programming language and Software development tool, respectively.

C# is a modern, object-oriented programming language[45,46]. Object-oriented modelling enables the programmer to define highly flexible classes that generate objects. Therefore, geometric data relations are significantly improved. For the case of Reindeer, classes exist for the assembly, elements, nodes, elements, details and a series of other contents.

GitHub

Collaborative development and version tracking were simplified by applying GitHub. Git is an open source version control system, and GitHub[47] is an online platform that applies Git[48]. GitHub stores the latest version of the project online and simplifies the process of merging iterations of code. Further, discarded versions of code can be reviewed and analysed.

- The complete source code is available at GitHub: github.com/CSDG-DDL/PTK
- The compiled plugin is shared at www.food4rhino.com/app/reindeer
- The Reindeer chapter presents [0.5 Reindeer](#) and is store at [Zenodo](#)[49]

Name	Period	Platform
0.1 Truss Bridge Algorithm	August 2015-June 2016	Grasshopper Components
0.2 Timber Toolkit	June 2016-March 2018	Grasshopper/Iron Python
0.3 PTK	March 2018-January 2019	C#/Visual Studio
0.5 Reindeer	January2019-November 2019	C#/Visual Studio

Table 1: Versions of the toolkit

6. State-of-The-Art: Relevant tools and methods

A series of existing tools and methods relate to the objectives and challenges of this study, and are presented in this section. Solutions have been developed by both companies and research institutions and have been shared as tools or methods. Several of the following topics are relatively new. Both Rhino.Inside and Buro Happold's work was published in 2018 and strengthens the relevance of Reindeer's approach.

The first part presents tools and methods that have been developed by leading firms and research institutions. The last part discusses various strategies for transferring timber geometry from design to digital fabrication.

6.1. DesignToProduction: Modelling strategy

The consultancy firm DesignToProduction is a pioneer within both parametric modelling of timber structures and digital fabrication of timber structures. Although many papers are available[5,10,24,50], only the content of one recent paper is discussed: "La Seine Musicale – A Case Study on Design for Manufacture and Assembly"[16].

In La Seine Musicale, which was designed by Shigeru Bahn, DesignToProduction was responsible for converting the design to production. The structure is a hexagram pattern that consists of 15 horizontal beams and 84 diagonals.

Registering joints

Their preliminary modelling is concerned with defining preliminary joints at the beam/diagonal intersections (among other tasks). These joints are then sorted into types based on their connected elements. A spreadsheet interface is used to assign the correct type to each joint. DesignToProduction names these joints as abstract containers that initially contain local beam tangents, surface angles, and other attributes, which are subsequently filled with fastener objects, and fabrication operations. Note that the element's fabrication operations are stored in the joints. When the design is ready for fabrication, the segments collects the relevant operations from the joints.

Detailing and fabrication

In La Seine Musicale, 2798 joints are generated from 8 main types and 120 subtypes. The most complex instance of a parametric joint type is always modelled first. By this method, other easier details are located within the solution space. When the design is ready, they transform the model into a detailed model that contains all operations. Afterwards, they apply the BTL format to transfer to the CAM.

6.2. Front Inc.: Parametric Building Information Generation

Building Information Modelling (BIM) has been discussed. Building Information Generation (BIG) is familiar but simultaneously very different from BIM. Front Inc. is a "*cross-disciplinary collective of creative individuals who bring specialist facade system design expertise to customers*"[51]. They have been involved in a series of ambitious projects, such as the CCTV-building, Barclays Center, and Morpheous Hotel[51]. The latter's aluminium façade system that covers the exoskeleton consists of 21 000 individual panels. To solve the complexity of these

projects, they have developed the BIG methodology. The following brief explanation is extracted from their paper and webinar[36,52].

The purpose of BIG is *“to collect, combine and give meaning to information generated in the design process and thereby give rise to a BIM model.”* While BIM has a standard set of properties and classifications, BIG is customised to generate only the information required in an appropriate format. *“The type of information does not have to be pre-specified, but can be assigned where needed.”* An important feature is the ability to store information related to the geometry by using attributes: a key and an associated value. These data are commonly stored by generating a text file, spreadsheet or database.

However, Front Inc. stores information inside the geometry, which is performed by using Rhinoceros and a self-developed plugin named Elefront[37]. By using this plugin, a designer can *“filter, reference, sort and order geometry in order to be able to perform the next sequence of functions on the geometry.”*

Their way of referencing geometry also enables another crucial aspect: scalability. Front Inc. describes a collaborative workflow, where they use multiple files that are linked. This workflow decrease file size and complexity and simultaneously enables multiple designers to work in parallel.

6.3. Buro Happold: Identify similar connections

Buro Happold was also involved in the Morpheus Hotel. They were responsible for the structural design, detail design and construction documentation for all steelwork connections. Their paper describes their methods for identifying nodes [34].

To reduce fabrication and erection time, they had to identify similar connection types, which was achieved by searching a geometry file that contains member centreline geometry, section shapes, and sizes. The connections were classified based on *“whether members were straight or curved, the member shapes, and sizes and the angles between adjacent members”*. This strategy reduced the number of connection joint types from 2500 to 400. The paper did not indicate whether this number related to individual scripts or unique joints.

Smart Clustering

Buro Happold has also released a plugin that has similarities to the previously described connection identification strategy. The plugin is named Smart Clustering and is a *“clustering algorithm that reduces the number of unique panels/nodes within a given surface”*[53]. The node clustering algorithm groups similar nodes. The user defines similarity by inputting angle tolerances in all three axes. The number of groups that are produced is dependent on the shape and tolerances, but does not include other attributes.

6.4. CITA and Innochain

Researchers at the Centre for Information Technology and Architecture (CITA) are performing research on parametric timber, from previous investigations[54,55] to their current research on multi-scalar modelling, and ways to parametrically design timber structures. Their research relates to both parametric timber and geometric relations.

Abstract networks and graph theory

CITA applies abstract networks when abstracting complex geometries. These networks are two-dimensional and consist of edges and nodes. Graph theory is used to analyse and manipulate the network. With this solution, the user is “able to access an ordered list of nodes (and corresponding indices), an ordered list of edges (and corresponding nodes indices at both ends) and the number of connected edges at each node. This strategy enables the construction of a Multi-Scalar Model that generates and links data by the only means of nodes and edges, and thus presents itself as a very lightweight format in the context of 3D modelling for architectural conception.”[56] This way of thinking is clever and highly relates to the solution described in the investigation chapters.

Multi-Scalar Modelling

Multi-scalar modelling is an approach in which an object (in this case, a timber component) is represented using multiple models on different scales[57]. CITA aims to “enable a new modelling paradigm where a continuous hierarchy is defined across scales within a graph, from an abstract network of lines to the complete fabrication data set of each architectural component.”[56]

The different scales include the fibre direction, blank model, component model, and timber processings[58].

Schema-Based Workflows

CITA is also developing a workflow to better visualise the complex relationships and components in a multi-scalar model. They are developing a Schema-Based workflow[30]. By applying Speckle (An open source data platform for AEC)[59], they convert their model into a database. Thus, custom queries can be written to retrieve the desired geometry. Additionally, they have developed a tool named SchemaBuilder. This tool aids the user to customise the properties that will be attached to a model, which is manually performed via an intuitive user interface with checkboxes. The purpose of this solution is to customise the export output for a specific purpose, e.g., structural analysis or digital fabrication.

6.5. Topologic: Non-Manifold Topology (NMT)

Topologic is a project led by Wassim Jabi and Robert Aish; it is “a software development kit and plug-in that enables logical, hierarchical and topological representation of spaces and entities”. The research focuses on Non-Manifold Topology (NMT), which is defined as cell complexes that are subsets of Euclidean Space[60,61]. This approach enables a comprehensive but lightweight and idealised model of a building/structure[32]. NMT differs from a solid geometry boundary representation and enables various types of geometries and intersections within the boundary representation[17]. The core is a class-structure with a superclass named Topology. This class connects to other classes named Vertex, Edge, Wire, Face, Shell, Cell, CellComplex and Cluster. When a geometry is selected, neighbouring objects, regardless of type, can be extracted.

This research offers a novel approach to a comprehensive topology model of a building structure. Since Reindeer and Topologic have been developed in parallel, the kit is not

integrated into the software. However, and as further work will explain, combining Topologic’s geometry hierarchy and Reindeer’s searching capability would be an interesting next step.

6.6. Rhino.Inside for Revit

Rhino.Inside is a recent development by McNeel and enables Rhinoceros to run inside another software. Rhino.Inside for Revit has a clever way of using multi-objective rules to filter wanted objectives. The various filter components can be combined and include components to check whether a geometry intersects, whether a value of a key is smaller/larger than a test value, and whether the element belongs to a category. This solution is powerful and has significant similarities to Reindeer’s search components, which will be subsequently presented. The solution enables multiple filters, filters to be flipped, and filtering based on both geometric tests and data properties.

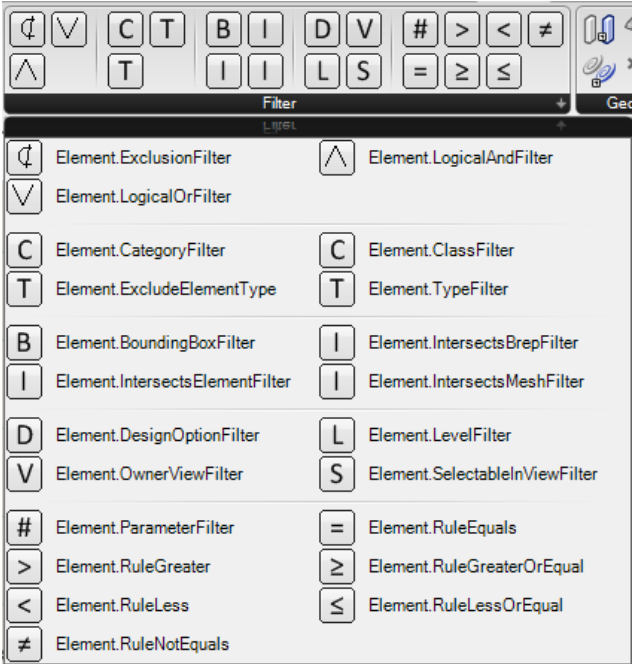


Figure 12: Possible filtering options using Rhino.Inside for Revit.

6.7. Timber Plate Shell Structures

Christopher Robeller is developing timber plate shell structures. His inspiration is from traditional joinery principles, and he applies advanced algorithmic geometry processing. The project is fascinating for many reasons. For example, how the design is constrained by the assembly and the joinery becomes an important part of the architectural expressions. However, the developed toolkit for the design of timber plate structures makes the work relevant to this study. Analysis, geometry generation and fabrication require custom codes. These custom codes have been made publically available as a Grasshopper plugin and have democratised the project’s developments[62–64].

6.8. Transferring a digital design to fabrication

Most timber fabrication processes require information about the initial blank, its operations and the final results. Conventional 3D formats contain only the final result, and in the worst case, the fabricator is required to remodel the 3D geometry using CAM-specific software.

How a model is transferred is decisive to ensure an efficient digital chain (flow from design to fabrication) [65]. The following three strategies are also discussed by DesignToProduction[50].

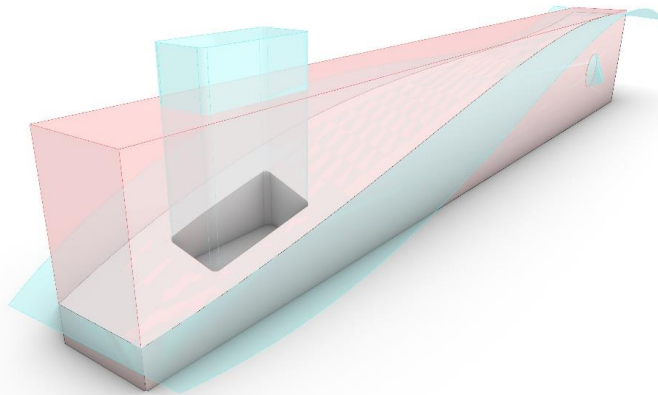


Figure 13 Example component. Red geometry represents the blank, cyan geometry represents the processings and white geometry represents the final results.

Feature recognition

Feature recognition enables the user to apply domain-specific CAD software. CAD-models are exported to a CAM system, and then, the feature recognition algorithm analyses the geometry and generates machining data. According to DesignToProduction [50], this method works well for simple geometries but quickly becomes unfeasible for complex geometries. Choosing this method for complex geometry causes laborious remodelling. Thus, errors may apply.

Figure 13 exemplifies this finding. If a rectangular hole is placed on a flat surface, the feature recognition would probably recognise the geometry as a rectangular hole, and thus, a pocketing procedure would be applied. Since the surface is curved, the rectangular geometry is less identifiable for a computer.

Custom digital chain

Large, prestige projects may have the budget to create a project-specific digital chain: If parametrically modelled, many components may have some degree of similarity[4]. Thus, a custom, machine-specific PostProcessor that bypasses the CAM system can be scripted.

The solution is clever but is often too technically demanding. Programming a custom digital chain is time-demanding and requires a team of technically skilled designers. La Seine Musicale is an example of a project that has been realised using a custom digital chain[24].

Robotic arms and associated software have made custom digital chains more accessible. Classic robotic arms do not have the same efficiency and strength as those of traverse-based CNC systems but have the advantage of being flexible. Software, such as HAL Robotics[66] and

KUKA PRC[67] (Parametric Robotic Control), enable a robot to be directly programmed from Grasshopper 3D. Figure 14 shows an installation constructed from glulam cuttings. The Rapid code for the ABB robot is programmed using HAL robotics.

The solution for applying a custom digital chain primarily relies on the same three ingredients: blank, processing, and resulting component.

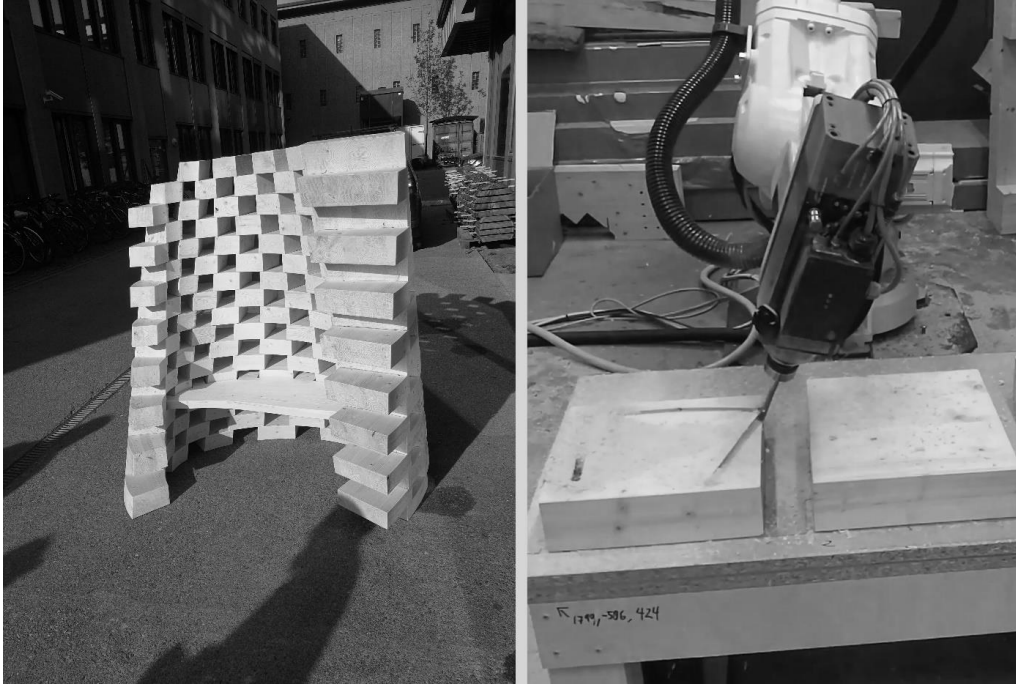


Figure 14 Custom digital chain using HAL robotics.

Building Transfer Language (BTL) and Woodpecker

Clipped from the developer's web-page, “*BTL and BTLx are formats that provide a parametric description of the geometry of wooden building components and their processing as well as structural information for prefabrication and assembly*”[68]. BTLx is a newly developed version of the format and is open and free for use and implementation. The benefit of this format is that it is machine-independent, which means that it describes processes as parametric geometry. A post-processor is required to translate BTL information into machine-specific NC code.

The format is tailored for timber structures, which means that information about the grain-direction, species and other parameters are included. Further, all processes are digital emulations of timber-specific machining processes. If programming the previous example, the rectangular hole would probably be defined as a mortise, pocket, or slot.

Simplified, the outputted XML structure appears as follows:

```

<BTLx>
  <Project>
    <Parts>
      <Processings>
        </Processings>
      </Parts>
    </Project>
  </BTLx>

```

This structure implies that a BTLx format has a project, a project can have multiple parts, and a part can have multiple processings.



Figure 15: BTLx is XML-based. The figure shows how a blank and tenon are defined.

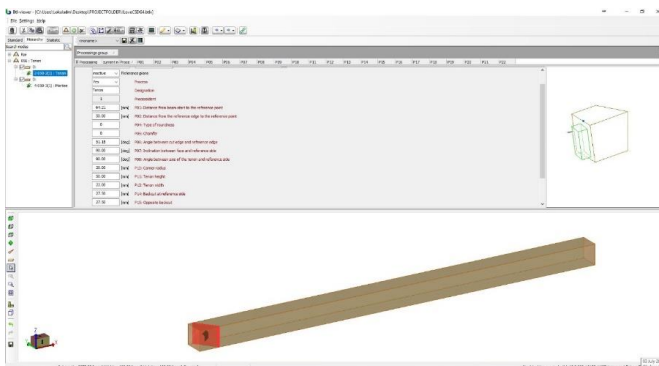


Figure 16 BTL Viewer. A beam with a tenon. The top half of the screen renders the properties of the tenon.

Woodpecker is a plugin for Grasshopper that enables BTL export. The plugin enables Drilling, Cut, Slot, Pocket, and Free Contour[21]. This plugin was released in 2014[50]; a new version was stated to be released in 2017 (has not happened)[24]. The new version is to be based on BTLx and include a more extensive set of BTL operations.

The plugin works well. However, the purpose of the plugin is to export a design to fabrication. The plugin does not visualise the result and does not output a processed geometry. The plugin is not integrated into the entire workflow; it is only integrated in the last step.

6.9. Summary

This section provided an overview of related tools and methods relevant to parametric timber detailing.

CITA, Topologic and DesignToProduction are examples of existing and well-working methods for establishing a parametric model with topology relations. Further, DesignToProduction and Front Inc. have methods for identifying specific geometry.

The BTL format is promising and provides a powerful translation between design and fabrication. Woodpecker is the only available toolkit that enables BTL translation; however, it is not well integrated in the entire design process.

This review reaffirms that both detailing preparation and fabrication transfer are complex tasks that require developed parametric thinking. However, numerous opportunities exist for implementing the described methods in tools that reduce the required technical knowledge. Implementing the transfer from design to fabrication in the entire design process has potential. Information about the BTL element can be gathered when defining an element, and generation of BTL operations can be a part of the architectural design process.

A tool that automates process-related aspects of parametric timber detailing is needed. Such tool would provide a new learning step between CAD/BIM and custom, advanced AAD.

7. Methodology

To solve the objectives of the study, practical knowledge and insights about the complete digital chain are required. The challenges are wicked and are not within the domain of one profession. Further, the industry faces complex challenges on a daily basis that are not sufficient for complete simulation in a lab. Thus, this study has been carried out via practice-related research.

Reindeer have been iteratively developed by gaining knowledge from state-of-the-art methods, reflection through writing papers and obtaining feedback at conferences. However, the main source of insight about how to develop Reindeer has been the application of the code to the design, manufacture and building of structures. A central feature of the research design has been to ensure an iterative flow from design-process insights via generalisation and implementation of functionality in the toolkit.

This chapter explains the different research activities applied in this study and how they were combined in activity configurations.

7.1. Research Activities

Practical research primarily consisted of five research activities[69]:

- 1) design, manufacturing and building structures
- 2) toolkit prototyping
- 3) toolkit development
- 4) parametric modelling
- 5) generalisation.

1 Designing, manufacturing and building timber structures

A central method for gaining insight and knowledge includes designing, manufacturing, and partly building timber structures. Most case structures have been developed by a multidisciplinary team (architect, structural engineer, and manufacturer), ensuring multiple perspectives.

During this study, two bridges, a gridshell, a water ramp (ski jump) and a log house have been built. With the exception of the gridshell, all the structures are permanent buildings that are required to follow building codes and be durable and cost-efficient.

The permanent status and realistic application of the structures have established requirements for the quality of the details and the efficiency of the chosen manufacturing process. Further, Moelven Limtre has been involved in two of the projects, which has challenged the digital interface between architects/engineers and manufacturers and has placed the research in a realistic context.

The Orkla Bridge project, which was thoroughly presented in the investigation chapter, played a unique role as the main case structure. Early in the design process, the application of a known structural and detailing system was determined: Two timber trusses with dowels and slotted-in-plates as connections. A doubly curved railing was also chosen. These two preconditions were defined to function as a benchmark for the developed toolkit: The toolkit had to handle a known but complex detailing principle and simultaneously handle complicated geometry.

2 Toolkit Prototyping

The initial development of the toolkit was a typical prototyping process. The prototyping was carried out using Grasshopper 3D and IronPython[70], which has both benefits and disadvantages. A key benefit is that Grasshopper 3D is a visual programming language, which renders prototyping an algorithm efficient and provides immediate feedback about how the algorithm functions and performs. Further, the required programming skills are limited.

The approach of the toolkit prototyping is intuitive and rapid. The goal of this method was to establish and understand a problem instead of solving a problem.

Although some plugins are developed and published using the described platform (such as Ladybug[71]), it was determined to be unfit for developing the publically available timber toolkit. Grasshopper native components are general and flexible (as they should be) but limit the calculation performance. The use of native components produced a prototyped toolkit that calculated too slowly.

3 Toolkit Development

The toolkit development was described in the software platform chapter.

4 Grasshopper-based parametric modelling

Some of the built case structures were designed, or partly designed, without applying any version of the developed toolkit. Native Grasshopper components and other relevant plugins were used to develop a design.

This approach was applied when the case project was primarily connected to practice: Often, a relation between a code's flexibility and the amount of work demanded to develop the code exists. When related to practice, time limits are essential. Thus, most of these codes were targeted at a specific problem and initially purposed as on-off solutions. Woodbury's terms "Throw code away" and "Copy and modify" are descriptive for this activity[9].

However, both the intellectual process of developing the code and the code functioned as an essential input for generalisation. The variety of the practice-related case structures challenged the flexibility of the toolkit.

5 Generalisation

A generalisation of knowledge from the design and build processes has been vital. Reflections and analyses of the designed structures, manufacturing processes, parametric models and prototyped solutions have produced extracted, generalised concepts. Tacit knowledge is gained via the other activities. Transforming tacit knowledge into words, diagrams, rules or relationships is often difficult. The challenge is to achieve an abstraction level that is detached from a specific project. However, when multiple projects are parametrically designed, similarities are identified.

The generalisation act has been practical, analytical, and "accidental." Practical generalisation has been achieved by identifying a lacking functionality of the toolkit while designing a structure. Analytical generalisations have been performed after a case structure has been designed and built to identify the core of a solved design problem. Accidental generalisations have often occurred using mundane activities, such as jogging or showering. These

generalisations have been related to the sudden identification of significant similarities among initially different structures. The findings from the generalisations are presented at the end of the Case Structure chapters.

7.2. Activity configurations

The research activities have been combined in four different configurations: Integrated, Applied, Detached and Implement. The names of the configurations reflect the extent to which a given version of the toolkit influenced them.

Integrated

The integrated configuration loops designing, parametric modelling, generalisation, and toolkit prototyping while designing a structure. The designed and built case structure makes the study more concrete; however, the structure is not the main goal of the configuration. By applying the configuration, the case structure directly informs the toolkit, and the toolkit directly informs the case structure. This configuration was applied while designing the Orkla Bridges. As described, the structural and detailing principle was locked early in the design process and set requirements to be solved via the simultaneously prototyped toolkit.

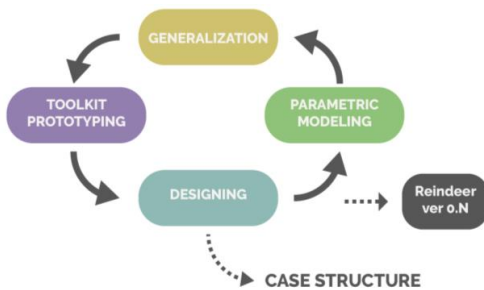


Figure 17 Integrated configuration.

Applied

The applied configuration designs a structure by applying a given version of the toolkit. The main goal of this approach is to design a well-functioning, built structure. Thus, if the toolkit is not applicable, additional custom algorithms are developed. This approach is practice-related and plays an essential role in evaluating the relevance of the toolkit and identifying lacking functionality. Lacking functionality was hacked while designing. The generalisation identified whether the lacking functionality was worth generalisation and implementation in the toolkit's functionality.

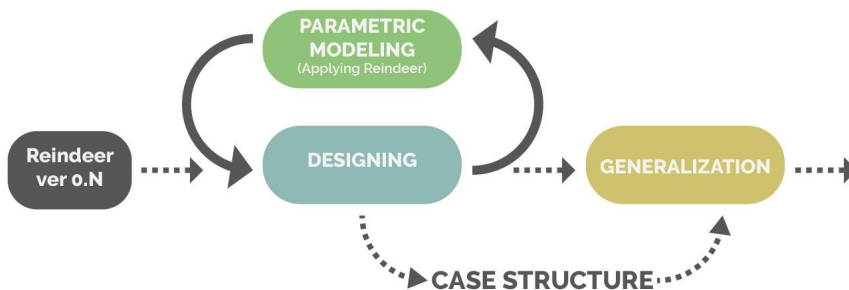


Figure 18 Applied configuration.

Detached

Although the detached configuration is also practice-related, the design process does not apply any version of the toolkit. Thus, the design process is not constrained by the toolkit, and a larger design space is potentially explored. By generalisation, both during the design process and after the design process, concrete parametric modelling challenges are extracted. The following questions were asked: What if the given design project was conducted using the toolkit? What functionality does the toolkit miss? How does the toolkit have to evolve to become applicable?

This thesis presents two case structures that applied the detached configuration: the log house and the shelves.

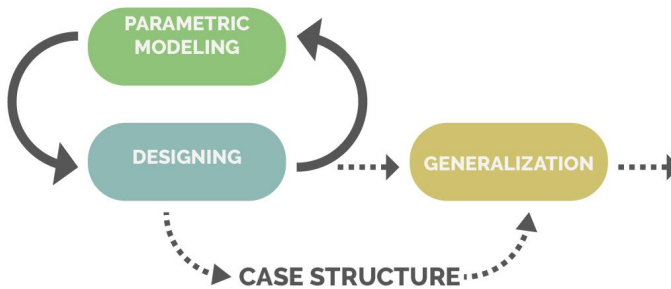


Figure 19 Detached configuration.

Implement

The implement configuration entails classic programming development. The input for the activity is a given version of the toolkit and extracted knowledge from a generalisation. The output is a new version of the toolkit.

Although the main activity is programming via toolkit development mode, the toolkit is continuously tested using small parametric modelling experiments. This activity configuration dominated the last 24 months of the study.

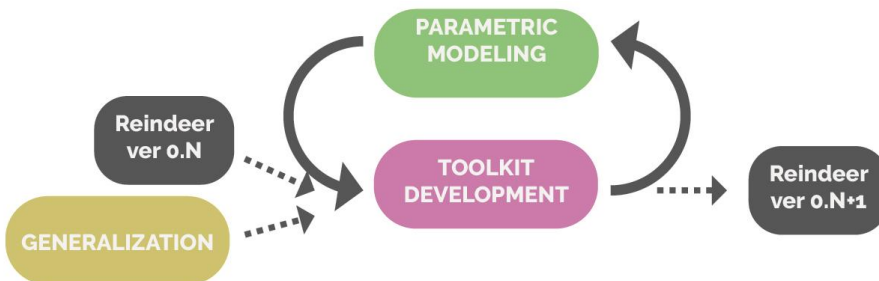


Figure 20 Implement configuration.

7.3. Summary

This research is related to and dependent on practice and has primarily generated mode 2 knowledge[72]. This research has similarities to both research by design[72], and practice-based and practice-led research[73]. Johan Verbeke defines research by design as “*The kind of research where the process of designing and experience from practise plays a crucial role in the research, not only as the input to be observed but more importantly as the method and outcome of the research*” [72].

Verbeke’s definition has strong similarities to the conducted research. However, practice-related methods have not been investigated and will not be discussed in this thesis. The core strength of the presented method of developing Reindeer has been an iterative process that involves ideas, prototyping solutions and realistic tests by building real structures.

II. INVESTIGATIONS



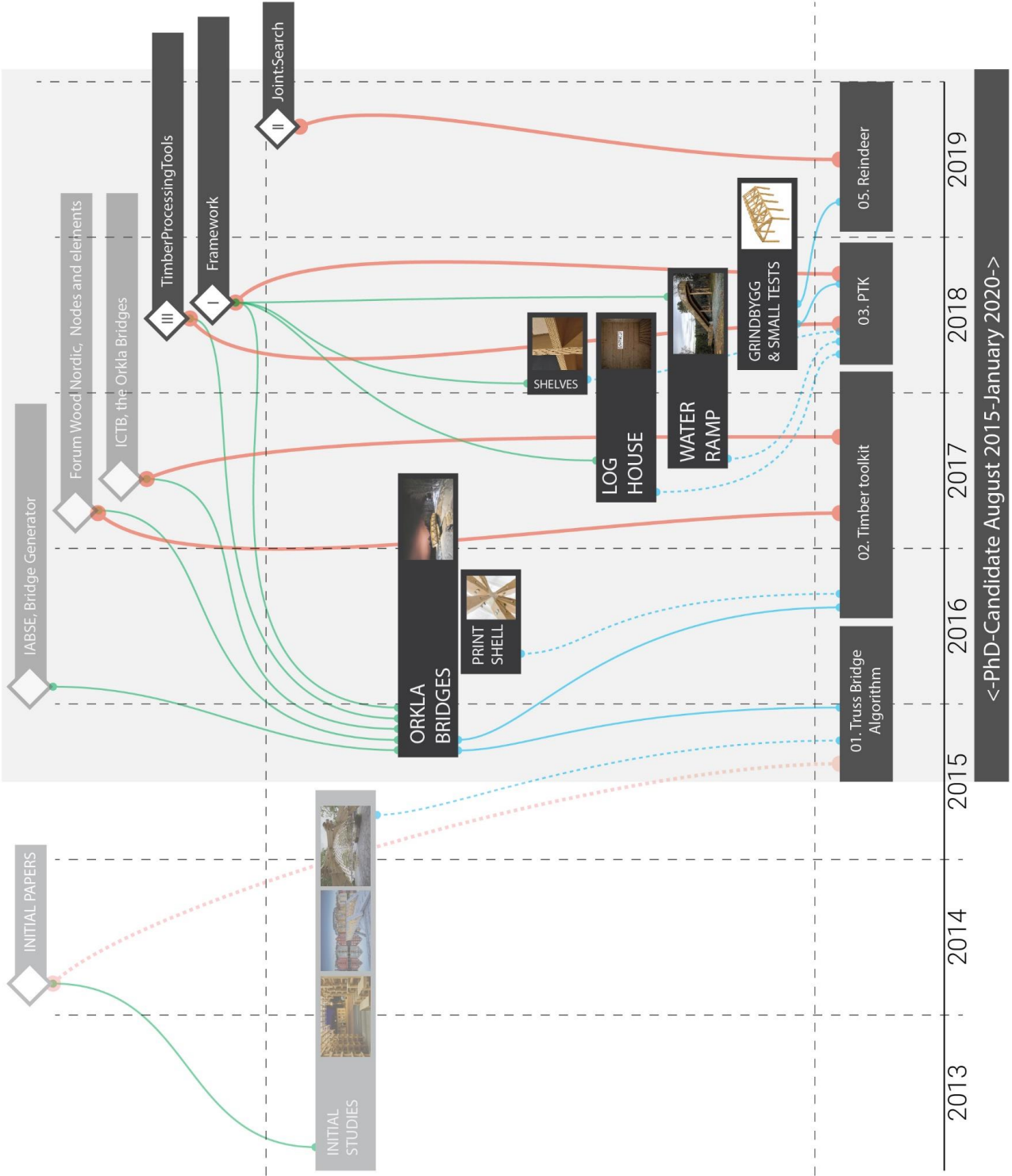
Papers

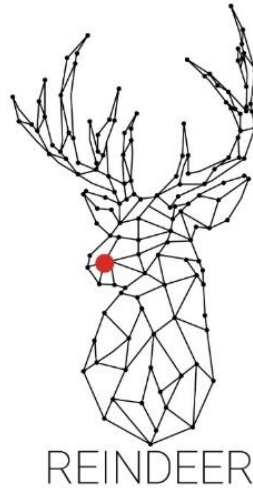


CASE
STRUCTURES



REINDEER
PLUGIN





8. Reindeer – Grasshopper-plugin

A reindeer is an animal that lives around the polar circle. In harsh and ever-changing climates and landscape topologies, the reindeer searches for food by sniffing out lichen beneath the snow[74]. Similarly, the Reindeer toolkit searches for joints using geometric properties. Even if the topology and typology changes, Reindeer will still find its joints. JointSearch is an important part of the Reindeer plugin.

The goal of this chapter is to provide an overview of Reindeer's intended workflow. This description is primarily front-end and explains how the components work and are connected. This explanation is expected to simplify the technical understanding of the papers. Example files, which further explain the concept, can be downloaded at www.food4rhino.com/app/reindeer

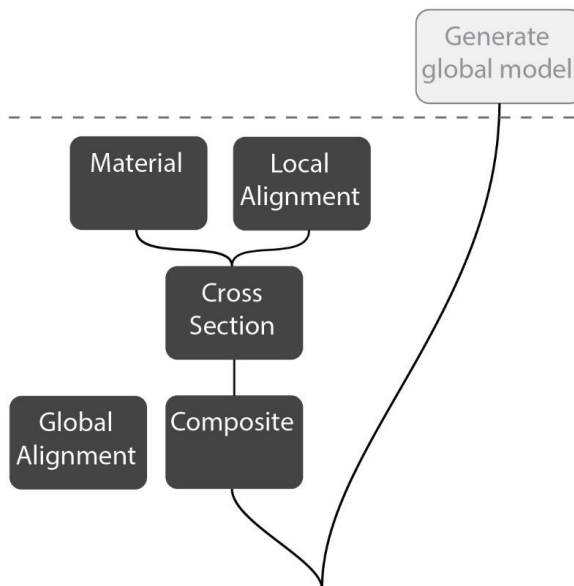
8.1. Workflow

The workflow of the Reindeer plugin is divided into the same 7 stages that were presented in the introduction, from inputting centre-curve geometry to digital fabrication. All seven steps are usually required when fabricating a structure. While exploring structural concepts, however, the first two stages are sufficient.

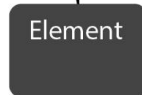
- 0) Generate global model (beyond Reindeer's domain)
 - 1) Define elements
 - 2) Assemble the structure, and define geometric relations
 - 3) Structural analysis of the structure
 - 4) JointSearch, search and output joints based on search criteria
 - 5) Detail the elements using TimberProcessingTools (and nodes)
 - 6) Process model (generate NURBS or CAM output)
 - 7) Manufacture and build

The next page illustrates the workflow.

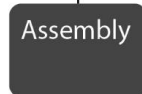
0) Global geometry



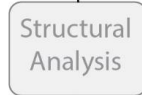
1) Element Generation



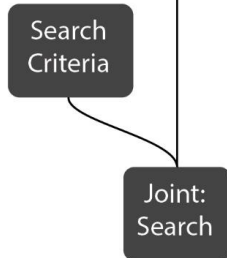
2) Assembly



3) Structural Analysis



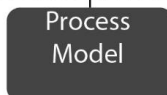
4) Joint Search



5) TimberProcessingTools Detailing



6) Process Model

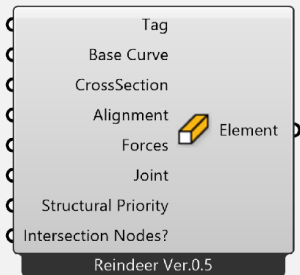


7) Manufacture and build



8.2. Step 1: Defining elements

An element requires a curve to be initiated. Additionally, the user can define the name, cross-section, material, cross-section shape, eventual sub-element, local alignment and structural priority. By using a component (Grasshopper-function) for each property, the element's definition becomes modular and flexible. Below are the components that in sum defines an element.



Component: Element

With the exception of the Base Curve, the element has default values for all inputs, which enables a design-process that can incrementally add information to the element.

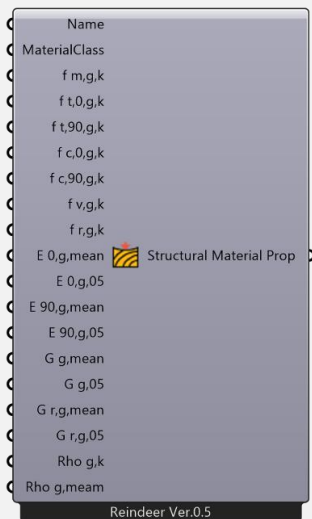
PS1: The boxes on the left are "components"

PS2: Published version does not contain Forces/Joint - input



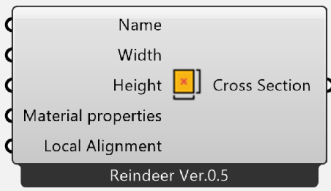
Component: Local Alignment

The local-alignment component enables local offset Y, local offset Z and rotation relative to the composite. If multiple parts exist in the composite, the local alignment component is used to compose the composite.



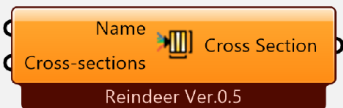
Component: Material

The material component inputs different timber-specific properties. Only structural properties are included but the toolkit is developed to be scalable, which means that it shall be trivial to expand the functionality and add the material-related properties of an architect, manufacturer or other discipline.



Component: Rectangular Cross-section

The current version of the toolkit enables rectangular cross-sections. In Reindeer, a unique cross-section is a combination of shape, material and shape.



Component: Composite

The composite component inputs cross-sections and merges them into one composite. A cross-section may be directly inputted into the element component but will still be treated as a composite with one cross-section.

Each part in the composite corresponds to the manufacturing sub-elements, which means that each part is assigned a unique ID and geometry and can be individually processed and outputted. This feature is crucial when manufacturing large glulam parts.

By combining cross-sections and local alignments in creative ways, many different composite cross-sections can be obtained. Figure 25, Figure 26 and Figure 27 illustrate a conventional glulam composite, an I-beam and a circular composite, respectively. The figures exemplify the power of modularising the inputs of the element.

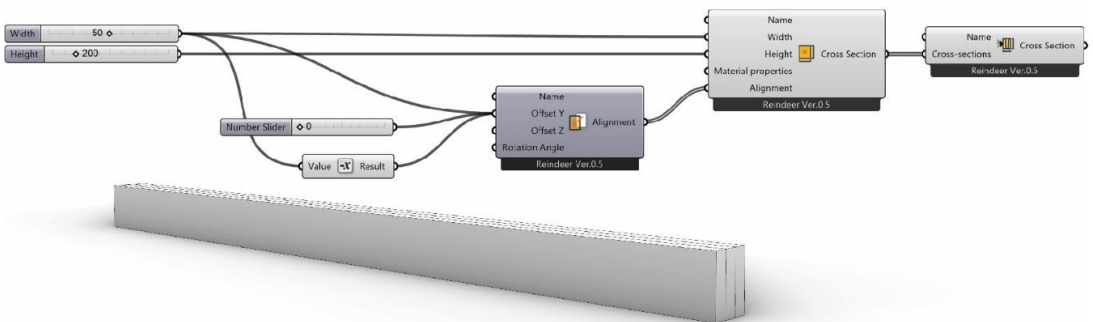


Figure 25: Conventional composite.

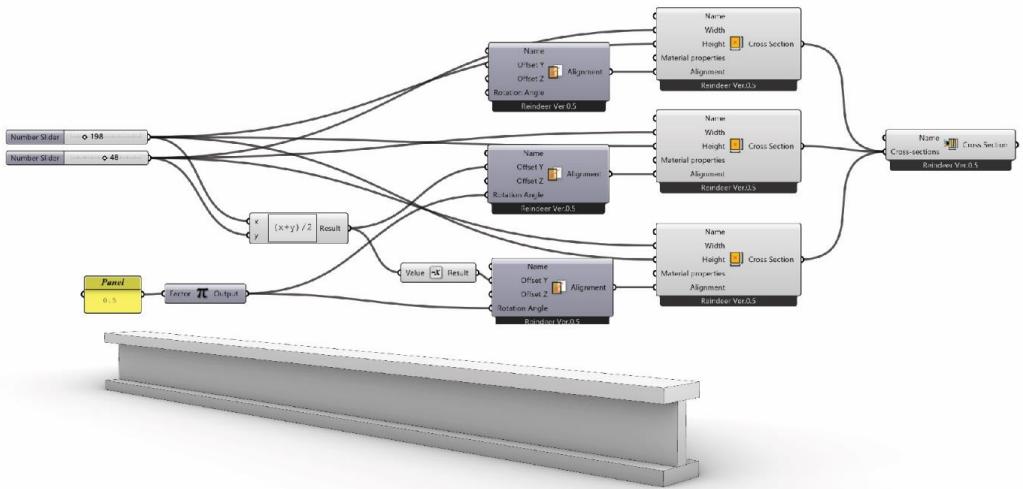


Figure 26: I-beam composite.

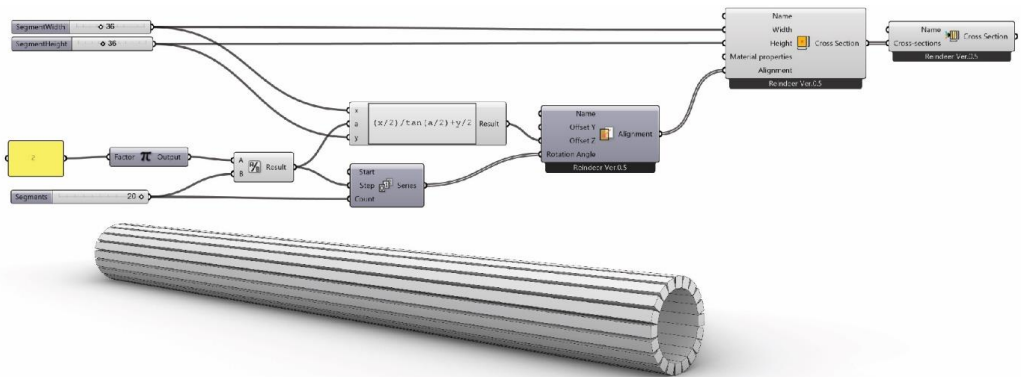


Figure 27: Circular composite. A functionality that was not intended while designing the toolkit.

Global Alignment

By default, the composite cross-section is positioned centric relative to the base curve, and the composite's height direction is aligned towards the global Z-direction. However, the default position and alignment is not feasible in many cases. Thus, the toolkit contains a series of global alignment components that manipulate Offset Y, Offset X, and the alignment.

The previously explained local alignment is transformed relative to the global alignment. The left side of Figure 28 shows a composite that is offset and oriented from the base curve. The right side of Figure 28 shows an additional local alignment.

The current version enables alignment relative to surfaces, vectors, points, and planes. Figure 29 shows different applications.

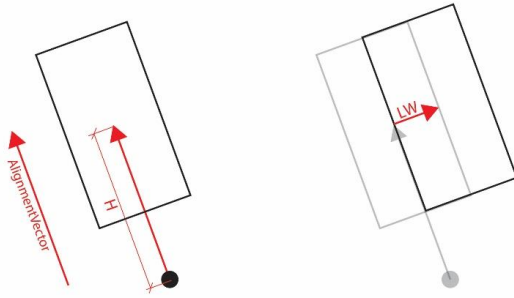


Figure 28: The left side of the figure shows that an alignment vector and the Offset Height globally align the element. The right side of the figure locally offsets the element in the Local Width-direction.

Component:
Normal vector from surfaces

Obtaining normal vectors from a guide surface has become common practice when designing advanced structures, such as shells. However, the principle can be equally powerful when designing conventional shapes. The essential functionality of this component is that it enables multiple surfaces to be input for one single element. The power is best explained using an illustration. Figure 29 B shows how the “height” direction of the column is always aligned towards the façade of the walls.

Table 2: GlobalAlignmentComponents. Refer to Figure 29 for a visual explanation.

Name	Inputs	Operation
A: AlignToVector	Vector Offset Width (Y) Offset Height (Z)	The element’s cross-section height direction aligns according to the vector input
B: AlignToSurface	Surfaces Max Distance Offset Width(Y) Offset Height (Z)	If it does not exceed the max distance, the element’s cross-section height direction aligns according to the closest surface that is normal to the element’s midpoint
C: AlignFromPoint	Points Curve Domain Offset Width (Y) Offset Height (Z)	The element’s cross-section height direction aligns according to the vector between the point on the element defined by the curve domain and this point’s closest inputted point.

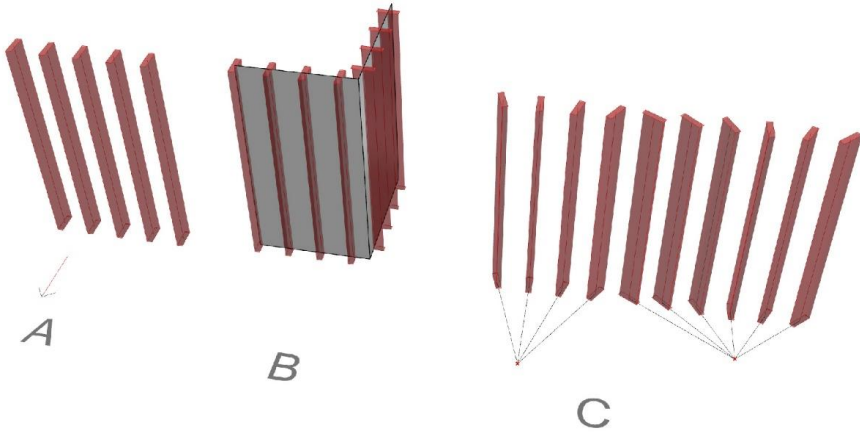


Figure 29: Three types of global alignments. A: From vector. B: From Surfaces. C: From points.

8.3. Step 2: Assemble the structure, define the geometric relations

The assembly component automatically generates nodes and joints based on the input elements. To be applicable for detailing, the end points and intersection points qualify as nodes. Figure 30 illustrates how a node is generated. A joint is a node and its elements and can be deconstructed into elements and nodes, and the elements can be deconstructed into the properties defined in step 1. Figure 31 shows how elements are transformed into joints, and Figure 32 shows how a joint contains essential information required for detailing.

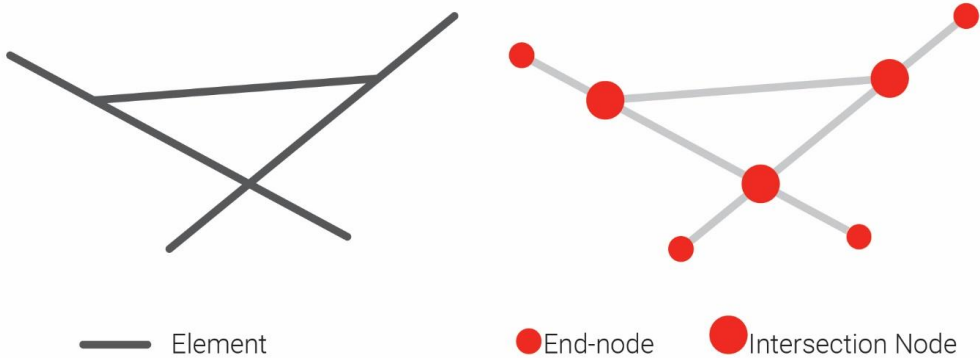


Figure 30 A node can be generated at element endpoints or based on element intersections.

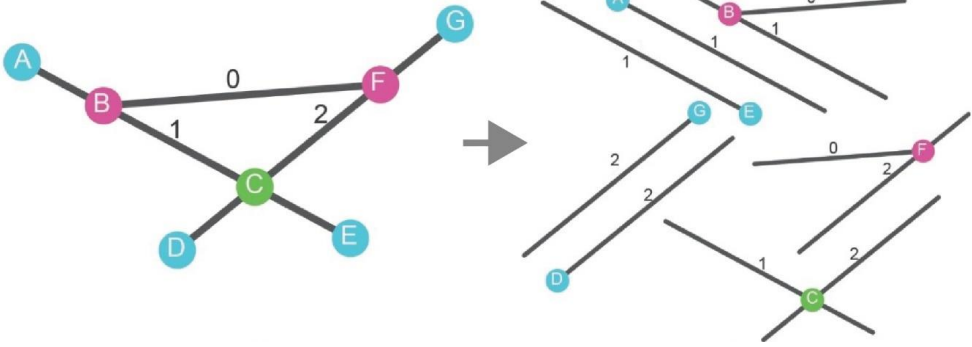


Figure 31: Node-focused joints and element focused joints. Current build supports only node-focused joints.

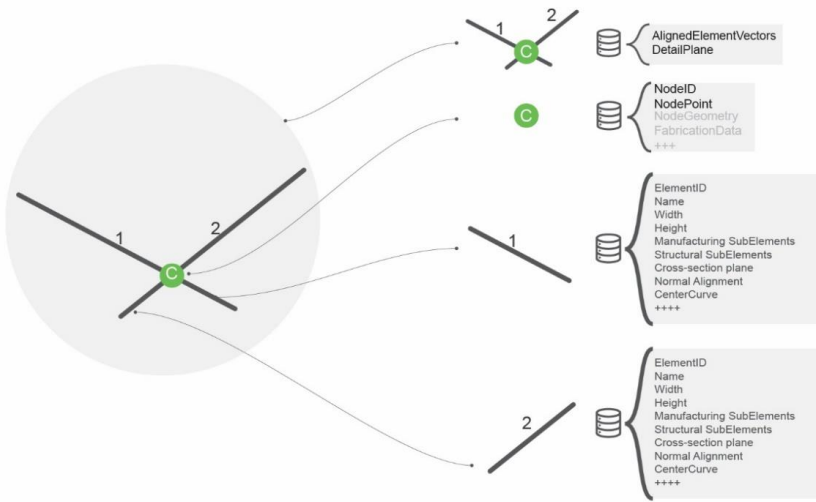


Figure 32: With the described relation among elements, nodes and joints, a person can select a joint, obtain its elements and node and then extract geometric data to apply to detailing.

8.4. Step 3: Structural analysis

The code at GitHub and previous versions have included structural analysis integration with Karamba3D. Due to bugs, this functionality is not publically available. However, the purpose of this step, in addition to analysis, is to enrich the element with structural results and use force results as search criteria and input for architectural expression.

8.5. Step 4: JointSearch and JointOutput

This step enables the user to define the joint types' solution space by using one or multiple search component criteria. These search criteria filter all joints and pass only the joints that fulfil all search criteria.

A solution space can be defined by properties, such as the number of elements, names of elements in a node, min/max internal angle and position of the node. Further, a solution space can be defined based on loads, for example, the maximum allowed normal force in the joint's connected element. However, defining a solution space by defining properties that shall not be true is equally effective. Thus, the user can input the search criterion components to be either true or false to be valid.

Figure 33 illustrates two shells that have three joint types: Foundation Node, Edge Node and Interior Node. Based on the name of the elements, the joints in both shells are assigned to its joint types.

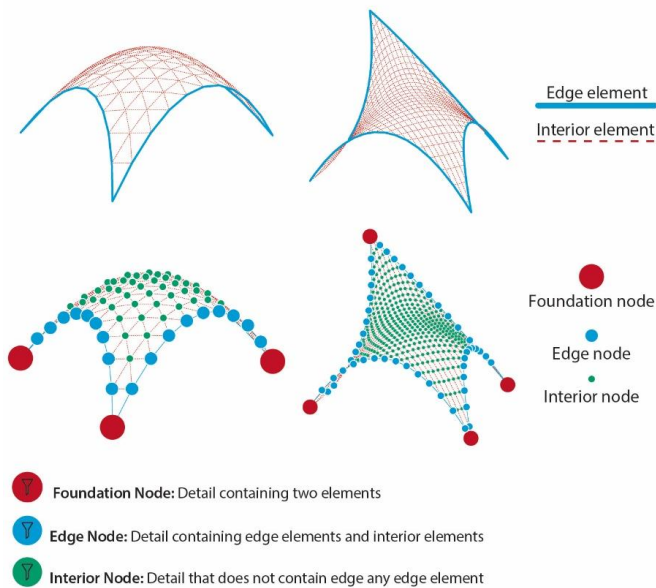


Figure 33 The three JointSearch criteria work regardless of shape or surface pattern.

An important property in one joint type may be irrelevant in another joint type. Some joints may allow a small variation in a property, while other joints allow all property variations—with the exception of a small domain. The relevant properties are dependent on the joint and parametric modelling strategy. Figure 34 and Figure 35 illustrate joint types based on various search criteria.

Name	Inputs	Checks
Element length	Min length Max length	Checks lengths of elements and returns true if all elements are within the allowed domain
Element angle	Min Angle Max Angle	Checks angle between each pair of elements and returns true if pairs are within the allowed domain
Element Amount	Min Amount Max Amount	Checks amount of elements and returns true if the amount is within the allowed domain
Element Name	Element Names Mode	Checks if the joint's element's names correspond to the names of the element. The mode input determines the strictness of the checker. Mode0: The joint's element's names must contain one of the inputted names. Mode1: The joint's element's names must contain at least all inputted names. Mode2: The joint's element's names must contain all inputted names and no other names. Mode3: The joint's element's names must correspond exactly to the inputted names
Detail topology	Mode	By choosing a mode, the checker finds joints that correspond to typical shape types. The following modes are included: Mode0: L-node: Two elements connected in an endpoint Mode1: T-node: One element is connected at the end, and one element is connected in the middle of the element (not at the ends) Mode2: X-node: Two elements are connected on the elements (not at the ends) Mode3: End node: Nodes with one single element connected Mode4: Star node: 3 or more elements connected at the end Mode5: Planar: All elements can be placed on a single plane Mode6: Orthogonal: All angles between elements in joint are 0, 90 or 180 degrees. $a\%(Pi/2) = 0$ Figure 36 illustrates the modes.
Node in region	Region Max offset distance	Checks if the point of the node is inside an inputted region. The inputted max offset distance determines if the point must be in the region or can be offset in the normal direction of the region.
Element Force components	Max force Min force	Checks if all elements are within the force domain specified by the user. Individual components for Torsion MX, Bending MY, Bending MZ, Compression FX, Tension FX, Shear FY, and Shear FZ.

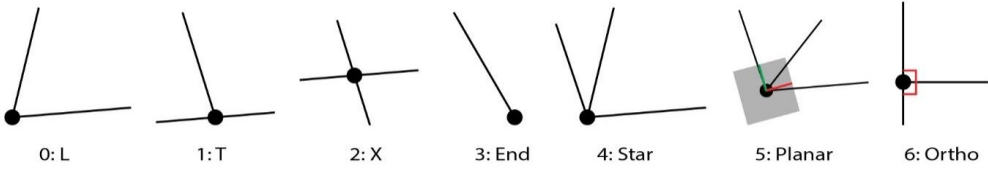


Figure 36: The joint topology component searches for joint topologies. These topologies are the currently supported topologies.

JointOutput

When detailing a joint, information about the related nodes and elements is needed. The more consistent the data that are streamed, the easier the detailing process. An inconsistency often demands codes to handle exceptions from a clear logic. Three functionalities are implemented to increase the consistency of the data stream.

1. The unified element vector outputs vectors that are parallel to the element and directed from the node.
2. The input named Sorting rule enables the user to choose sorting the elements by structural priority, alphabetically, element length or clockwise (relative to the node plane).
3. Node plane generators enable the user to apply different logics for defining a node plane. Figure 39 shows two examples. Table 1 describes the implemented components.

The JointSearch component outputs nodes, elements, unified element vectors and the node plane. Further, the node, element and sub-element can be deconstructed into their properties. In this manner, the properties inputted in step 1 are neatly organised and ready for detailing.

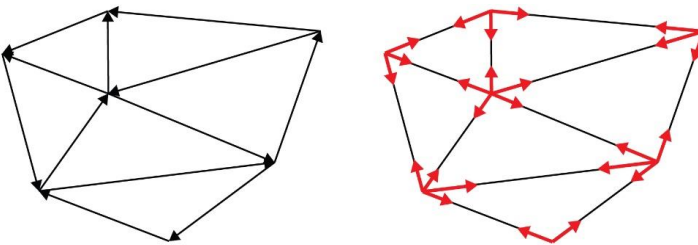


Figure 37: Unified Element vectors An element has a fixed direction based on the curve that is used when initialising the element. The Unified element vectors are node-based and directed from the node centre.

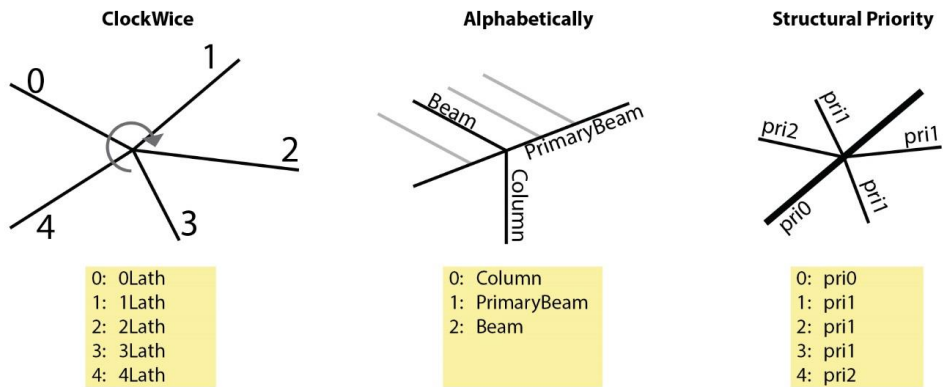


Figure 38: Element sorting: Three different sorting rules for outputting the elements.

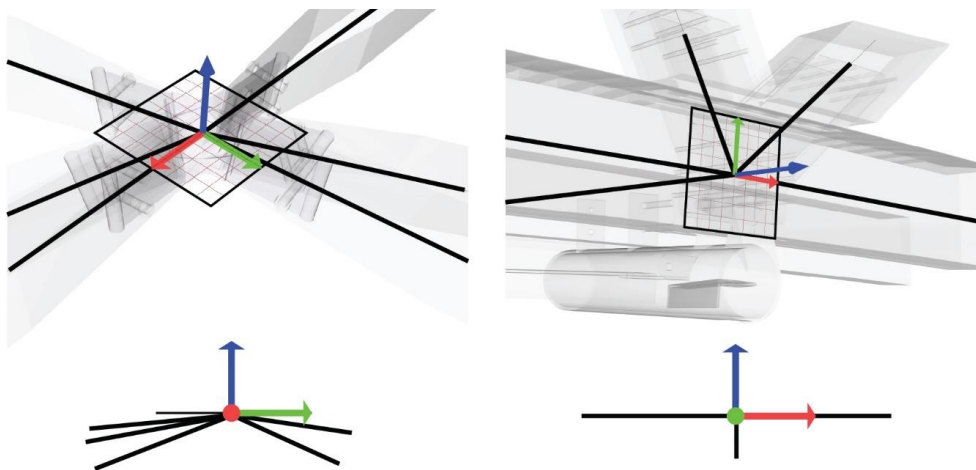


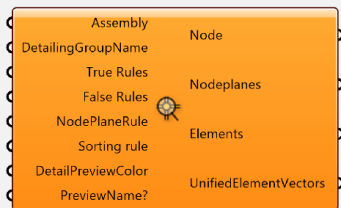
Figure 39: Node-planes: The left side of the figure shows a gridshell connection. In this case, a plane is generated based on the direction of the elements. Alternatively, a plane can be generated based on a guide surface. The right side of the figure shows a bottom chord node of a truss. In this case, the plane's normal direction is parallel to the secondary beam. The plane is aligned according to the direction of the chord.

Table 3 Node plane generators

Name	Inputs	Logic
NormalFromMesh	MeshGuide Name of X element	The normal of the mesh closest to the node determines the Z-axis of the plane. The X-axis is aligned according to the inputted X-element
NormalFromSurface	SurfaceGuide Name of X element	The normal of the surface closest to the node determines the Z-axis of the

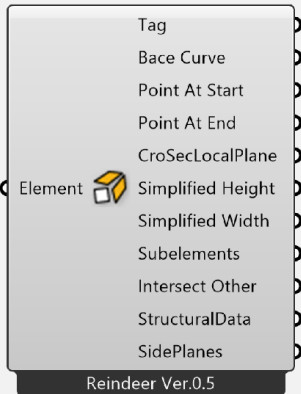
		plane. The X-axis aligns according to the inputted X-element
NormalFromElementAverage	Name of X element	For each element, a point is moved one unit-length from the node in the direction of the element. These points are used to create an average plane that has its origin in the node, and the X-axis is aligned according to the inputted X element
FromVector	NormalVector Name of X element	The inputted normal vector defines the Z-axis of the plane; the X-axis aligns according to the X element.
AlongElement	Name of Z element Name of X element	Z element is employed as Z vector of the plane; the X-axis aligns according to the X element.

Component: Detail Search



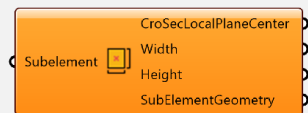
The previously described searching, extraction, and sorting of details are compiled as one component. The essential inputs are the assembly, false/true search criteria, selected mode of element sorting, and node plane generator. Additionally, the name of the joint and colour to preview the valid details can be defined.

The output is the node, node plane, elements, and the unified element vectors. Since only one node/node plane and multiple elements and element vectors exist in each joint instance, the data are outputted in a tree structure. The tree outputs all valid details, and each branch corresponds to one joint instance.



Component: Deconstruct Element

The element object is not usable for detailing purposes until it is deconstructed into its properties. The deconstruct element outputs a series of important geometrical data to simplify the detailing process.



Component: Deconstruct manufacturing sub-element

The deconstruct element outputs sub-elements. This component further deconstructs each sub-element into width, height local cross-section-plane, and BREP-geometry.



Component: Deconstruct node

Currently, the functionality of the node deconstruction is limited. Only the location of the node is deconstructable. However, future development will implement important data.

Detailing configuration

By applying the presented functionality, the user can select wanted joints, consistently sort the output and deconstruct the elements. The following figures show a gridshell that is built by interior and edge elements. The scheme shown in Figure 40 shows how the JointSearch identifies 3- or 4-legged interior joints by combining a search criterion that defines the element names that are not allowed and a search criterion that specifies the number of elements.

The left side of Figure 41 shows how the JointSearch outputs geometry by default. However, the joint plane is not applicable, and the order of the outputted elements is not consistent. The right side of the figure provides a better basis for detailing. First, the joint plane component has generated a plane based on the shell surface. The X-axis of the plane is aligned in parallel to one of the interior elements. Second, the elements are sorted clockwise according to their joint plane. Note that element0 is parallel to the X-axis.

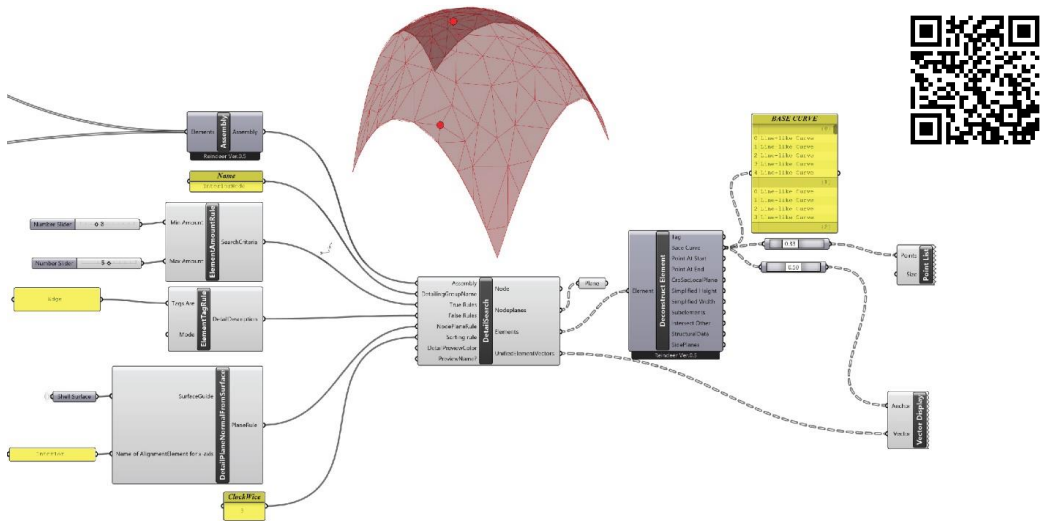


Figure 40: This figure shows how to output interior joints that contain 3 or 4 elements. The video in the QR code demonstrates the process of building the algorithm.

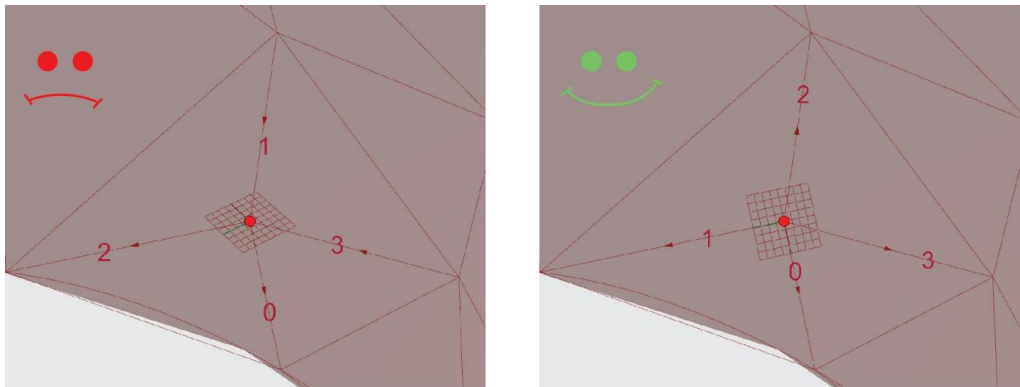


Figure 41: Default output and output prepared for gridshell detailing. The right figure shows unified element vectors and alphabetical sorting of the elements.

8.6. Step 5: Detailing the elements (and nodes)

Detailing a timber element using the Reindeer toolkit is only possible by applying subtractive operations named TimberProcessingTools. Drilling, cutting, pocketing, and tenon shapes are implemented components and operations that subtract material from a predefined blank. The output is primarily a NURBS element or BTLx instruction; Refer to Figure 42.

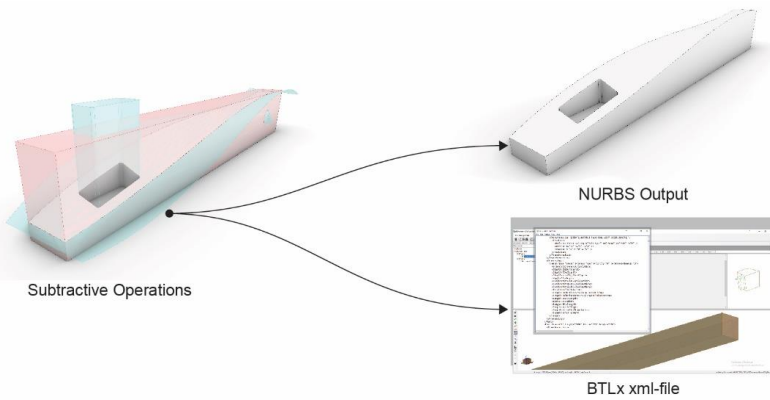


Figure 42: The user applies subtractive operations. While designing, Brep geometry is being outputted. When the design is ready, a BTLx file can be exported for fabrication.

Individual elements can be detailed without interacting with its joints. In most cases, however, detailing is performed by outputting elements from the JointSearch. In this process, the user can individually detail each joint type. Algorithmically, the following challenge arises: A unique element will certainly belong to multiple joints. For this reason, TimberProcessingTools output instructions regarding how an element is being processed. Step 6 assigns processing instructions to correct elements and process the model. For this reason, all timber-processing components are CPU light; the heavy calculations are performed in step 6, process assembly.

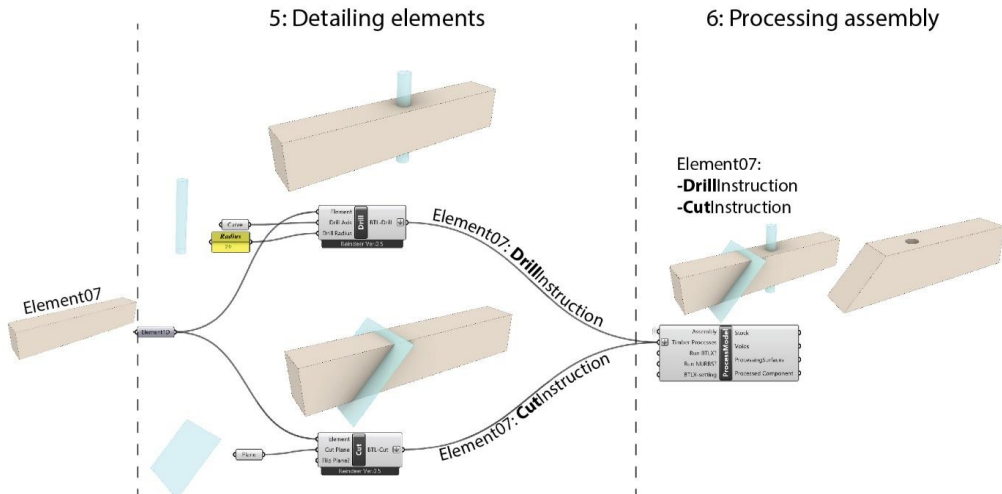


Figure 43: The TimberProcessingTools outputs a timber processing instruction.

Components: TimberProcessingTools

Currently, a limited number of processings are included. However, the algorithm is developed to be scalable, and adding operations is relatively straightforward. All components input an element and output a subtractive operation. The additional input that instructs the various operations varies. The following table presents the implemented timber processings.

Name of operation	Inputs (in addition to element to be processed)	Description
Cut	The cutting plane, Toggle to flip the direction of the cut	Cuts an element based on the inputted plane. Initially, the positive side of the plane is the part that will be removed. The toggle enables the direction to be flipped
Drill	Drill axis(line), radius	Creates a circular hole with the inputted radius and positioned according to the drill axis
Pocket	Parallelogram, angle1, angle2, angle3, angle4	Creates a pocket based on the parallelogram. The inputted angles define the angles between the sides and the parallelogram.
Pocket from element	Element	In many cases, a pocket is performed to make a slot for another element. This component uses another element as input for the pocket
Mortise	Mortise plane, width, length, depth, flip toggle, shape radius, shape mode	The mortise is generated based on the plane, width, height, and depth. Additionally, a radius/chamfer can be defined.
Tenon	Tenon plane, width, length, depth, flip toggle, shape radius, shape mode	The tenon is generated based on the plane, width, height, and depth. Additionally, a radius/chamfer can be defined.
Custom shape	BREP	To extend the usability of the Reindeer algorithm, inputting a custom BREP is possible. Note: This component only generates NURBS and is not included in the BTLx file.

8.7. Step 6: Processing assembly (Generate NURBS or CAM output)

This step inputs the assembly and timber processing. The primary outputs are a processed Brep assembly and a BTLx file. Additionally, the blank, voids, and processing surfaces are outputted as an organised tree structure. Latter outputs are purposed for custom fabrication, such as robotic milling directly from the Grasshopper environment. Figure 44 illustrates the essential outputs. Disabling the BTLX/Brep output accelerates the algorithm.

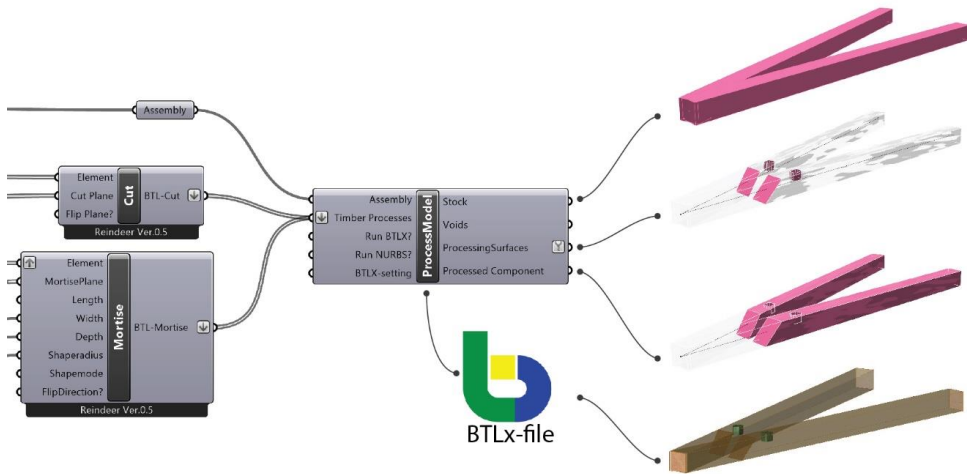


Figure 44: An element is detailed by a cut and mortise. The output is the blank, voids (not illustrated), processed surfaces, processed component and BTLx file.

8.8. Step 7: Manufacture and build

This stage is beyond the domain of the Reindeer plugin. The BTLx format is readable by a series of CAM software, including CAD WORK and SEMA [75].

The separated output of the blanks, voids, and processed surfaces intends to be applicable to custom fabrication either by using the Rhino/Grasshopper interface and plugins such as HAL Robotics or custom digital chains to other CAM-software.

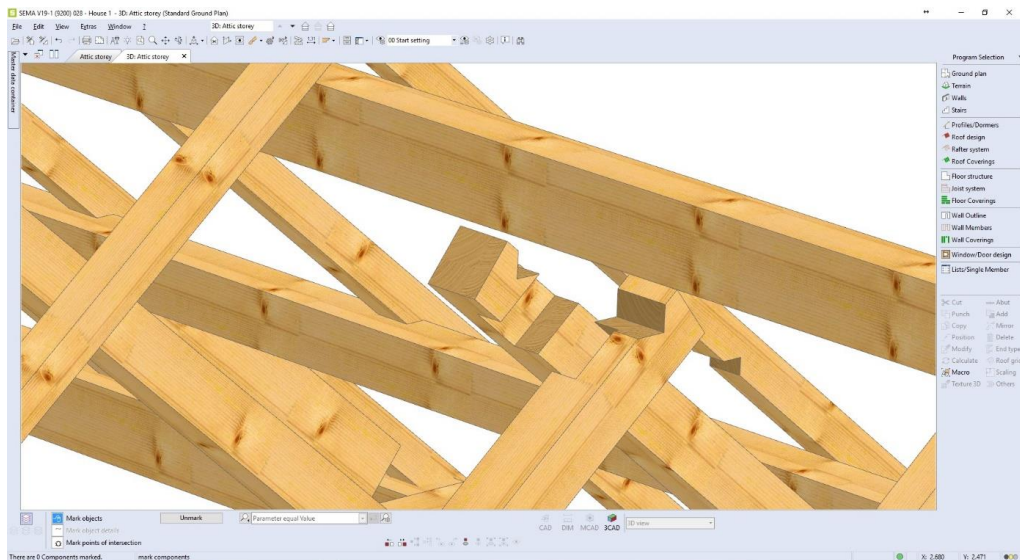
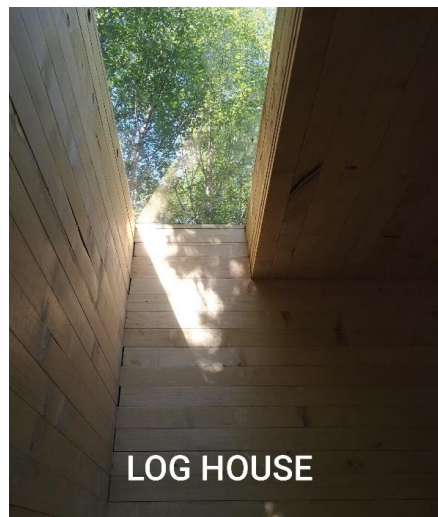
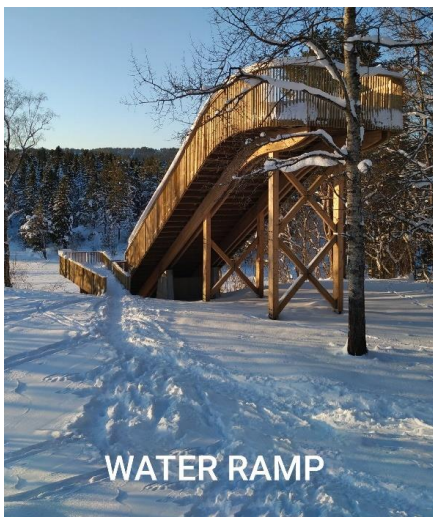
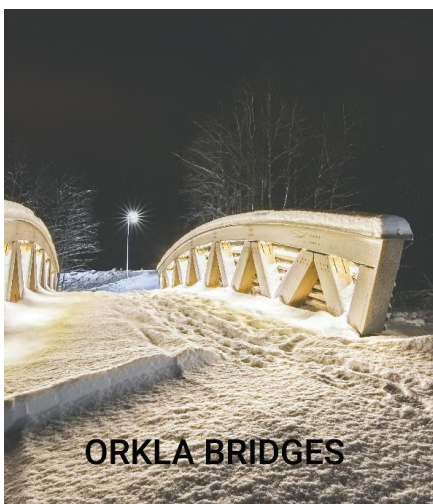


Figure 45: A timber structure generated using Reindeer, outputted as BTLx and imported by SEMA.

9. Case-Structures

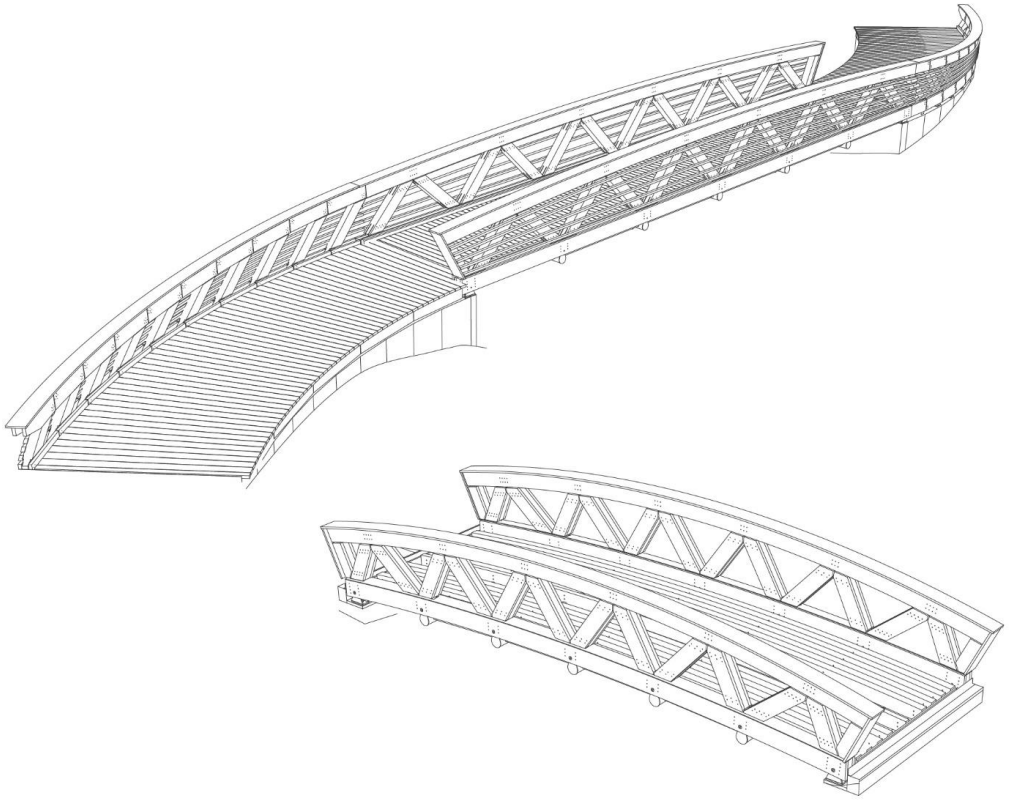
Designing, manufacturing, and building structures are essential components of this study. This chapter presents each case structure, which is organised in the seven steps workflow. Many of the generalised concepts were initially one-off solutions to practical problems, which are described in the way in which they were solved. The end of each case-project chapter reveals how these problems were generalised and implemented as a functionality in the Reindeer toolkit.

The cases are chronologically presented when they were initiated but do not necessarily correspond to when they were completed. In addition to these four projects, three smaller experiments are shown: projects that do not deserve a full chapter but that exemplify aspects of the study that are not addressed by the four main case structures.





ORKLA BRIDGES



9.1. Orkla Bridges

The Orkla Bridges are permanent and part of a pedestrian path along the Orkla River. The bridges pass two side rivers, which span 10 and 15.5 metres. Although pedestrians are the main users of the bridges, the bridges had to be dimensioned for an 18-ton wheel loader (15-ton axle load) due to snow removal purposes.

The structural system consists of two glulam trusses with a suspended secondary steel structure that carries a timber deck. Dowels and slotted-in-steel plates constitute the detailing system. The shortest bridge has a classical arch shape, and the longest bridge has connected doubly curved railings.

With the exception of the foundation, both bridges are entirely designed by applying parametric modelling. The steel structure was modelled by Grasshopper-native components, while the timber elements were designed using the toolkit version named “0.2 Timber Toolkit”.

Location	Orkanger
Type	Permanent structure
Completion	2017
Design period	October2015->February2017
Design Team	John Haddal Mork, Marcin Luczkowski, Steinar Hillersøy Dyvik
Other collaborators	Moelven Glulam, SINTEF
Project owner	Orkdal Municipality

Integrated

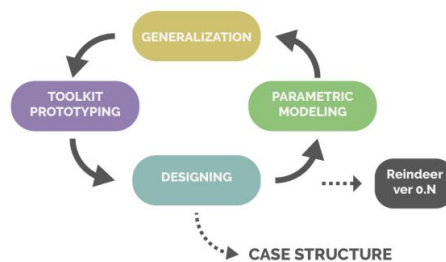


Figure 46: The Orkla Bridges project applied an integrated activity configuration. The activity configurations are explained in the methodology chapter.

Step 0: Constructing wireframe geometry.

The input for the Reindeer plugin is the centre curve of the glulam elements. Following list is a brief description of the logic that generates the curves.

- 1) Two curves are manually modelled. These curves represent the bottom centre curve and top centre curve.
- 2) The curves are divided into the desired number of point.
- 3) According to the desired width, the planes are translated transversal to both sides.
- 4) Chords, bars, and a secondary structure are generated based on the planes. Additionally, a tertiary structure, deck, and other geometry are generated based on the same logic but are not illustrated in the figure.
- 5) The centreline geometry is stored in the lists of Top Chords, Bottom Chords, Bars, Secondary beams and Zero-force bars.

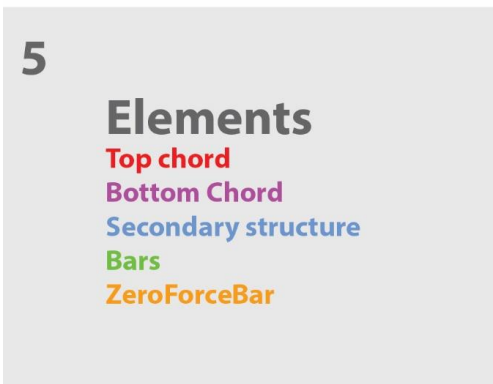
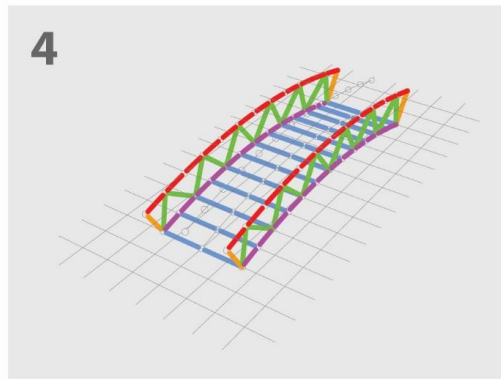
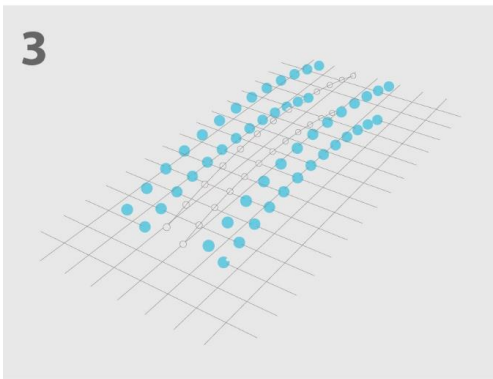
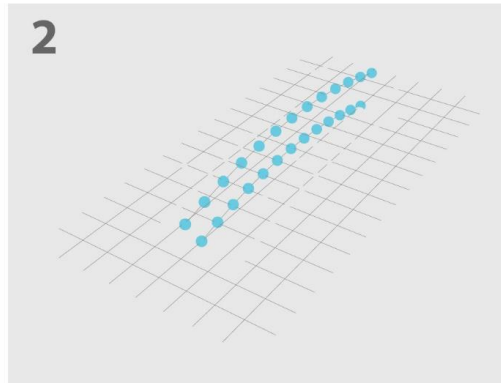
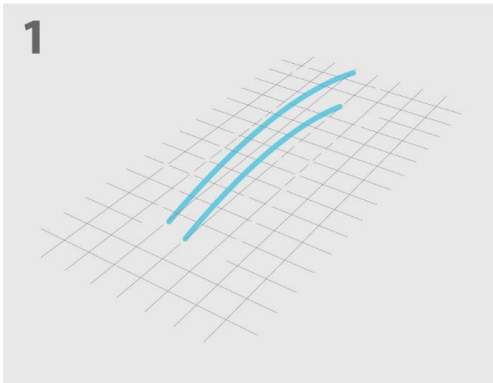


Figure 47: Construction of the wireframe geometry.

Step 1: Defining elements

The chosen detailing system highly influences the manufacturing strategies. The chords and bars are slotted to make space for the steel plates. The slot can be processed from the top, and a CNC chain-saw can be applied. However, a more conventional method is to split the element into sub-elements that are individually processed and then glued together. The latter method was chosen, which enabled milling the slot from the side.

The chords were defined as a composite constructed with three sub-elements. The steel-plates are placed between the sub-elements. Figure 48 shows the bottom chord composite.

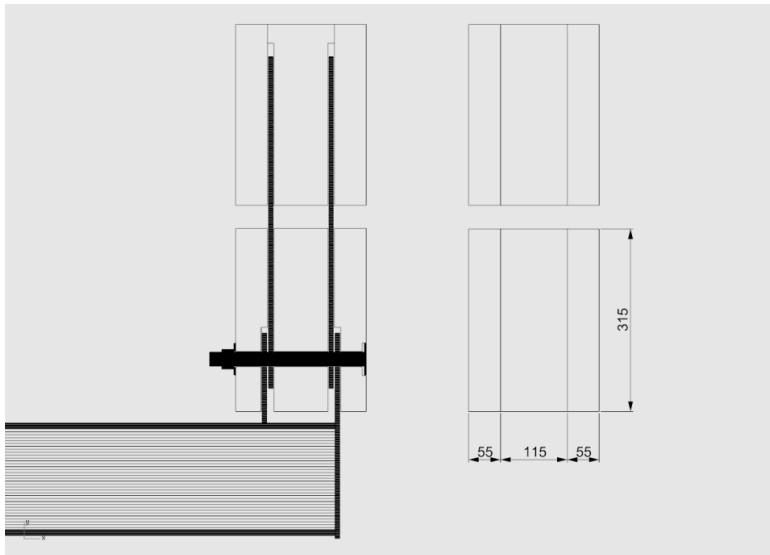


Figure 48: The bottom chord and bar elements were constructed by three sub-elements.

Step 3: Structural Analysis

Karamba 3D was used to explore plausible truss shapes. The shape of the top curve of the Evjen Bridge was developed by exploring plausible shapes. However, a more thorough analysis was performed by a specialised analysis software.



Figure 49: Rather late in the design process, the top chord shape was changed due to structural requirements.

Step 4: Searching for joints

The bridges consist of a series of joint types. However, to explain the chosen solution, the identification of the four main joints are described: Bottom Chord, Top Chord, Foundation, and Zero-Force joint types. The next sub-chapter will explain the bottom chord joint type in more detail.

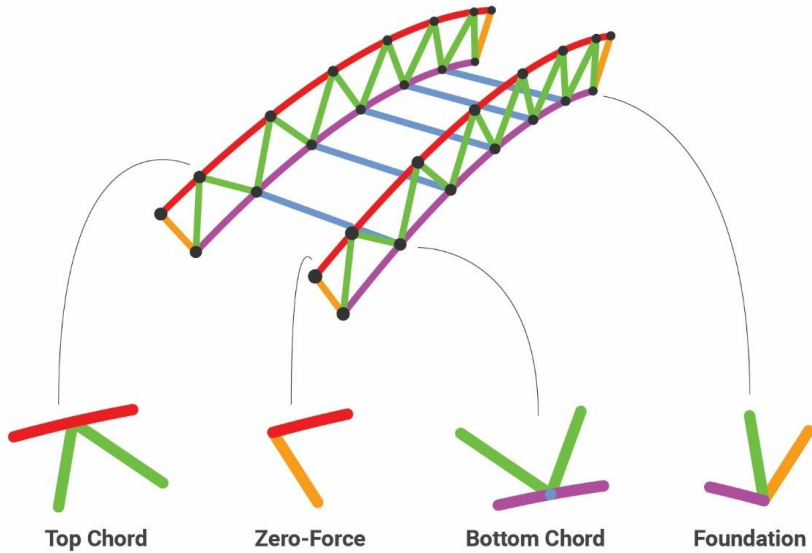


Figure 50: Parametric joint types.

Name	True Rules	Node Plane Generation
Top Chord Connection	Contains a Bar Contains a Bar Contains a Top Chord	Plane normal from a custom vector parallel to the secondary beam and directed towards the centre of the bridge. X-alignment that corresponds to the direction of the chord.
Zero Force Connection	Contains a top Chord Contains a Zero-Force Element	Similar to Top Chord Connection
Bottom Chord Connection	Contains a Bar Contains a Bar Contains a Bottom Chord Contains a Secondary Beam	Similar to Top Chord Connection
Foundation	Contains a Bar Contains a Zero-Force Element Contains a Bottom Chord	Similar to Top Chord Connection

Step 5: Detailing elements and nodes.

This step is illustrated using the most complex joint type—the bottom chord connection. Dowels and a custom steel plate connect the two bars and the chord. Additionally, the detail also contains the secondary, circular, steel beam suspended by a bolt in the steel plate. The principle of suspending the secondary beam is a well-known concept that is mostly applied in larger structures. In these structures, the pinned connection is placed below the bottom chord. To reduce the height of the structure, the pinned connection is elevated and hidden in the bottom chord.

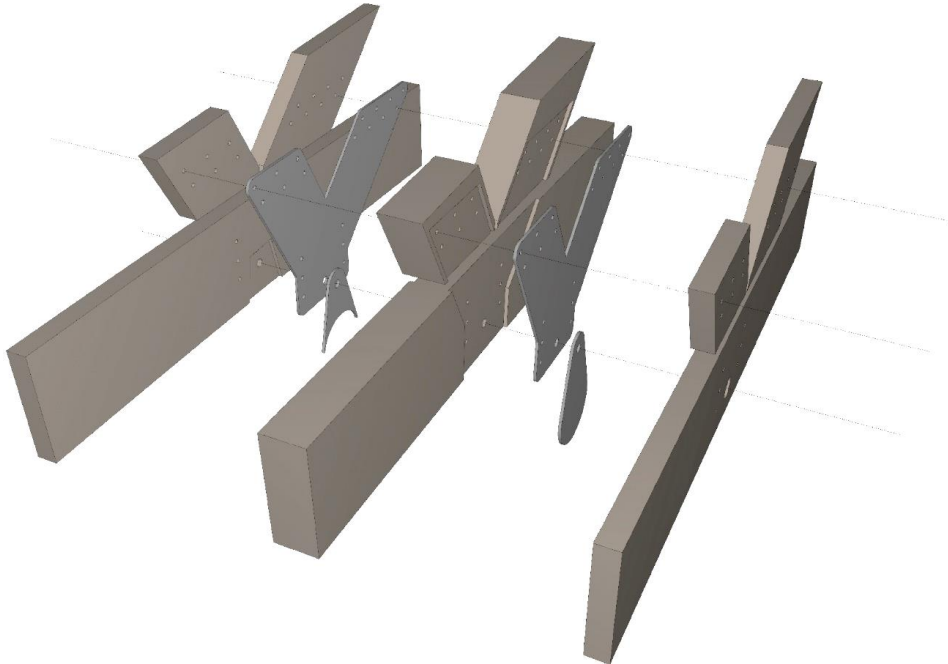


Figure 51: Explosion model that shows the manufacturing sub-elements.

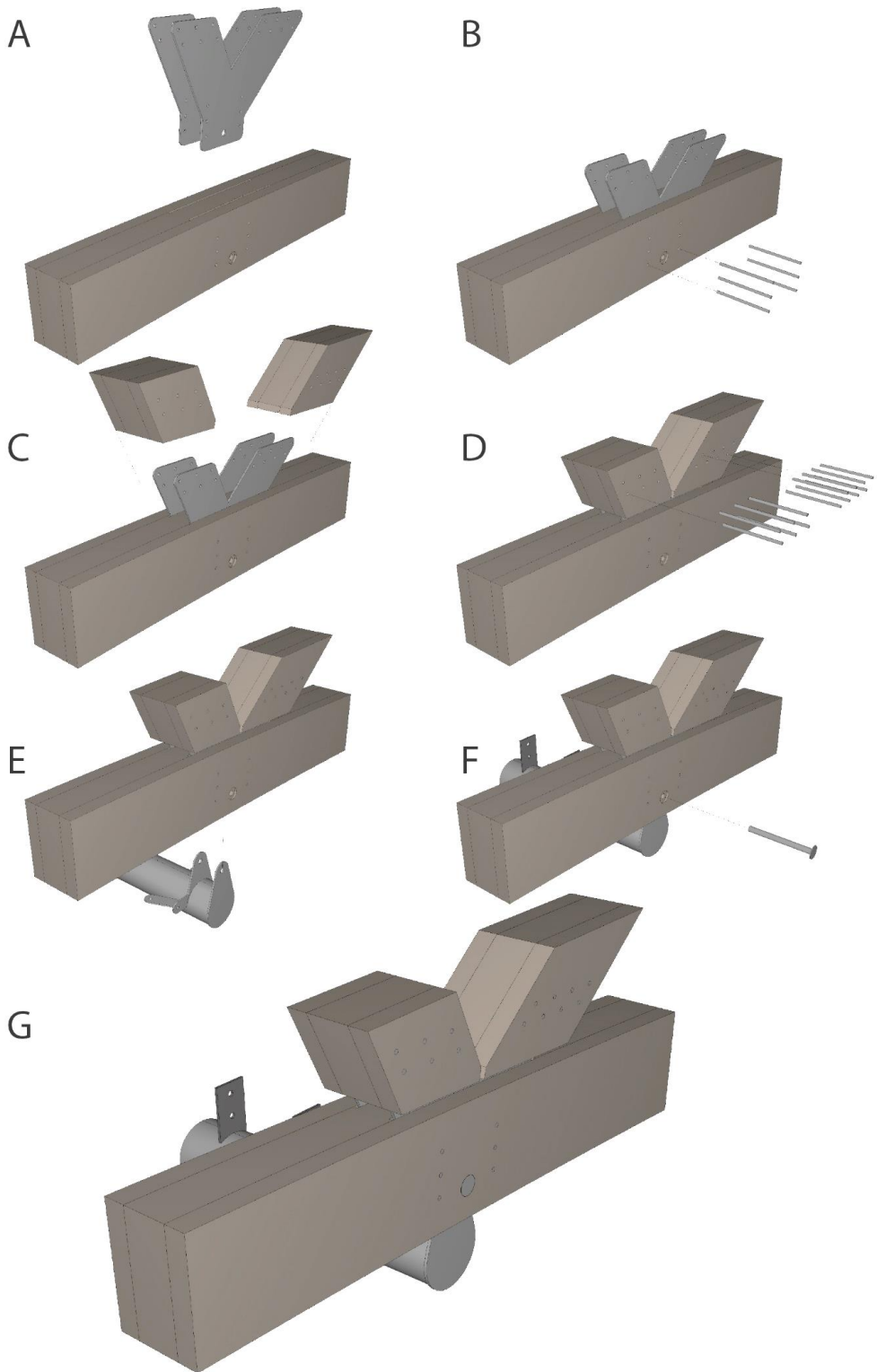


Figure 52: Assembly order of the bottom chord detail

Since the detailing system is well known, knowledge-based engineering (KBE) could be applied. Distances from the bar-edges to the dowels (X/Y-direction), minimum distances between two dowels (X/Y-direction) and dowel dimensions are examples of standardised parameters that are dependent on the joint's forces. These parameters were implemented in the model to instruct the geometry generation. All joints are geometrically unique but topologically similar. Closer to the foundation, the number of dowels had to be increased; refer to Figure 53.

Both the bars and the chords are detailed solely by applying drill holes, pockets, and cut processings. As explained in Figure 53, all holes for the dowels are drilled. The bars are cut in two directions, and the space for the metal plates are pocketed. Note that while the rise of the arch varies, the direction of the suspension remains vertical.

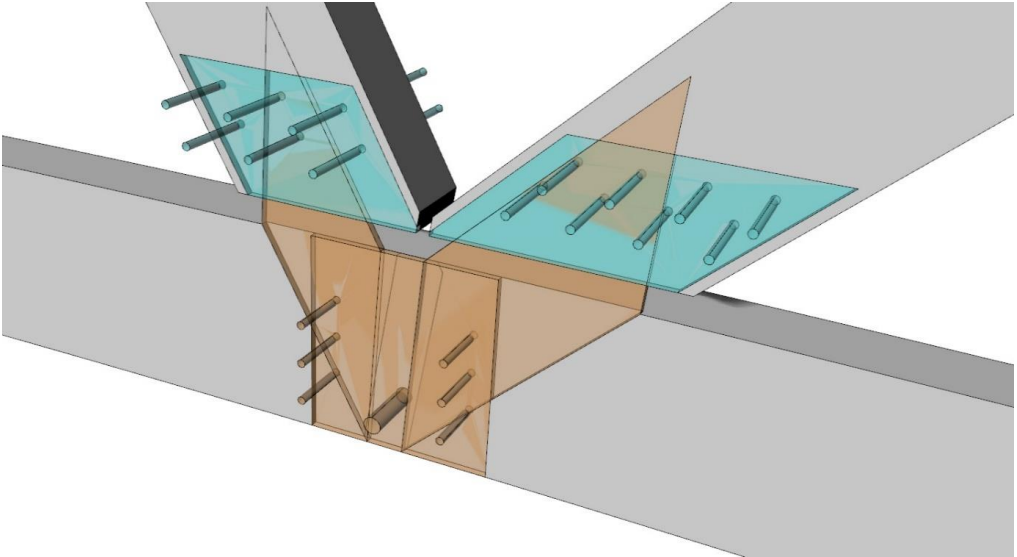


Figure 53: Timber processings. The double-cut processings of the bars are not included in the illustration.

Step 6: Processing assembly

The manufacturer's CAM software was CAD WORK, which enabled transfer of the model via BTL. When the model was transferred, the manufacturers applied a post-processor to convert the generic processings into machine-specific instructions.

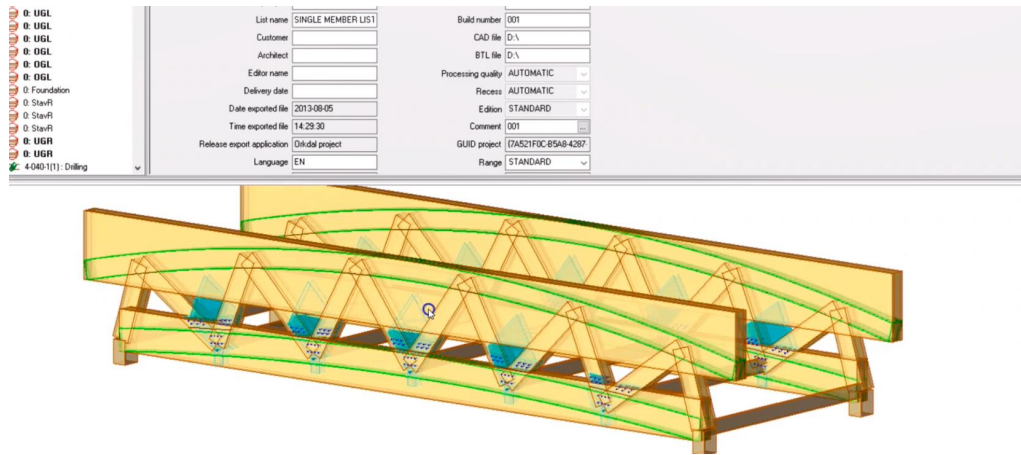


Figure 54: The BTLx-file of Follo Bridge.

Step 7: Manufacture and build

With the exception of the curved elements, a Hundegger Speedcutter manufactured all elements. The curved elements were milled by a larger CNC machine. The steel elements were cut using CNC and the trusses were preassembled to reduce the amount of work at the building site.

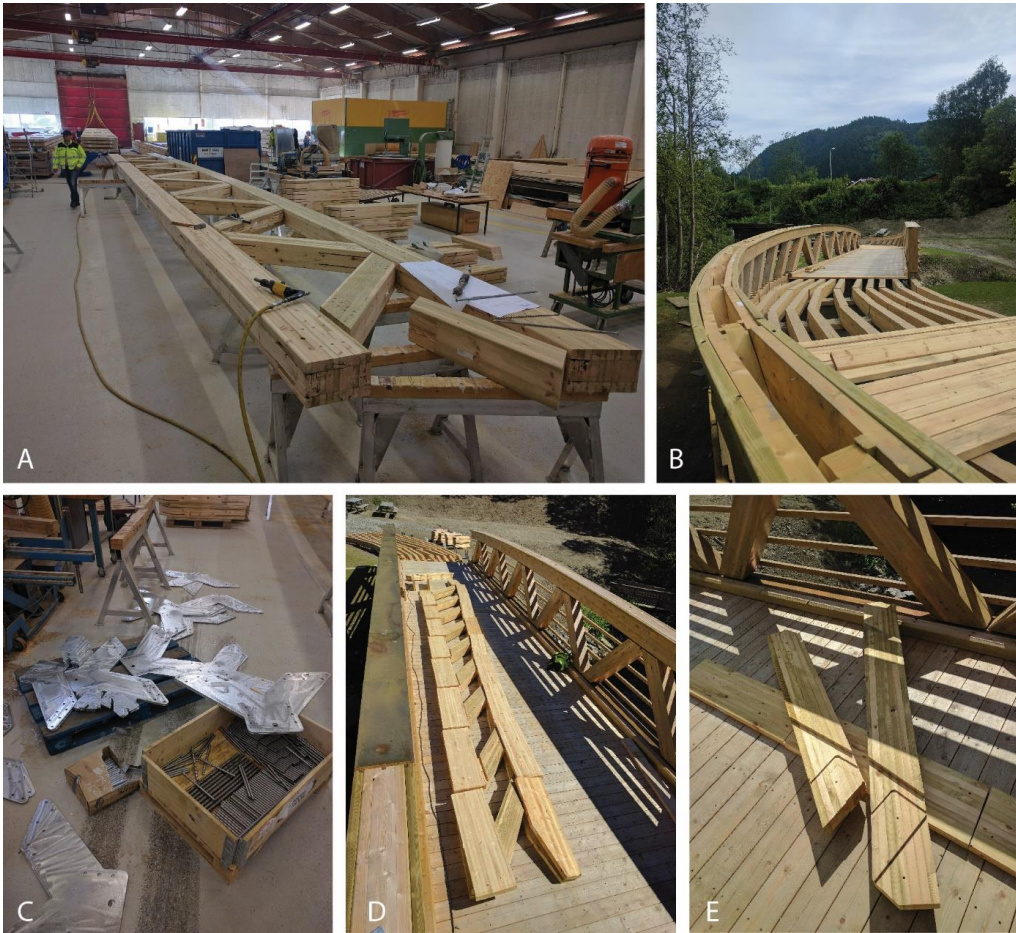


Figure 55: Fabrication and assembly of Evjen Bridge.

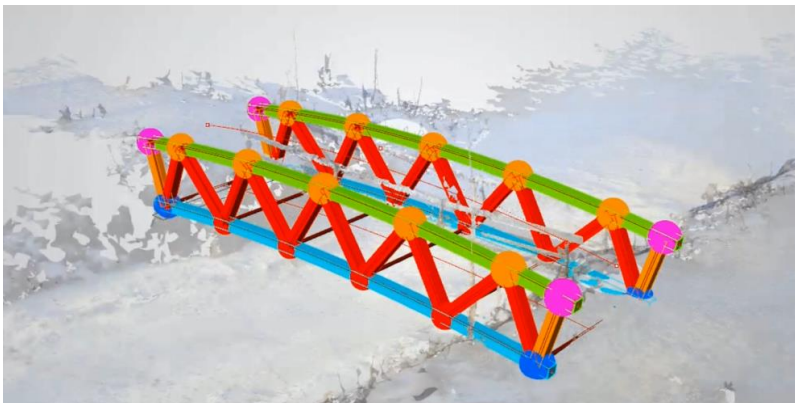


Figure 56: The video shows how the joints are being identified while sketching. Further, the video shows the Bottom Cord Joint Type.

Result



Generalisation of the case structure

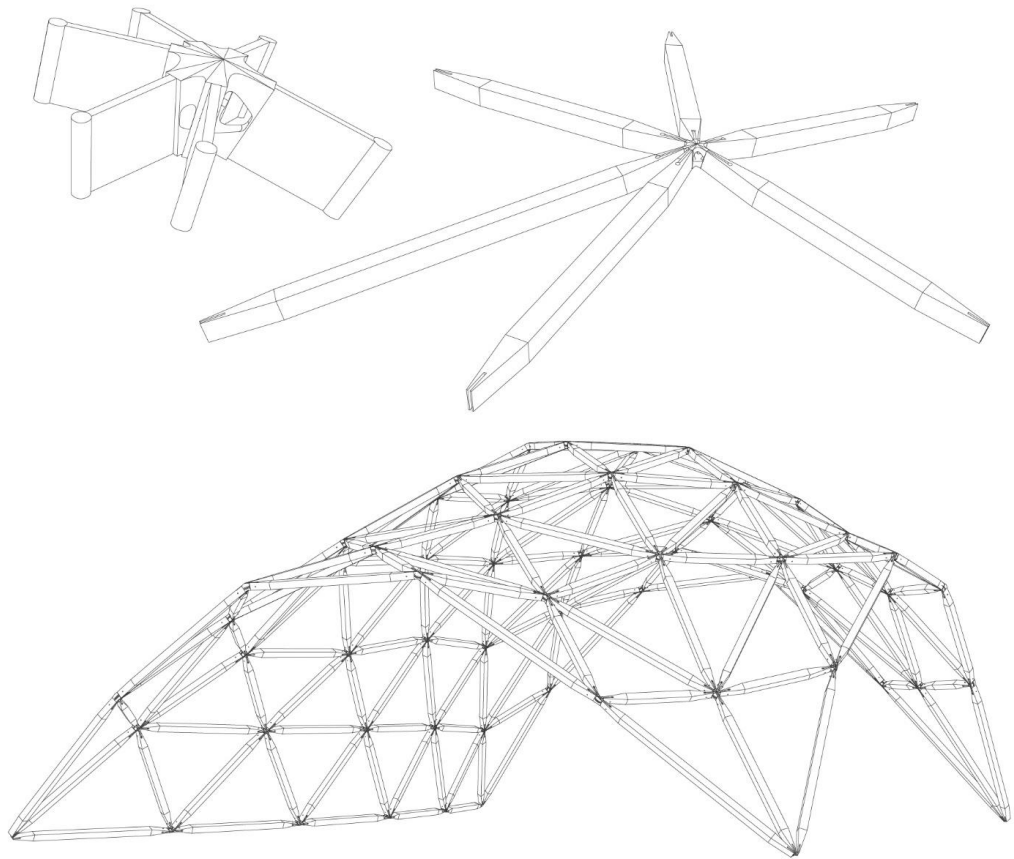
The Orkla Bridges have been the key informant to most concepts developed in the Reindeer plugin. The next case structures were mostly refined and expanded using previously developed concepts. Following is a list of generalizations:

- **Conception of an element:** The focus given to different stakeholders' conceptions of an element was triggered by designing the Orkla Bridges. The manufacturer demanded a 3D model that contains each individual sub-element and promoted the development of the composite functionality.
- **BTL:** Similar to the composite, implementation of BTL was advanced by the demand of the manufacturers. However, the design team simultaneously needed to preview a developed design. Thus, TimberProcessingTools was developed.
- **DetailSearch:** When creating a relationship between elements and nodes, identifying nodes and their elements became a laborious process. The solution was to create a node-oriented list structure of the curves that represent the elements, names of the elements, and nodes. By checking the names in each instance, identifying the branch that belongs to each joint was possible. As described in the Reindeer chapter, this process has evolved into a universal search based on various properties in a joint.
- **Slow performance:** Use of the toolkit became extremely difficult at the end of the project due to constant lags caused by poor programming. The toolkit required a total remake, and at this stage, rebuilding the toolkit using C# was decided.

-



PRINTSHELL



9.2. Printshell

The Printshell project was designed, manufactured, assembled, and disassembled in a month. Printshell was a small pavilion that was designed as a gridshell structure and exhibited at the Trondheim Maker Faire. Initially, the goal of the project was to explore the potential of 3D-printed gridshell nodes; however, it became an important case project for the Reindeer toolkit. This project was carried out in the middle of the Orkla Bridge project. While designing the gridshell, we realised large similarities to the process of building a bridge. In the bridge, a parametric relation had to be established between the connections and the bars, chords, and secondary beams. A similar relation had to be established between the nodes and the laths, which enabled us to realise that both projects could be generalised to be described by elements and nodes. Thus, this project is an important milestone in this study: the conception of similarities flips. Parametrically, a gridshell is more similar to a truss bridge than a gridshell is to a solid shell; Refer to Figure 57.

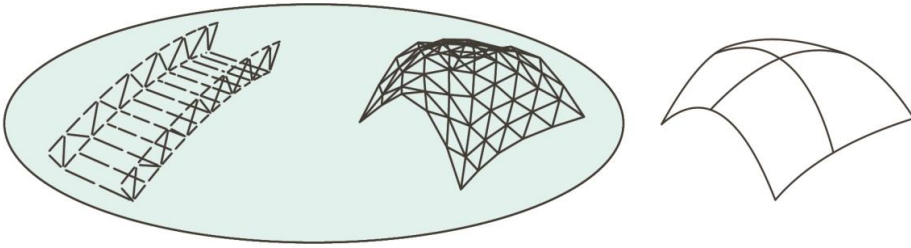


Figure 57: As we are discussing a structure's parametric system and logic, a gridshell is more similar to a truss bridge than to a solid shell.

Location	Solsiden, Trondheim
Type	Temporary pavilion
Completion	2016
Design period	August 2016
Design Team	John Haddal Mork, Marcin Luczkowski, Steinar Hillersøy Dyvik
Other collaborators	Addlab Gjøvik
Project owner	CSDG experiment

Detached

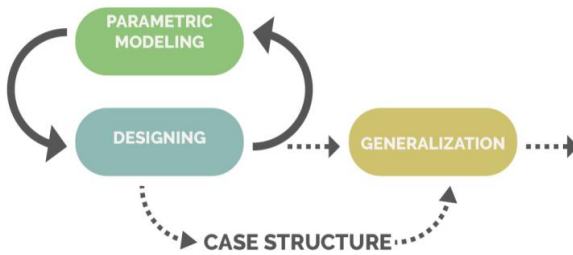


Figure 58: Printshell was designed via a detached activity configuration. The activity configurations are explained in the methodology chapter.

Step 0: Constructing Wireframe geometry

The input to the toolkit was the centre curve of the laths in the shell structure. The following text provides a brief explanation.

1. Guide curves were manually drawn.
2. A network surface was generated from the curves.
3. The surface was converted to a gridshell by applying a pattern.
4. The shape was optimised by dynamic relaxation and reducing the element's lengths to be shorter than the manufacturing limitations of 1200 mm.
5. All laths were stored in one list.

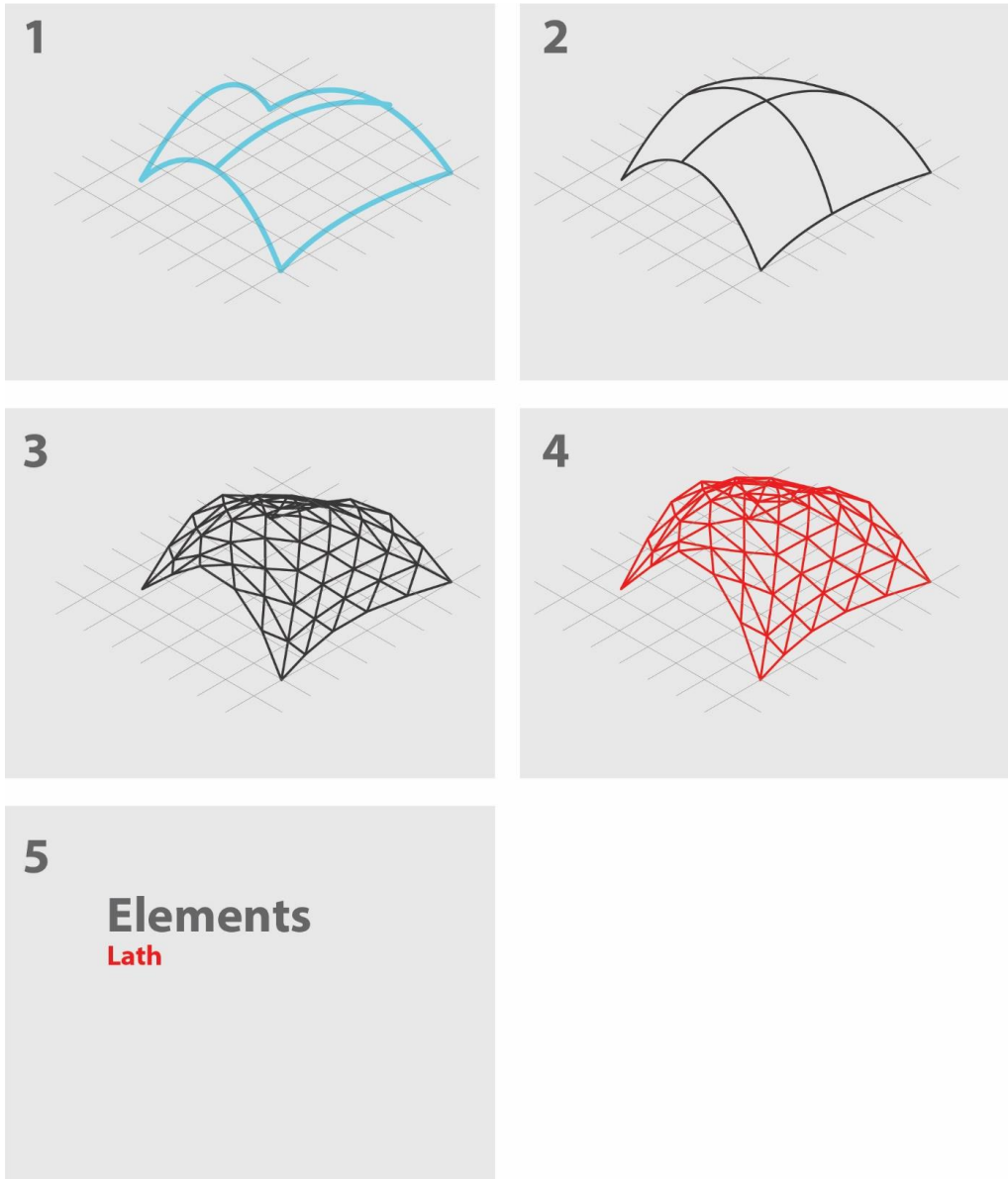


Figure 59: Construction of the wireframe geometry.

Step 1: Defining elements

The Orkla bridge project had a complex composite but a straightforward alignment: the Z-axis (height) of all elements were aligned according to the global Z-axis. The alignment in the Printshell case was opposite: a simple cross-section of 48×48 mm but all elements had to be individually aligned following the curvature of the shell. More precisely, the Z-axis of the bar is parallel to the normal vector of the surface evaluated at the point closest to the centre point of the lath.

Step 4: Searching for joints

The Printshell project had only two joint types: the foundations and the shell node joints. The primary reuse of the Orkla bridge algorithm was the component that established the relation between the nodes and the elements. Since all joints contained a “bar,” no existing solution distinguished the foundations and the shell node joint types.

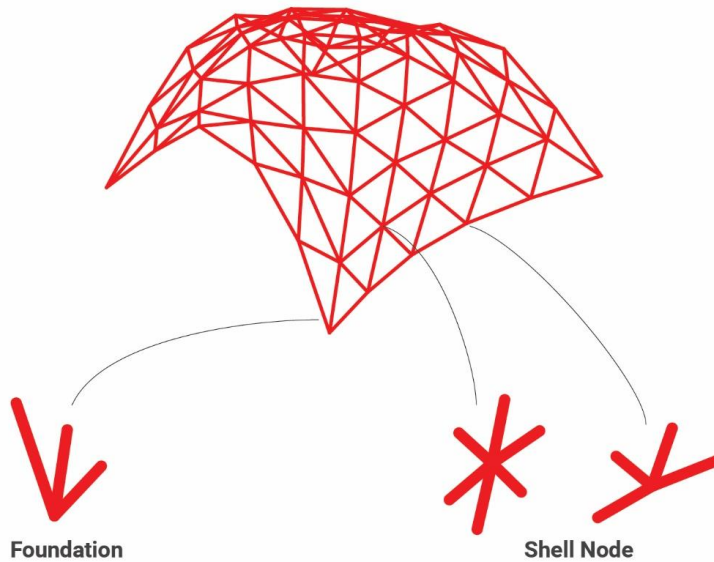


Figure 60: Parametric joint types. NB: Refer to the following description: the foundations were manually dispatched from the shell node list.

Name	True Rules	Node Plane Generation
Shell node Joint	Contains a Bar	Plane normal from the normal of a plane generated from the average of points on the elements x mm from the node.

All joints were outputted from the same data stream but a prototyped algorithm that checked the number of elements in the joint dispatched the foundation joints.

Further, when extracted, the order of the elements was random. To detail the connection, the elements had to be consistently sorted in clockwise order relative to the node plane.

Step 5: Detailing elements and nodes.

In this project, the geomtric complexity was absorbed by the 3D-printed connection. The length was the only variable in the element: the theoretical length of the curve with a fixed size (that corresponded to the size of the connection) subtracted on both sides.

The connection's shape was determined by the direction and alignment of the elements. An element-aligned "puzzle-shaped" arm is the interface between the node and the element. Each neighbouring arm was organically morphed. To reduce the print volume, material is removed from the centre of the connection. The centric star aligns to the node plane. These simple rules create a large variety of connection instances.

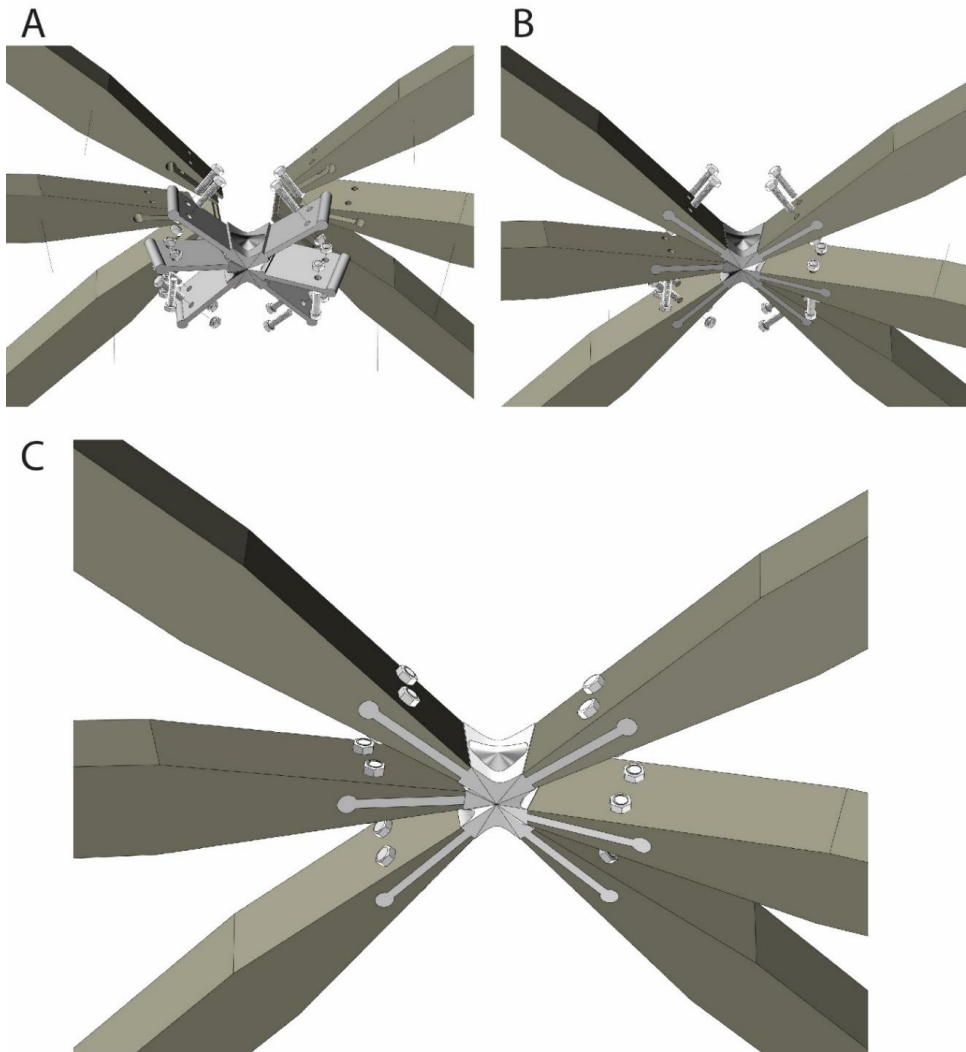


Figure 61: Explosion model of the shell node joint type. Each lath has identical geometry at the ends. Thus, the complexity is solved by the node's geometry.

Step 7: Manufacture and build

The elements were manually processed in a workshop. The nodes were printed using an advanced nylon 3D printer.

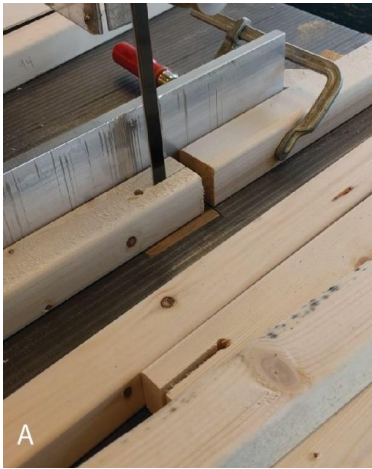


Figure 62: Low-tech manufacturing of the timber elements. High-tech manufacturing of the nodes.

Results



Generalisation of the case structure

- **Element alignment:** This project proved that it was crucial to creating a flexible element alignment either by components such as the AlignElementToSurface-component or inputting a custom element normal vector.
- **From name-filter to search-rule:** At this stage of the development, DetailSearch was only an algorithm that filtered names. This project showed that the toolkit would be more flexible by adding other search criteria, such as the number of elements. If the project were designed with the current version of Reindeer, several ways to identify the joints would exist. The following two tables illustrate possible identification strategies.

Name	True Rules	False Rules
Shell node Detail	Has at least 3 elements	-
Foundation	Has two elements	-

Name	True Rules	False Rules
Shell node Detail	-	The node is close to the ground region (a closed curve on the ground plane)
Foundation	The node is close to the ground region (a closed curve on the ground plane)	-

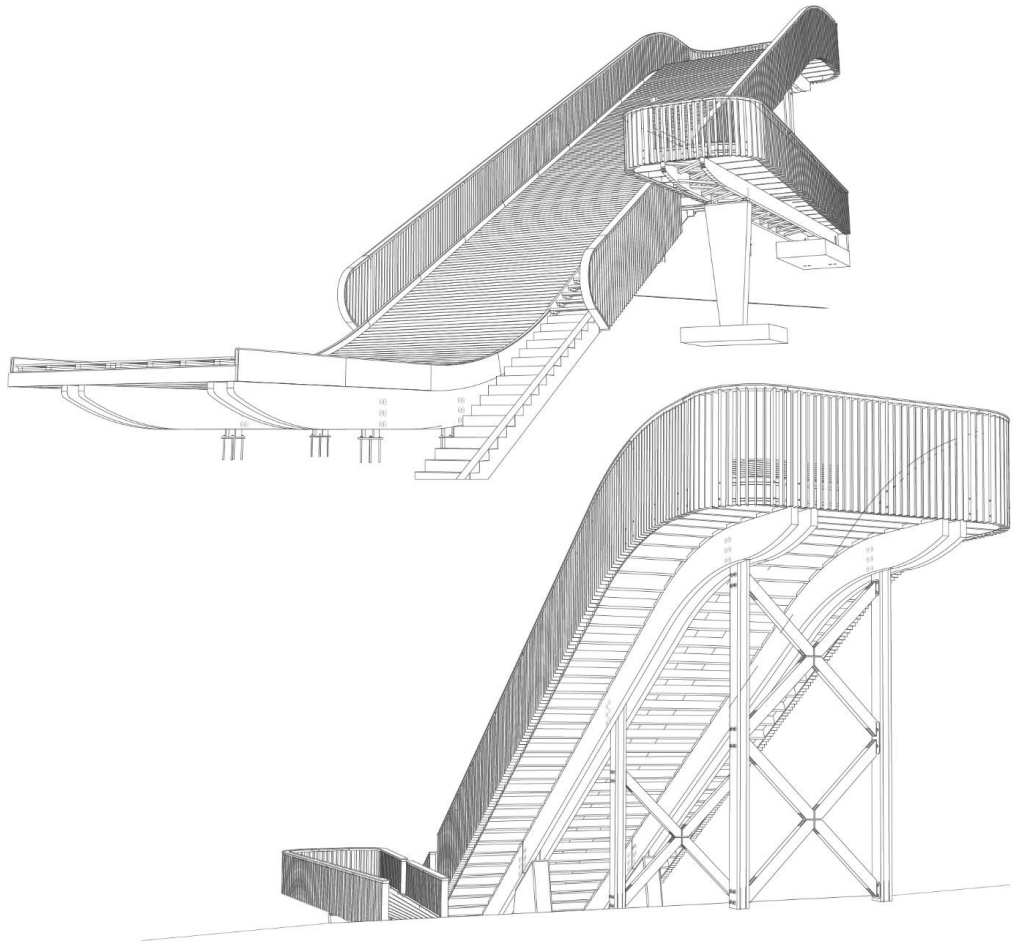
PS: The false rules are valid if the answer is no (false). In this case, when the algorithm checks if the node is close to the ground region, and the answer is no (false), then it is a valid shell node joint type.

- **Consistent element output from the joint:** The element’s location and the relation between two elements instructed the detailing of the connection. This case showed that offering logics to output a consistent order of elements was crucial.
- **Detail plane generators:** The Orkla Bridge offered one type of joint-plane generation. The Printshell exemplified the relevance of generating a node plane based on the orientation of the elements.

-



WATER RAMP



9.3. Water Ramp

The water ramp is a training facility for freestyle athletes. The facility offers a ski jump but the athletes land in a lake after jumping to reduce the risk of injury. In addition to functioning as a ski jump, the structure also serves as a public viewpoint. In this project, a conventional column and beam system were chosen. Two double primary beams support an array of cantilevering secondary beams. The front half of the primary beams is directly connected to the foundations. In the back half, two pairs of columns support the beams.

Similar to most ski jumps, a stair connects the top and bottom. The width of the stair corresponds to the length of the cantilever, which enables the timber stair to merge into a concrete stair and continue into the water.

The time spent designing the water ramp was not academically financed. This fact changes both the design timespan, number of hours available and resources available from the manufacturer. Thus, different findings were harvested from this design process.

Location	Trondheim, Norway
Type	Permanent structure
Completion	2018
Design period	June-September 2017
Design Team	John Haddal Mork, Anders Gunleiksrud (Rallar)
Other collaborators	Moelven Glulam, Rambøll, Bystøl AS
Developer	Trondheim Municipality

Applied

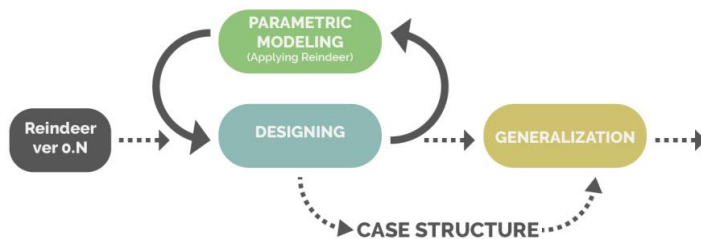


Figure 63: The water ramp was designed by applying the first version of the toolkit. Thus the ramp was an applied activity configuration. The activity configurations are explained in the methodology chapter.

Step 0: Constructing wireframe geometry

1. A manually drawn line represents the direction of the ski jump and the position of the bottom curvature.
2. Three lines represent the kicker, uphill, and top platform. The kicker and top platform are horizontal and defined by two individual lengths. An angle and a total elevation define the uphill line. The requirements of this ski jump were 30 degrees and a 12-metre elevation.
3. The three lines are joined and filleted at the top and bottom.
4. In the transversal direction, the curve is duplicated on both sides to represent the primary beams.
5. The beam curve is vertically offset equal to half of the primary beam height and half of the secondary beam height. Lines that represent the secondary beams are evenly distributed on the offset curve. A dummy line is generated to maintain the relationship between the primary beam and the secondary beam.
6. A grid system defines the distance between the columns/foundations. The positions of the columns are calculated at the primary beam. This position is then vertically projected to the 3D terrain and generates columns.
7. A double and a single cross are generated.

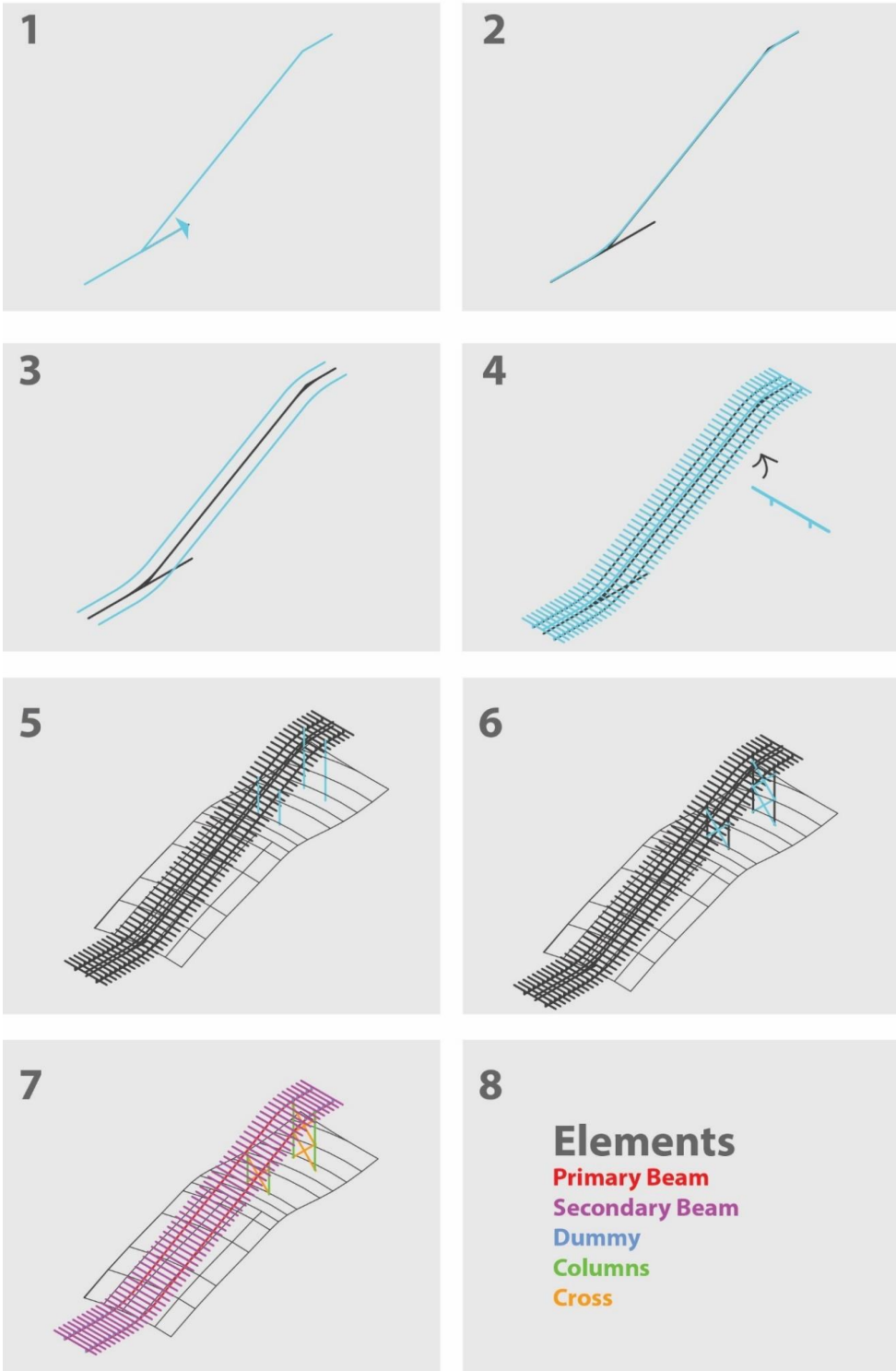


Figure 64: Construction of the wireframe geometry.

Step 1: Defining elements

The manufacturing process of the water ramp did not require sub-elements. However, this structure was also algorithmically generated by applying composites. Inspection of the primary beams and connecting columns reveals that a primary beam is connected to both sides of the column. Instead of creating two, independent primary beam elements, defining a composite of two sub-elements and a centric void equal to the width of the column was more reasonable. However, the previous version of the code did not allow voids in the composite. The lacking functionality was hacked by creating a dummy sub-element, which was equal to the void and was not manufactured.

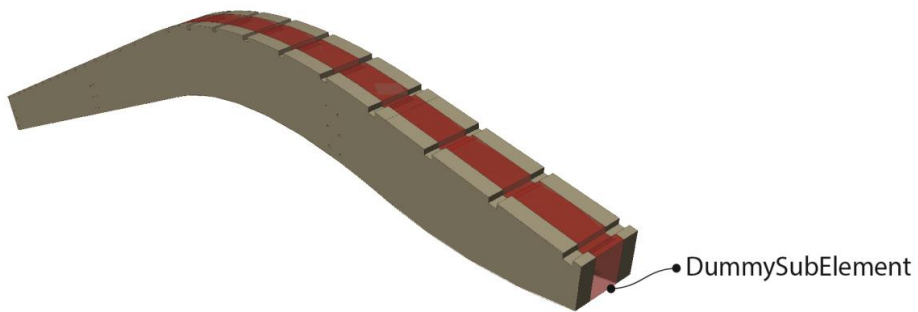


Figure 65: To create a double primary beam, the previous version of the toolkit required application of a DummySubElement.

Step 4: Searching for joints

At first glance, the water ramp structure seems less complex than the bridges. However, the water ramp had a considerably larger number of joint types, which were filtered by searching the element's names.

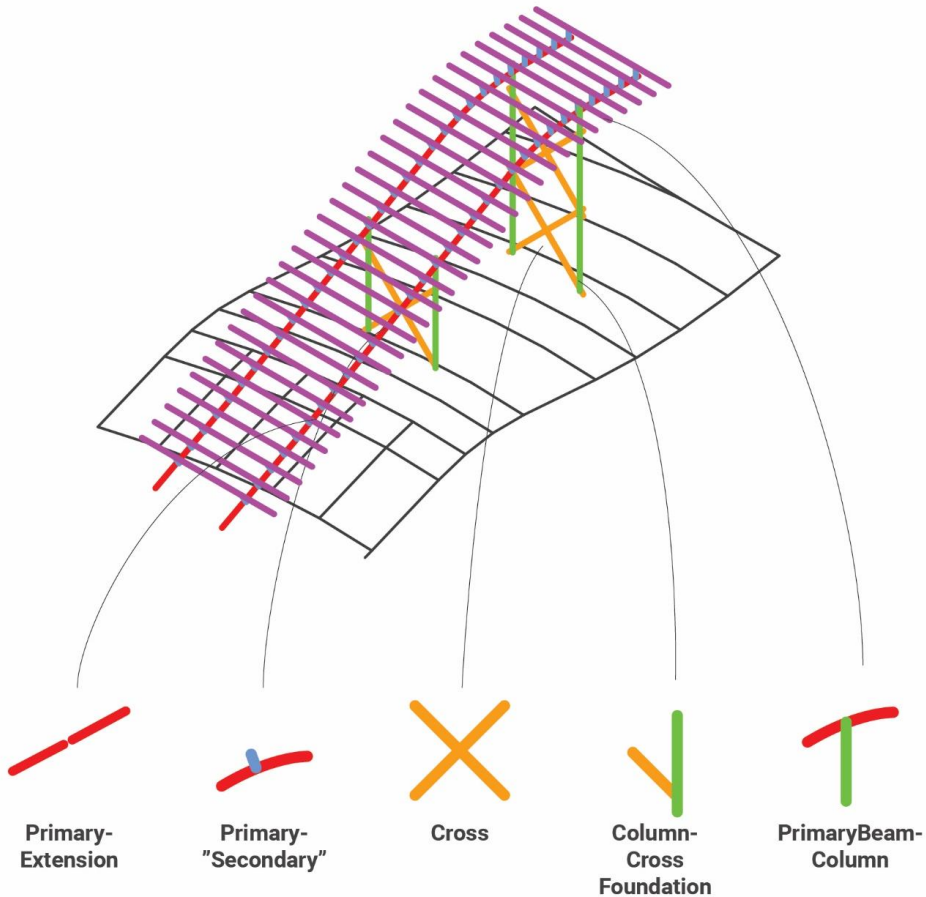


Figure 66: Parametric joint types.

Step 5: Detailing elements and nodes.

A water ramp and a timber truss bridge are not visually similar. However, the detailing principle has similarities. The foundation joint type, cross-bracing, and primary beam extension are connected by slotted metal plates. In this case, the use of bolts instead of dowels was required as the double primary beams need to be connected with bolts that can withstand the transversal load.

The similarities enabled copying the TopChord joint type from the Orkla Bridges and modifying it to a columns-foundation joint type. As previously mentioned, this project was a commercial project, and the flexible reuse saved several hours of defining and modelling of the detail.

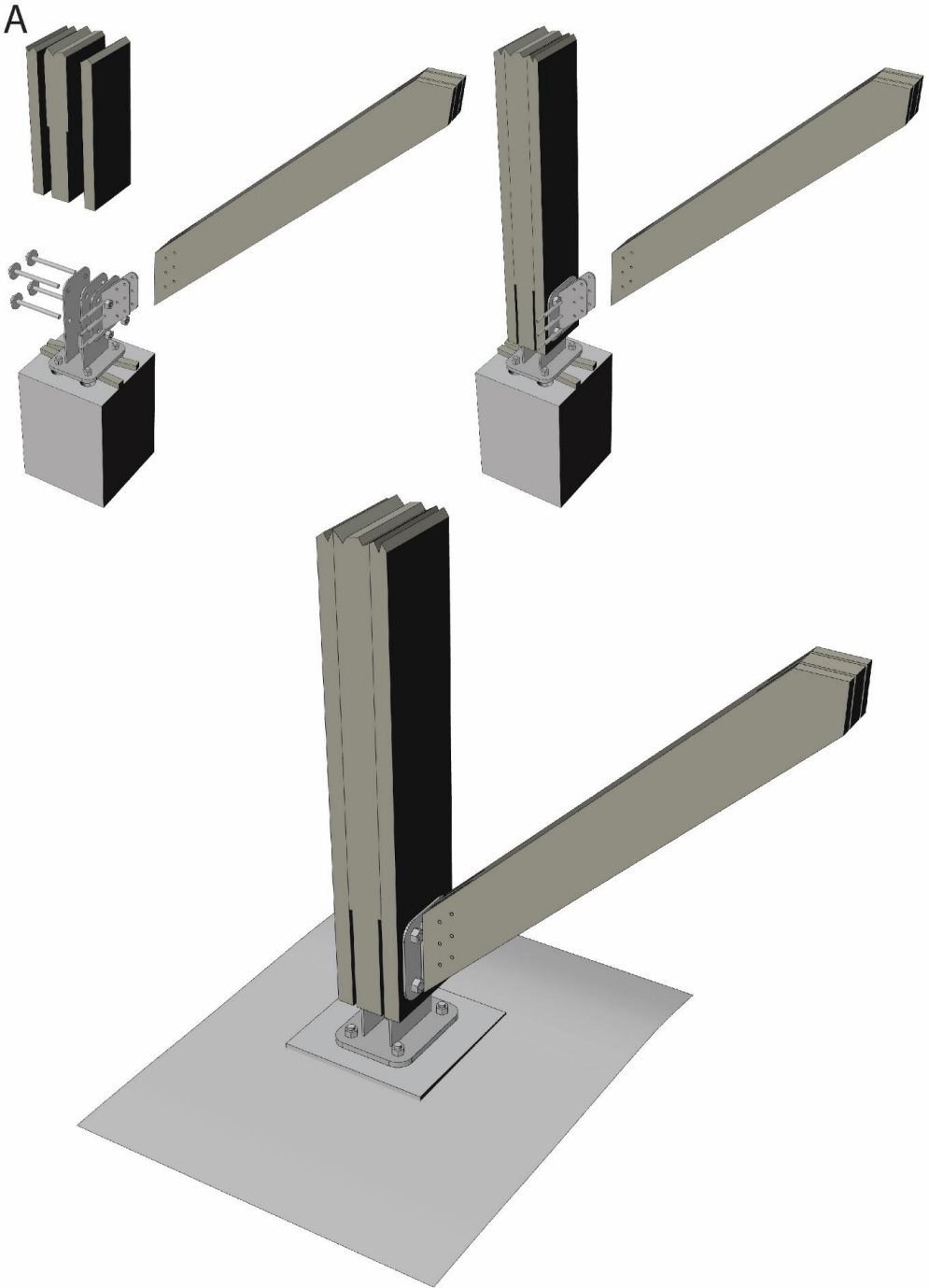


Figure 67: Explosion model of the foundation joint type. The diagonal bar is part of the cross.

The secondary beams were designed to be slotted into the primary beams to simplify the assembly process. In each instance of this joint type, the primary beam was pocketed by a rectangle whose shape was equal to the shape of the secondary beams plus a small tolerance.

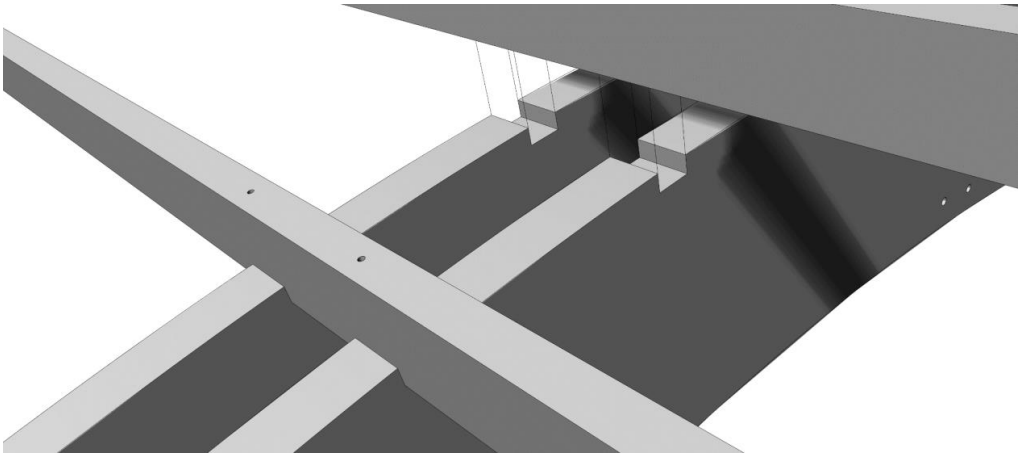


Figure 68: The secondary beams were designed to be slotted into the primary beams.

Step 6: Processing assembly

The entire project was parametrised for digital fabrication and modular pre-assembly, from the primary structure to the deck and railing. However, the manufacturing and building process was externally controlled and caused two changes:

- 1) Although the same manufacturing firm was chosen in this project, a new constructor was chosen for the project. This constructor was not familiar with BTL and required a conventional STEP/IGES 3D model. Thus, the parametric model was baked as NURBS geometry and conventionally exported.
- 2) The glulam structure was digitally fabricated; however, the secondary structure, deck, and railing were manually built on-site.

These two changes will be discussed at the end of this study.

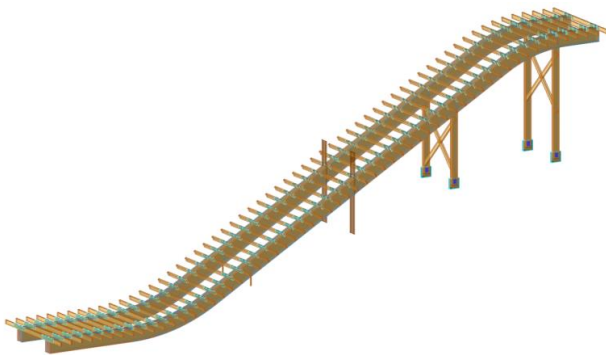


Figure 69: A BTL file of the water ramp. The video shows a conceptual study of where to place the water ramp. Further, the video shows an adjustment of the TopChord joint type.

Step 7: Manufacture and build



Figure 70: The top pictures show the assembly of the load-bearing structure. The platform and railing were fabricated on-site.

Results



Figure 71: All photos on this page: Sophie Labonnote.

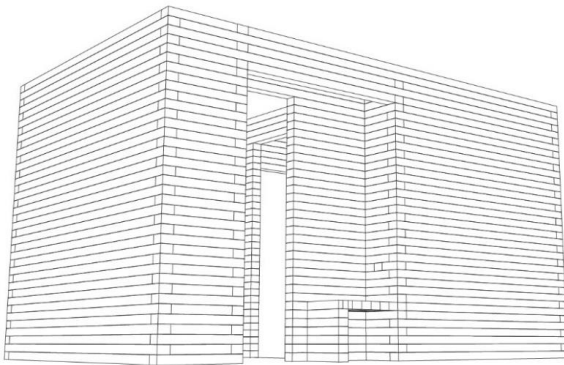
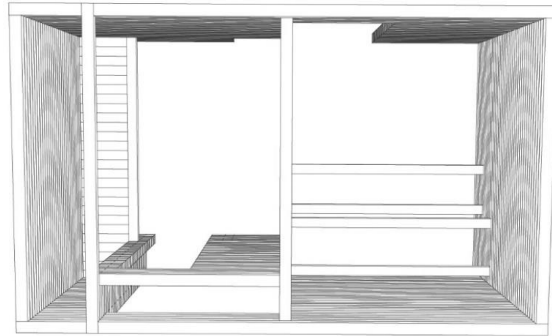
Generalisation of the case structure

- **Void in composites:** The primary beam was described as two sub-elements and a centric void. In the 0.2 Timber Toolkit, a void could only be hacked by using a dummy sub-element in the middle. This lack of functionality enabled a focus on creating a more flexible composite algorithm that allows voids and local alignments.
- **Offsetting cross-section from centre-line geometry:** Timber structures are often built by bypassing elements, which is the case for the secondary beams at the water ramp. Instead of being centrally connected to the primary beams, they are positioned on the top beam. Structurally reasonable: the main force is directly transferred from the secondary beam to the primary beam. The challenge is that the algorithm cannot identify nodes/joints when the centreline geometry does not intersect. A common solution, which was implemented in this project, is to define a dummy element that is connected to both elements. According to the rules of Reindeer, this solution does not join the two beams in one joint. The solution implemented in the current Reindeer is that the composite enables an offset value from the centreline geometry. Currently, this solution works well for detailing but the structural analysis does not consider this offset. Future work may develop an algorithm for creating dummy structural sub-elements with high stiffness to account for eccentric loads.
- **Slotting using the other element:** As illustrated by the primary and secondary beam detail, a lap joint is a conventional detail in timber structures. In this project, the problem was solved by creating a custom pocket that was informed by a rectangle for each instance. However, this process is laborious. The current Reindeer toolkit enables the user to apply a second element to inform how the first element should be pocketed. Additionally, a custom tolerance value can be input.
- **Custom contours:** The ends of the primary beams are filleted to create a more slender expression, which was solved by sending a conventional 3D model to the manufacturer. However, this approach revealed the relevance of inputting a custom cutting contour of an element. This functionality is not supported in the Reindeer algorithm.

-



LOG HOUSE



9.4. Log House

The log house is a small sauna in a secret, rural location in the woods of Sweden. Still, the extensive use of parametric modelling was considered reasonable. Similar to Printshell, this case project was not directly related to the Reindeer development. However, the project had a substantial impact on the development of the DetailSearch.

The sauna is built by elements of 73×98 mm solid wood. These elements have a double tongue and groove. In this project, first, a log house algorithm was developed, and second, a design was developed. This case is relevant with regard to how joints are identified.

For this project, intricate joinery principles were avoided. Instead, every other log is shortened or extended to create an overlapping corner. The corners have a hole for a threaded rod that stretches from the bottom of the structure to the top of the structure.

Location	Deep woods of Sweden
Type	Permanent structure
Completion	2018
Design period	September 2017-June 2018
Design Team	John Haddal Mork, Anders Gunleiksrud (Rallar)
Other collaborators	-
Developer	Private

Detached

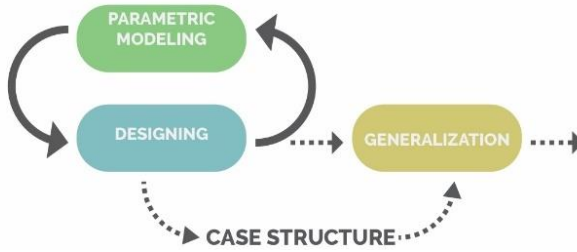


Figure 72: The Log House was designed using a detached activity configuration. The activity configurations are explained in the methodology chapter.

Step 0: Constructing wireframe geometry

1. Draw walls, floors, ceilings, and benches using surfaces
2. Array an XY-plane that corresponds to the layer elevations of the logs
3. Generate centrelines by intersecting the planes and the surfaces
4. Output all lines on the same list. In parallel, a related list with wall names is generated

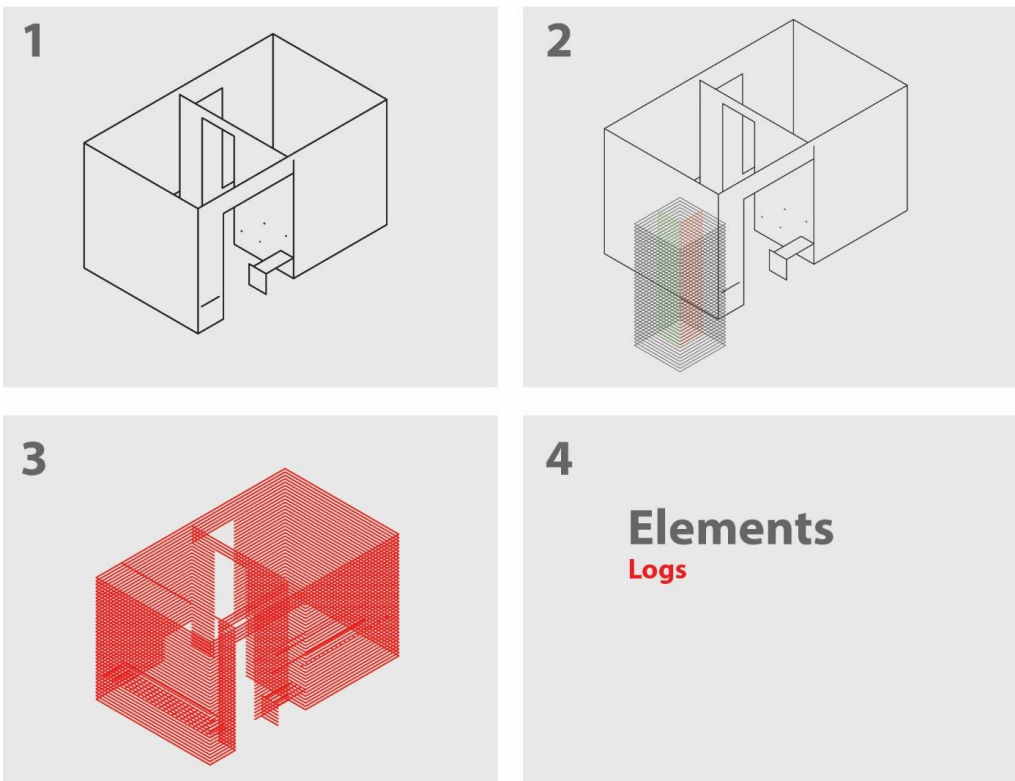


Figure 73: Construction of the wireframe geometry.

Step 4: Searching for joints

This project did not apply any version of the Reindeer toolkit. The same relation between elements and nodes had to be established. When visually analysing a log house structure, four common joint types were identified:

- L-joint: a corner
- T-joint: an end of a wall meets a continuous wall.
- I-joint: a single end of a wall
- X-joint: two continuous walls intersect.

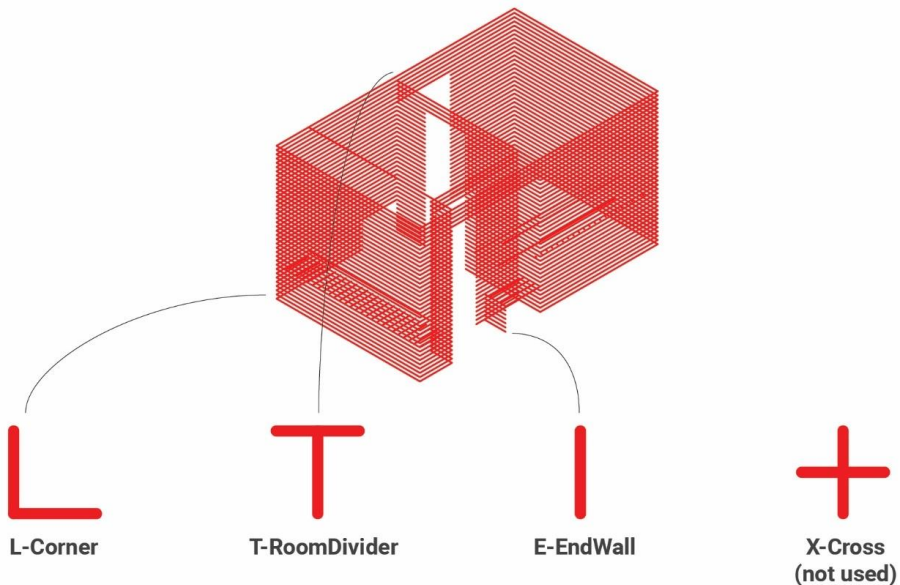


Figure 74: Parametric joint types.

A human easily detects different joints. At first glance, instructing an algorithm to perform the same task is more cumbersome: with the exception of the I-joint, all joints consist of two elements. However, the location of the node relative to the elements distinguishes them. If reparametrising the length of the element from 0 to 1, the logics are defined as follows:

I-joint: Consists of only one element

L-joint: Both elements have the joint's node at parameter 0 or 1

X-joints: None of the elements has the joint's node at parameter 0 or 1

T-joint: One of the element has the joint's node at parameter 0 or 1

The logics that were developed were poorly organised Grasshopper code with a series of steps for filtering each joint's ID. The ID was then used to filter curves, nodes, wall-names, and other information related to the same joint.

Step 5: Detailing elements and nodes.

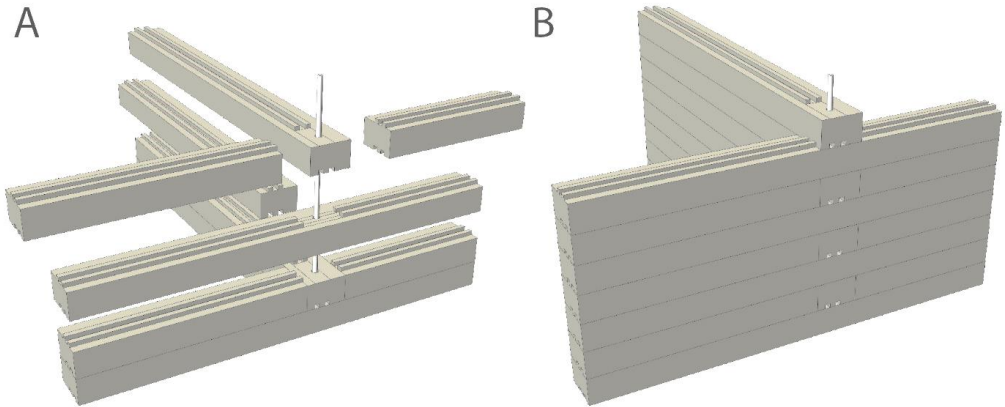


Figure 75: A simple detail. Every other element is trimmed. Additionally, a suspended threaded rod keeps the structure stable.

Step 6: Processing assembly/Step 7: Manufacture and build

This project was manually prefabricated off-site. Thus, the goal of the assembly processing was to output manual fabrication instructions and ensure element logistics. The developed solution was a sticker that contains fabrication instructions: length and optional positions of drilling holes.



Figure 76: The output from the Grasshopper model was A4-sheets of stickers, which contained ID, position, length, and eventual holes.

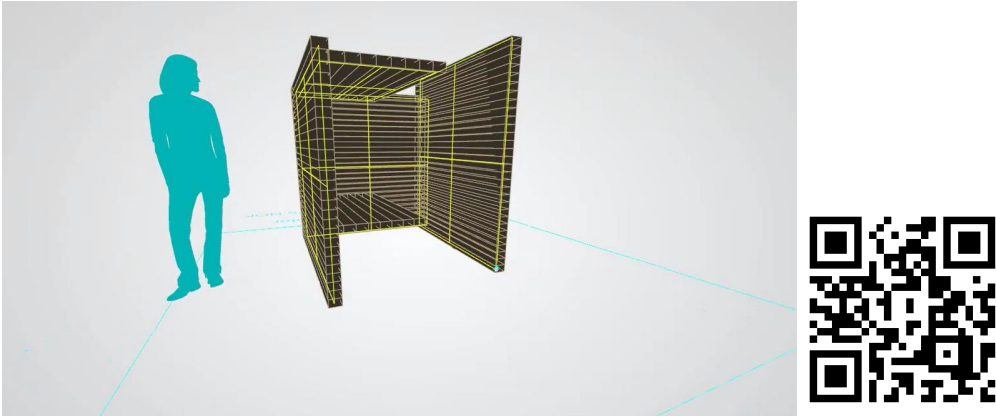


Figure 77: The QR code shows how the user can sketch using surfaces. The background algorithm analyses the surfaces and generates logs according to the embedded detailing principle.

Results



Generalisation of the case structure

- **Detail Topology Search:** The identification of different joint types in the log house proved the value of being able to distinguish the basic shape of a joint. Thus, Reindeer has implemented the joint topology component. In addition to the previously mentioned topologies, the inclusion of star-shaped (3 or more elements), planar and orthogonal joints was reasonable.
- **Flexible output:** The toolkit is tailored for digital fabrication by applying BTLx. However, projects are not always completed as expected. The water ramp, Printshell, and log house exemplify this notion. The examples show the importance of ensuring that the toolkit provides an output flexibility from the TimberProcessingTools.

9.5. Smaller tests

None of the presented, built structures were designed by applying the latest version of the toolkit. In this section, I briefly present a few imaginary case structures that illustrate the functionality that is embedded in the toolkit.

From Log house to wall and shelves

The log house algorithm contained four joint types determined by the topology of joints. Fascinatingly, a shelf and a frame wall are parametrically similar to a log house layer.

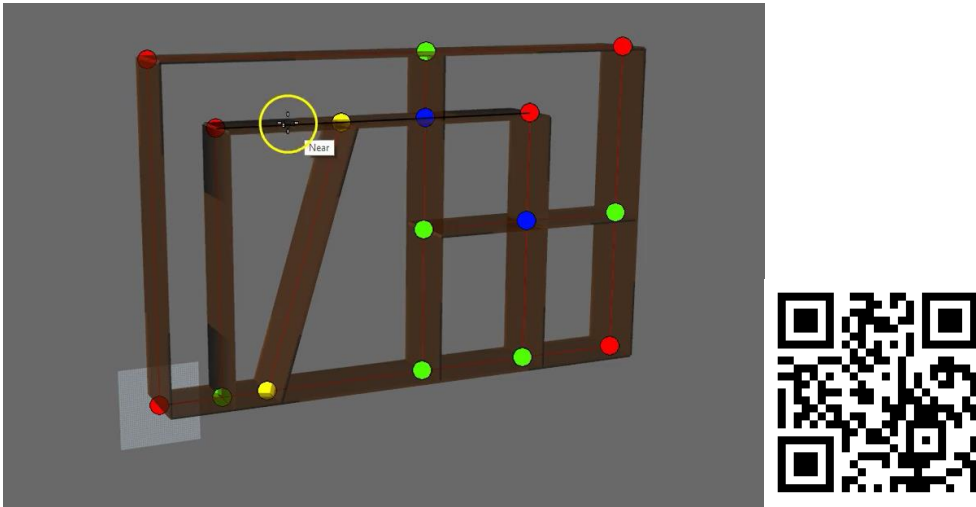


Figure 78: Shelf generator. The sketch process is observed by opening the video in the QR code.

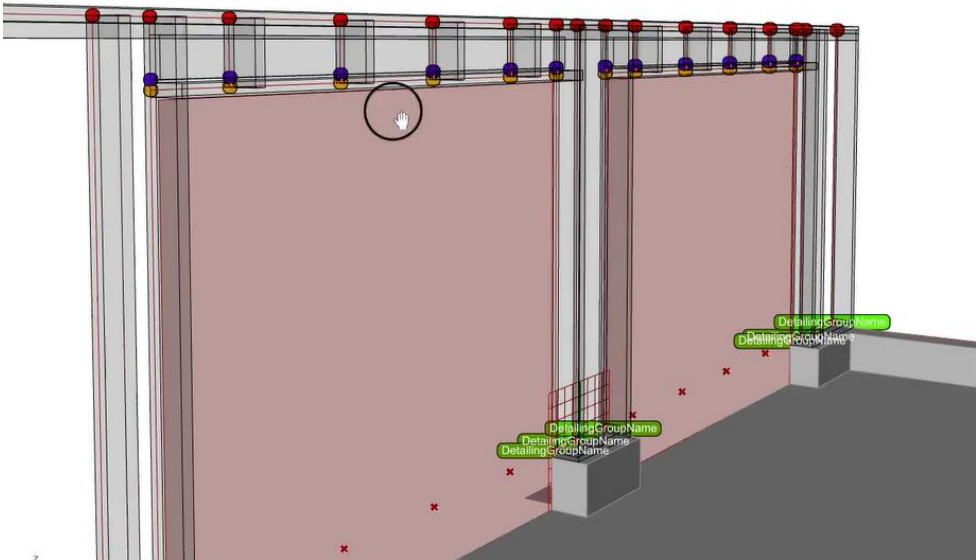


Figure 79: A wall. Details identified by Reindeer.

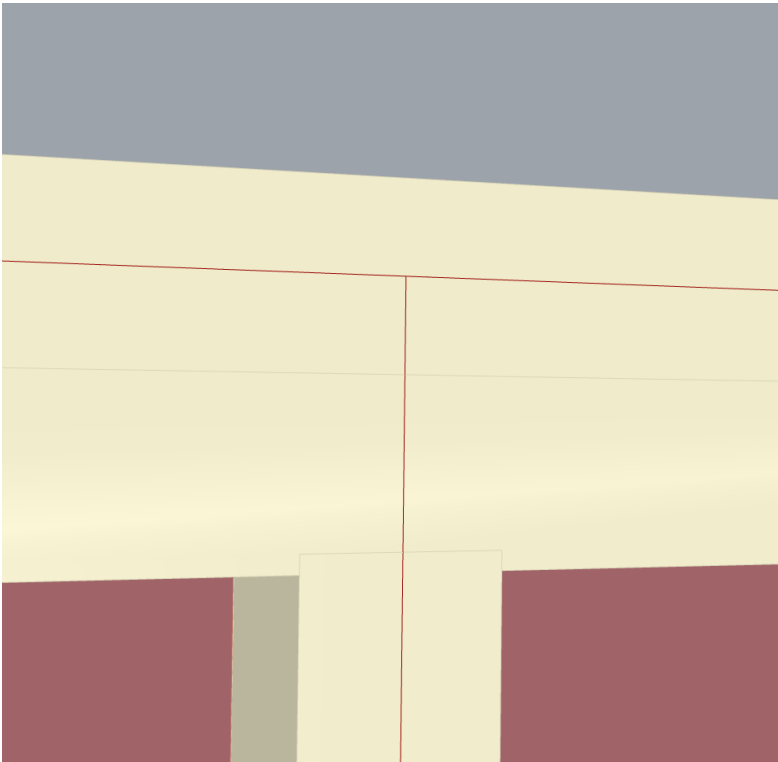


Figure 80: The red joint in the figure. Similar to conventional pre-cut systems, the column is slotted into the beam.

Grindbygg/Bent

A bent (Grindbygg in Norwegian) is a traditional rigid frame structure that is connected by carpentry joints. A pilot of a parametric grindbygg was designed to test the functionality of TimberProcessingTools.

The digital fabrication of a bent is not unique. The collaborating manufacturer has an existing workflow that applies SEMA and a Hundegger Speedcutter. However, the workflow was reported to be slow and dominated by redundant tasks. The Grindbygg example illustrates the flexibility of the Reindeer Toolkit.

As illustrated by the video in the QR-code (refer to Figure 81), the parametric model was fully flexible for parameters such as length, width, depth and roof angle. The number of structural elements were updated accordingly. DetailSearch applies various search criteria to identify the parametric joints.

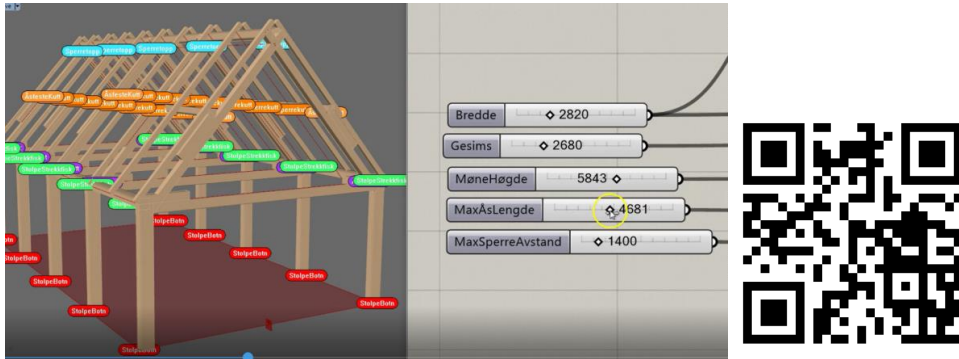


Figure 81: The video shows how the Grindbygg is being modified. The last part of the video shows the detailed carpentry joints.

Some of the joints were detailed using tenon/mortise, pocketing, drilling and cutting. The following pictures show the parametric structure transferred to SEMA.

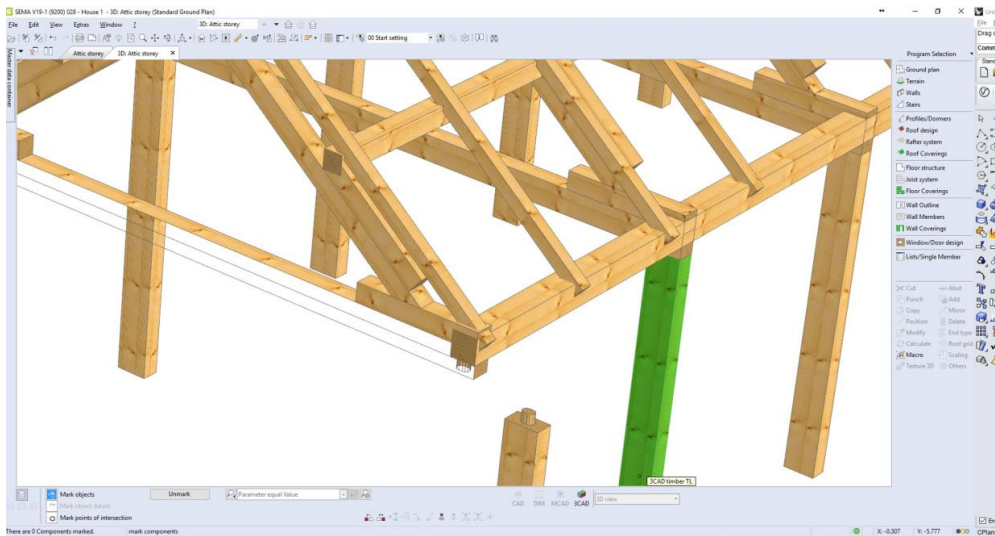


Figure 82: Exported to SEMA.

10. Papers

Figure 83 explains the relation among the papers included in the thesis. Paper I is broad and explains the main concepts of the toolkit. Paper II focuses on JointSearch and related preparatory steps. Article III explains the TimberProcessingTools. The latter article summary is extended with a section that explains the related algorithm strategy.

Paper I

Toward mass-customized architecture. Applying principles of mass customization while designing site-specific, customer-inclusive and bespoke timber structures

Paper II

JointSearch: Efficient parametric detailing preparation through user-defined and property based joint type filtering

Paper III

Parametric timber toolkit: A timber tailored approach

The order of the papers is based on readability and are not based on the date of publishing.



Figure 83: Relation among papers.

10.1. Paper I: Framework and basic explanation of concepts

Paper title:

Toward mass-customised architecture. Applying principles of mass customisation while designing site-specific, customer-inclusive and bespoke timber structures

Publisher:

Digital Wood Design

Innovative Techniques of Representation in Architectural Design

2019

Book chapter, Springer

ISBN 978-3-030-03676-8

Co-Authors:

Marcin Luczkowski, Bendik Manum, Anders Rønnquist

Co-Authors level of contribution:

Most of the paper was written/illustrated by Mork. Luczkowski wrote the 3.2 “Structural engineer’s conception of an element” and 4.2 “Structural engineer”. Luczkowski contributed to the content in the paper. Manum and Rønnquist assisted in structuring the paper and finding errors.

Summary

Paper I is broad and presents the background of the toolkit, including the intended workflow, conceptions of an element, definition of a detail (now joint), how detailing groups (now joint types) are identified, and how elements are detailed using subtractive tools.

In this paper, special attention is given to the conception and requirements of geometry by architects, structural engineers and manufacturers. The paper explains the different conceptions of both an element and a detailed geometry and proposes a unified solution.

An element is proposed to contain both manufacturing sub-elements and structural sub-elements. In this way, all data of an element can be stored in the main element.

Further, the paper defines a detail and discusses how the toolkit divides details into different detailing groups by filtering each detail based on custom search criteria inputted by the user.

Detailing geometry varies among the involved stakeholders. The input of detailing a timber element is equal for the profession, e.g., a drill hole. However, the architects require a preview geometry, the manufacturer requires a blank and a processing, and the structural engineer requires a structural model. The paper proposes one tool for each operation that outputs three different geometries.

Four case structures are presented. In addition to the Orkla Bridge project, the water ramp, shelf, and log house are presented. All structures are similarly presented by illustrating the different detailing groups and explaining their respective search criteria.

Three reflections were made. First, many architects are on a quest to create uniqueness in their project, which contradict mass customisation, where some aspects require standardisation and

(variable) repetitions. The paper claims that parametric modelling is a great tool that enables a standardisation with a high degree of flexibility. Instead of standardising the physical result, a flexible workflow can be standardised, which is possible as time is spent creating the system—instead of the generated instance—when parametrically modelling. If knowing how to create flexible parametric models, significant amounts of code and workflow can be reused for future projects.

The second reflection concerns similarity. Parametric modelling changes what is considered to be a similar structure. What appears as visually and radically different structures from an architectural point of view is very similar from an algorithmic point of view. The log house and shelf exemplify this issue. Conversely, two functionally similar structures are fundamentally different from an algorithmic perspective. A truss bridge is more similar to a free-form gridshell than a beam bridge. This reflection relates to the parametric system. A parametric system built by elements and nodes is easily adapted to becoming a gridshell and a timber-truss bridge. However, the parametric system of a beam bridge is less similar.

The third reflection relates to the architectural quality. Does the toolkit ensure great, bespoke timber structures? The toolkit does not ensure great architecture. At best, the toolkit reduces the amount of work spent on redundant tasks. However, the hours earned can be spent achieving better architecture. Architecture is made by architects; tools are tools that simplify the creative process.

Afterword

Vocabulary change:

- Detail->Joint
- Detailing Group->Joint type

The paper presents an 11-step workflow. The content has large similarities to the presented 7-step workflow but the paper's workflow is more exploded. However, the logic of detailing groups has changed. This paper proposed inputting the search criteria to the assembly component. To detail a detailing group, a person inputted the name of the detail in the detailing group component. This logic was laborious, lacked intuition and required more computing power. Thus, the detailing group evolved into a JointSearch that inputs the assembly.

10.2. Paper II: JointSearch

Paper title:

JointSearch: Efficient parametric detailing preparation through user-defined and property based joint type filtering

Publisher:

Submitted for journal publication

Co-Authors:

Marcin Luczkowski

Co-Authors level of contribution:

The paper was written/illustrated by Mork. Luczkowski contributed to the content in the paper.

Summary:

This article focuses on Reindeer's JointSearch and has a technical focus. A generalised joint filtering methodology is presented. The article claims that the number of different joint types is infinite. Thus, a joint filtering methodology that enables the user to modularly define a joint type's solution space that is used for filtering joint instances is needed. The presented principle pertains to 1D timber elements; however, the methodology is applicable for other industries, element types and materials.

Both preparatory steps—JointSearch and JointOutput—are thoroughly explained using concepts, flow-charts and code.

Defining elements: The article explains how elements are defined using modular components.

Identifying Joints: The article explains how a node/joint is identified at the endpoints and intersections of elements. While the end-point algorithm is straightforward, the latter event requires a more thoughtful code to achieve satisfactory performance. The presented solution employs R-tree to limit plausible intersections to check. Further, a simplified class diagram of the assembly, joint, element and sub-elements is presented.

JointSearch: The core of the article explains how JointSearch works. To ensure scalability, the search criteria's format is strictly defined: Only questions about the joint can be asked; the answer to the question must be true or false; and the question allows user-defined variables. This question is implemented as a delegate method that inputs a joint and returns a Boolean value. These search criteria can either be inputted to be true to be valid or inputted to be false to be valid. When a joint is tested, all criteria must be valid.

JointOutput: If a joint is valid, it is being outputted for detailing. To ensure a most possible coherent data stream, the user can specify the logic that orders the elements and the logic that generates the node-plane. Further, the article explains how a joint is deconstructed into elements and nodes and further deconstructed into parameters and sub-elements.

The concepts are exemplified via small tests, which show the flexibility of the search criteria and the coherency of the joint output.

10.3. Paper III: TimberProcessingTools

Paper title:

Parametric timber toolkit: A timber tailored approach

Publisher:

World Conference on Timber Engineering (WCTE)

WCTE2018, Conference, Oral presentation

Seoul

20 - 23 Jun. 2018

Conference, Oral presentation

Co-Authors:

Marcin Luczkowski, Bendik Manum, Anders Rønnquist

Co-Authors level of contribution:

The paper was written/illustrated by me. Luczkowski contributed to the content in the paper. Manum and Rønnquist assisted in the structuring of the paper and identifying errors.

Summary

Paper III links directly to TimberProcessingTools and problematizes the gap between digital design and physical processing of timber structures.

Tacit knowledge becomes embedded in a carpenter, and based on available tools, he refines a blank into the desired shape. As described in the background chapter, the paper discusses how wood processings are primarily subtractive.

Conversely, when modelling 3D structures in a digital environment, a person starts with a blank canvas. Points, lines, curves, surfaces, and solids are digital geometry that are used to additively build a geometry.

The paper raises two challenges about this method. The first challenge pertains to forgetting or not knowing the machining limitations when digitally modelling a design. If this issue is not addressed or resolved before manufacturing, errors may occur. The second challenge relates to transferring a digital timber structure to fabrication.

The paper presents how the toolkit's subtractive tools are functioning: The user applies tools that mimic wood processings, such as drilling, cutting, and pocketing. The outputs are NURBS geometry and a 3D file that is readable for manufacturers (BTLx). The expected result is that the user is exposed to similar limitations and possibilities that will be encountered when fabricating the structure.

Further, the paper explains how the bottom chord at the Orkla Bridges was designed using subtractive tools and illustrates both a digital model and the physical result.

The paper emphasises that the parametric toolkit shall not replace the value of manually crafting objects. These activities will always offer a more in-depth understanding that is valuable during digital design.

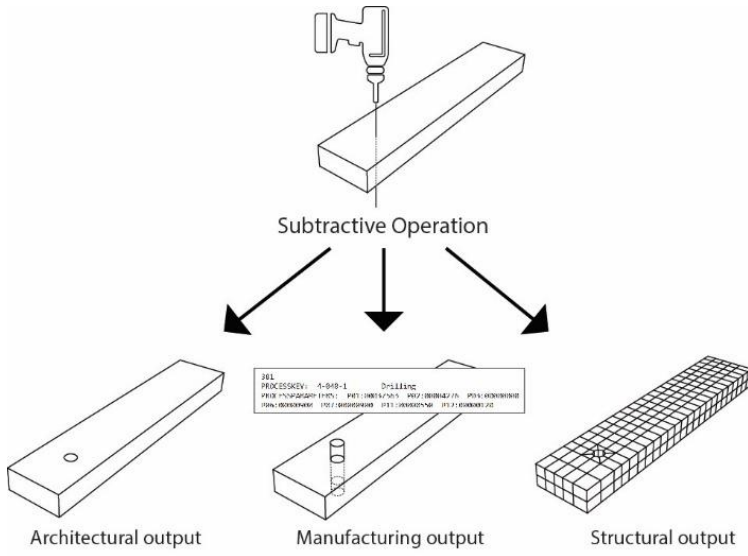


Figure 84 Subtractive detailing tools.

III. DISCUSSION, CONCLUSION AND FURTHER WORK

11. Discussion

11.1. Algorithm strategy used in Paper III - Timber processing tools

Explaining the algorithm strategy was beyond the domain of the paper. Thus, the following section provides a better understanding of the TimberProcessingTools algorithm.

Algorithm strategy

The algorithm strategy for timber processing has similarities to the search criteria strategy. TimberProcessingTools outputs an instruction that contains the element and a delegate method. This method stores inputs from the processing and how the processing is performed on the element. The processing assembly inputs the timber processes, links them with its elements and performs the processes. Figure 85 shows how TimberProcessingTools and ProcessModel components are connected. The following section provides a more detailed, systematic explanation:

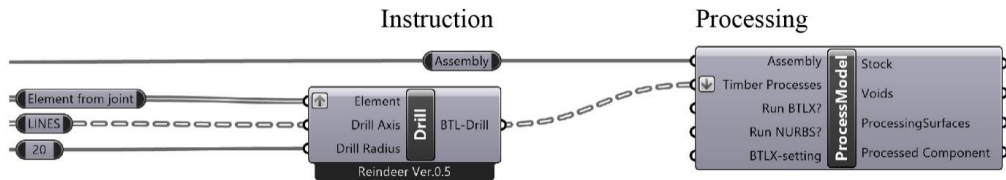


Figure 85: Drill instruction and processing.

1. Generate timber processing instruction

Each timber processing has various inputs. These inputs are stored in an initiated object. Each of these objects has a delegate method. With the element, the delegate method is wrapped as an OrderedTimberProcess and outputted from the component.

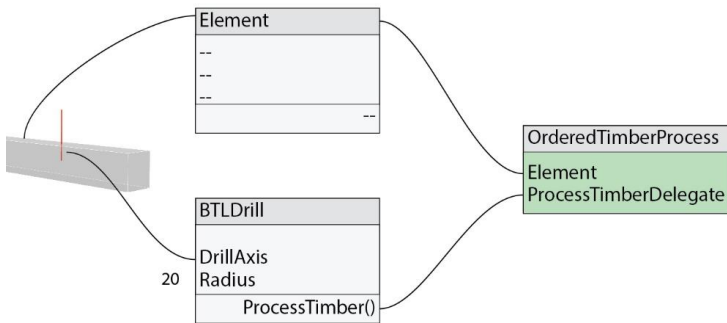


Figure 86: Wrapping an element and a ProcessTimberDelegate.

2. Link timber processings and elements

When the timber processings are inputted in the process model component, all timber processings are linked with the correct elements. This step is performed by using the address of the element object. Figure 87 illustrates the linking. Figure 88 renders the actual linking algorithm.

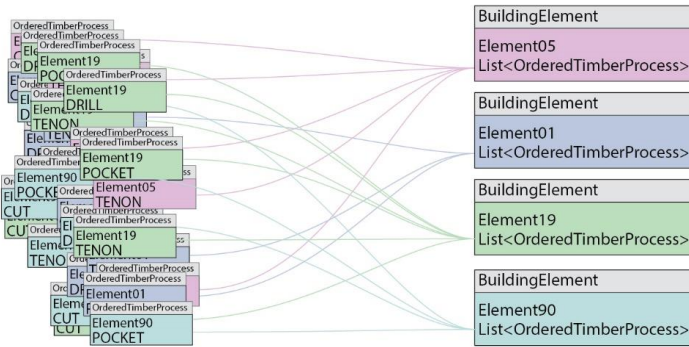


Figure 87: Linking the elements and timber processes. One element may have multiple timber processes.

```
foreach (ElementID element in _elements)
{
    List<OrderedTimberProcess> OrderedProcessesInElement = _orderedTimberProcesses.Where(o => o.element == element).ToList(); (1)
    List<PerformTimberProcessDelegate> processDelegateInElement = new List<PerformTimberProcessDelegate>();

    foreach (OrderedTimberProcess Process in OrderedProcessesInElement)
    {
        processDelegateInElement.Add(Process.PerformTimberProcess); (2)
    }
    //Find all correct _orderedded Timberprocesses. Store in list
    BuildingElements.Add(new BuildingElement(element, processDelegateInElement)); (3)
}
```

Figure 88: Linking elements and processes. Step 1 identifies all `_orderedTimberProcesses`, where the wrapped element is equal to the element for which the search is performed. Step 2 extracts the `Process`. Step 3 initiates a building element that contains both the initial element and its processes.

3. Run through each element and each element's manufacturing sub-element.

All timber processings are performed on the manufacturing sub-elements instead of the architectural elements. Thus, the algorithm needs to loop through each element and then loop through each manufacturing sub-element in the element.

This stage generates a BTLx-part that is equal to the manufacturing sub-element.

```
<Part Count="1" Length="2131.44" Width="100" Height="100">
  <Transformations>
    <Transformation GUID="{884c643c-726b-4772-b8dc-ce0dd9884867}">
      <Position>
        <ReferencePoint xsi:type="PointType" X="-1000.286" Y="-53.552" Z="-50" />
        <XVector X="0.31" Y="-0.94" Z="0" />
        <YVector X="0" Y="0" Z="1" />
      </Position>
    </Transformation>
  </Transformations>
  <Processings>
    Timber processings will be added here
  </Processings>
</Part>
```

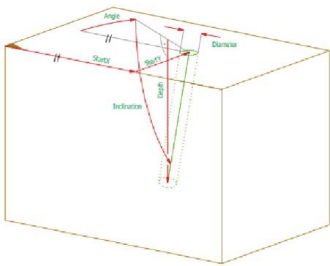
Figure 89: A BTLx part; defined by location, orientation, length, width, and height.

4. Run all sub-element's delegate method

At this stage, all delegate methods in a sub-element are run. The delegate method inputs geometric data related to the sub-element and returns a BTLx processing and BREP that represents the void. How the BTLx processing is defined and how BREP is generated varies from processing to processing. The following section applies a drill processing as an example.

To create a BTLx process, various parameters need to be harvested. Figure 90 shows the information that is needed for drilling processing. The delegate method calculates these numbers and stores them in a BTLx-Drill processing object (refer to Figure 91). The calculated numbers are based on a reference side relative to the BTLx part. Therefore, the most reasonable reference side is calculated.

BTLx's geometric description



BTLx's numeric description

Name	Type	Default	Min	Max
StartX	LengthPosType	0.0	-100000.0	100000.0
StartY	WidthNType	0.0	-50000.0	50000.0
Angle	Angle3Type	0	0.0	360.0
Inclination	AngleType	90.0	0.1	179.9
DepthLimited	BooleanType	no	no	yes
Depth	WidthType	50.0	0.0	50000.0
Diameter	DiameterType	20.0	0.0	50000.0

Drill-instruction in the BTLx-file

```
<Drilling Name="Drill" Process="yes" Priority="0" ReferencePlaneID="3">
  <StartX>679.93</StartX>
  <StartY>450</StartY>
  <Angle>0</Angle>
  <Inclination>90</Inclination>
  <DepthLimited>yes</DepthLimited>
  <Depth>50</Depth>
  <Diameter>40</Diameter>
</Drilling>
```

Figure 90: Geometric and numeric description obtained from www.design2machine.com. The grey box is a parametric description of a drilling operation.

```
Point3d localPlaneEndPoint = new Point3d();
Refside.RefPlane.RemapToPlaneSpace(AlignedDrillLine.To, out localPlaneEndPoint);
Point3d localPlaneProjectedEndPoint = new Point3d(localPlaneEndPoint);
localPlaneProjectedEndPoint.Z = 0;

double ToMM = CommonFunctions.ConvertToMM();

DrillingType Drill = new DrillingType();
Drill.StartX = localPlaneInsertPoint.X * ToMM;
Drill.StartY = localPlaneInsertPoint.Y * ToMM;

if (Math.Abs(localPlaneEndPoint.Z) < Refside.RefSideZLength)
{
  Drill.Depth = -localPlaneEndPoint.Z * ToMM;
  Drill.DepthLimited = BooleanType.yes;
}
else
{
  Drill.DepthLimited = BooleanType.no;
}

Drill.Diameter = Radius * 2 * ToMM;
```

Figure 91: DrillingType object is being initiated. The code illustrates how a drilling instruction is converted to BTLx parameters.

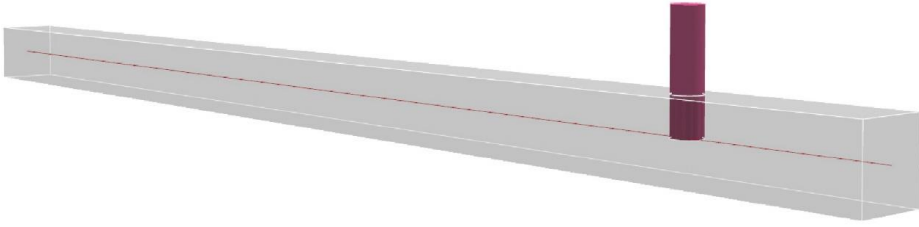


Figure 92 The pink cylinder applied in the subtractive Boolean operation.

The generation of BREP geometry is more straightforward than a BTLx description. A cylinder is constructed based on the information obtained from the delegate method's object; refer to Figure 93. When the BREP and BTLx processing is returned, it is added to respective lists associated with the manufacturing sub-element.

```
//Making NURBS GEOMETRY  
Circle Circle = new Circle(new Plane(DrillLine.From, DrillLine.Direction), Radius);  
Cylinder Cylinder = new Cylinder(Circle, DrillLine.Length);
```

Figure 93: Generation of the subtraction Cylinder.

When all delegate methods have returned BREPs that represent the timber processings, a Boolean difference operation is performed. The last step is to output the desired geometry/file format. The file content is generated by serialising the BTLx class structure.

11.2. A specialist's tool or a mainstream tool?

Is it too demanding to be both an excellent architect and a skilled parametric modeller? The required knowledge is ever-expanding and includes knowledge of digital design tools and topics from other disciplines.

Visual programming has drastically simplified parametric modelling, and future innovations will probably further simplify parametric modelling. Reindeer is a small step towards further democratising parametric modelling: The tool hides a few of the technical challenges in black boxes, and the BTLx translation and DetailSearch simplify the parametric modelling experience.

I continue to defer advanced parametric modelling to specialists. Parametric modelling will become more common, and if most architects know its basic principles, communication and teamwork will be improved.

I have contemplated whether I will use Reindeer in future projects. Reindeer is applicable to geometric tasks that are more complex than the presented case structures. However, if the complexity and data amount are significantly increased, the Grasshopper plugin will become too slow. In these cases, hacking the core algorithm of Reindeer and creating bespoke, project-specific components would be more reasonable.

Reindeer has implemented advanced techniques in a tool that will be applicable for most technically interested architects. The toolkit reduces the gap between intermediate skilled parametric modellers and the stunning works performed by DesignToProduction, CITA, and other designers. If users grow out the capabilities of Reindeer, that would be a good thing.

11.3. Redefining similarity

What is similarity? Two humans can appear very different, for example, different sex, different skin colour, different nose size, and different height. However, if you focus on their basic characteristics, they are almost identical, for example, two legs, one heart, one head, and two eyes. By grasping knowledge in this study, I am convinced that parametric modelling is changing our conception of similarity in building projects. Similarity in architectural design is not limited to function, materials and shape. Increasingly important, similarity pertains to the topology and parametric system that generates the geometry.

As reaffirmed by Aish in the introduction[14], we are not directly drawing the geometry. Instead, we are modelling a system that generates geometry. Thus, similarity pertains to the system more than the shape. A bridge can be more similar to a gridshell than to another bridge with an equal span, material, and shape. Similarly, a parametric log house system can be more similar to a shelf than a Cross Laminated Timber house system.

To fully utilise the power of parametric modelling, we must be aware of this shift. If using the old rules of similarity, we do not harvest the potential of reusing parametric systems from project to project.

Architects are often afraid of constructing similar buildings. A building is unique and adapted to its function, local building culture, and surroundings. If abstracting similarity to the level of a parametric system, we can reuse parametric systems while developing bespoke buildings. The log house to shelf and truss bridge to water ramp transition exemplifies this notion.

If we are to reuse parametric systems, the models need some level of flexibility in their design space. As discussed, the flexibility of a code is often dependent on the time invested to develop the code. From project to project, a budget for developing a flexible and highly reusable system is not likely. However, as it is likely that an algorithm can be used in multiple projects, the investment can be spread long-term. In addition to rethinking similarity, we need to shift our mindset from thinking that we are creating a structure to thinking that we are creating a structural system—a parametric, flexible structural system that is a basis for multiple projects.

Since DetailSearch is purely based on a joint's property, a parametric joint type can easily be reused and adapted from project to project regardless of whether the topology and function significantly differ. A design office can then build an ever-growing library of parametric joint types that can be reused, combined and modified in future projects. This step is a small step to ensure mass customisation of the architectural industry.

11.4. Consistent detailing

When manually detailing, purposed for digital fabrication, it does not matter if two different joints are logically similar: you have to draw both anyways.

Conversely, creating one joint type that handles a large design space is potentially timesaving. The fewer joint types that need to be developed, the less work required. Further, when developing a second joint type, the first joint type can be copied and adapted to suit another purpose. The result, at best, is a more consistent detailing and more coherent architectural structures.

11.5. Control

According to Carpo, digital fabrication is similar to a prosthetic extension[4]. I find controlling a machine easier than controlling a carpenter. I do not blame the carpenter. Phone calls, notational drawings, and mail are sufficient for some purposes but are highly imprecise. Rather, if I know how to instruct a machine and design for assembly (self-positioning building components), I am more confident of transforming a design into reality.

A relatable example is the water ramp. The entire project was parametrically designed, prepared for digital fabrication, and planned for off-site pre-assembly. The contractor digitally fabricated the main structure but wanted to build the deck and railing on-site. Building the deck and railing on-site was time-consuming and caused delays and changes. While the primary structure turned out exactly as planned, the railing and platform were significantly modified. The built detailing principles were improvised and less durable than the planned principles.

With the prosthetic extension that digital fabrication provides and the belief that architects will be reinstated as virtual craftsmen, I envision a beautiful future. Although some examples have been developed, I look forward to creating new solutions based on old and clever, pre-industrial

detailing principles and expressions; intricate carpentry joints; crafted timber surfaces; and the application of custom roots for special shapes. The possibilities are endless.

Large volumes are better produced by machines but do not make the carpenter excess. Ideally, this reduced workload enables the carpenter to spend his or her skills on one detail, one handrail, one door hinge or a special window frame with a custom bookshelf. In this way, we can reinstate the carpenter as a craftsman and better utilise both humans and machines.

An architect's level of control in a building project is highly dependent on the available time and budget. Often a budget is fixed. Thus, finding ways to make the design process more efficient while maintaining or increasing the architectural quality is important. I have been discussing how Reindeer automates technical challenges, how we must redefine similarity, and how we must start thinking reusable structural systems. If adequately performed, these adjustments can contribute to making a design process more efficient.

11.6. Productivity, Emissions, and Waste

How does Reindeer address the introduced global challenges? Believing that a PhD study comprises a large contribution would be foolish. Thus, this discussion primarily occurs from a general perspective and reflects on how parametric modelling and digital fabrication influences productivity, emissions, and waste.

Redundant work decreases productivity. Errors and bad interfaces among professions also account for decreased productivity. If correctly performed, parametric modelling automates redundant work and simultaneously reduces the chances of hidden errors. Parametric modelling and scripting enables designers to customise interfaces among professions.

Given that a design requires custom building elements, digital fabrication is likely to increase productivity. Timber components are sequentially manufactured, and geometric variations do not significantly influence the manufacturing speed. However, one must be aware that rationalising and decreasing complexity by decreasing the number unique components has merit. Manually modelling one component—which can be scattered within a structure—is likely to be more efficient than creating a sophisticated algorithm that generates unique components.

Design for assembly[76] and design for deconstruction[77] are two relatable terms. Design for assembly pertains to creating structures for which assembly is easy and efficient. Self-positioning joints that are fixed by screws/bolts in freeform shells are modern examples, and log houses and Japanese joinery techniques are traditional examples.

Design for deconstruction encompasses being able to reuse building materials by deconstructing them into their original components. Avoiding glued composites is key.

Designing for assembly is most likely to be designing for deconstruction, as long as components are not glued or assembled using other non-reversible methods. Thus, designing for assembly as a strategy may increase productivity while decreasing construction and demolition waste. I see a bright future in which architects can be inspired by traditional joinery techniques while designing new, sustainable and efficiently assembled buildings.

If we want to reach above vision, developing digital tools that enable architects to more easily design sophisticated timber-based connection systems that are transferable to digital fabrication is important. With this line of reasoning, Reindeer is a small contribution to an increase in productivity and a decrease in construction and demolition waste.

12. Conclusion

This study set out to develop a parametric toolkit for visual programming that was customised for detailing fabrication-ready timber structures. The aim of the toolkit was to reduce the time and algorithmic knowledge required for preparing a model for timber detailing and transferring a detailed timber structure to digital fabrication.

Three arguments support the notion that Reindeer reached its aim:

- Generation of topology is automated via the assembly component
- BTLx-generation is integrated into the design tools, which reduces the need for additional work when transferring a design to fabrication
- The JointSearch method does not require advanced algorithmic knowledge

The case structures have exemplified and proved that the toolkit works and is applicable for various structures and functions. However, a limitation of this study is the absence of feedback from external users. Reindeer was published while finalising this study, and future feedback will determine the degree that Reindeer has reduced the required algorithmic knowledge.

Many of the described and developed solutions relate to parallel developed work by CITA, Front, and DesignToProduction. The main contribution to the field of practitioners has been the development of a freely available toolkit. Thus, the contribution also pertains to democratising known but complicated parametric patterns[9].

The fact that JointSearch is independent of global topology renders the parametric model significantly less fragile. The illustrated examples have proven JointSearch to be both flexible and precise in its filtering. Further, the method is material-independent. Thus, JointSearch is considered the main methodological contribution to the field of parametric modelling research.

By conducting this study, I am convinced that AAD and digital fabrication is the key to making our building industry sustainable while maintaining the flexibility of our design. To realise this vision, we are dependent not only on the pioneers but most importantly the masses. Reindeer is a small step towards further democratising the use of AAD and digital fabrication.

The toolkit reduces the amount of work spent on redundant tasks. The hours earned can be spent producing better architecture. However, architecture continues to be created by architects; tools are tools that simplify, inform and strengthen an intellectual process.

13. Further work

This study started with a practical approach and evolved into a more theoretical approach. The result is a toolkit that is based on a user's perspective. TimberProcessingTools, especially JointSearch, has shown potential. A natural progression of this work is to investigate how Reindeer's approach can be combined with other existing methods. Two main directions are identified.

1. *Custom Key-values*

Combining Reindeer with Elefront's[36] custom key-value attributing would increase the capability of the tool. If the element had a custom key-value input, the parameter could be employed for search criteria and output via the joint.

2. *Topologic's topology class structure and Reindeer*

Combining Topologic's[32] class structure and Reindeer's JointSearch would drastically increase Reindeer's applications. Joints do not need to be a point-node and its 1D-Elements but can also be

- EdgeJoint: Two or more faces meet.
- PointNode: A 1D element intersects a face.
- A cell (e.g. a room) intersects another cell

In practical terms, these joints can describe the connections between connected Cross-laminated Timber (CLT) plates or a connection between a CLT wall and a beam. If the topology is provided by Topologic but enhanced with structural properties, Reindeer would, for example, be able to filter and distinguish all connections in a CLT structure.

Reindeer is currently designed for timber structures. A future tool that combines the best from Elefront, Topologic and Reindeer's JointSearch, deserves to be material- and profession-independent, e.g., is applicable to detailing a ship's steel structure. However, this development requires a higher abstraction level and is at risk of abandoning its original intention of democratising AAD.

References

Chronologically sorted

- [1] Barbosa F. Reinventing construction: A route to higher productivity. McKinsey&Company; 2017.
- [2] Birol F. Towards a zero-emission, efficient, and resilient buildings and construction sector. Global status report 2017. UN Environment; 2017.
- [3] Construction and demolition waste - Environment - European Commission n.d. http://ec.europa.eu/environment/waste/construction_demolition.htm (accessed June 26, 2019).
- [4] Carpo M. The alphabet and the algorithm. MIT Press; 2011.
- [5] Scheurer F. Digital craftsmanship: from thinking to modeling to building. Digital Workflows in Architecture Birkhäuser, Basel 2013:110–129. <https://doi.org/DOI:10.1515/9783034612173.110>.
- [6] Carpo M. The Second Digital Turn. Mit Press; 2017. <https://doi.org/10.2307/j.ctt1w0db6f>.
- [7] Mork JH, Luczkowski M, Manum B, Rønnquist A. Toward Mass Customized Architecture. Applying Principles of Mass Customization While Designing Site-Specific, Customer-Inclusive and Bespoke Timber Structures. Digital Wood Design 2019:221–249. https://doi.org/10.1007/978-3-030-03676-8_7.
- [8] Aish R. First build your tools. Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design 2013:36–49.
- [9] Woodbury R. Elements of parametric design 2010.
- [10] Scheurer F. Materialising complexity. Architectural Design 2010;80:86–93.
- [11] An Overview of GenerativeComponents - GenerativeComponents Community Wiki - GenerativeComponents - Bentley Communities n.d. https://communities.bentley.com/products/products_generativecomponents/w/generative_components_community_wiki (accessed June 8, 2019).
- [12] Krymsky Y. The Architecture Software Revolution: From One Size Fits All to DIY. ArchDaily 2015. <http://www.archdaily.com/778619/the-architecture-software-revolution-from-one-size-fits-all-to-diy> (accessed January 18, 2017).
- [13] Dynamo Studio | Computational BIM Design | Autodesk n.d. <https://www.autodesk.com/products/dynamo-studio/overview> (accessed July 3, 2019).
- [14] Aish R. From intuition to precisison. From Intuition to Precision 2005.
- [15] Oxman R. Thinking difference: Theories and models of parametric design thinking. Design Studies 2017;52:13. <https://doi.org/10.1016/j.destud.2017.06.001>.
- [16] Usai S, Stehling H. La Seine Musicale. In: De Rycke K, Gengnagel C, Baverel O, Burry J, Mueller C, Nguyen MM, et al., editors. Humanizing Digital Reality: Design Modelling Symposium Paris 2017, Singapore: Springer Singapore; 2018, p. 201–9. https://doi.org/10.1007/978-981-10-6611-5_18.
- [17] Jabi W, Soe S, Theobald P, Aish R, Lannon S. Enhancing parametric design through non-manifold topology. Design Studies 2017;52:96–114. <https://doi.org/10.1016/j.destud.2017.04.003>.
- [18] Skullestad JL, Bohne RA, Lohne J. High-rise Timber Buildings as a Climate Change Mitigation Measure – A Comparative LCA of Structural System Alternatives. Energy Procedia 2016;96:112–23. <https://doi.org/10.1016/j.egypro.2016.09.112>.
- [19] Robeller CWM. Integral Mechanical Attachment for Timber Folded Plate Structures. PHD. ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2015.
- [20] Menges A, Schwinn T, Krieg O, Willmann J, Gramazio F, Kohler M, et al. Advancing Wood Architecture: A Computational Approach. 2016.

- [21] Woodpecker. Food4Rhino 2013. <https://www.food4rhino.com/app/woodpecker> (accessed June 16, 2019).
- [22] Makkonen M. Stakeholder perspectives on the business potential of digitalization in the wood products industry. *BioProducts Business* 2018:63–80.
- [23] Chang T-C, Wysk RA. *Computer-aided manufacturing*. Prentice Hall PTR; 1997.
- [24] Stehling H, Scheurer F, Roulier J, Geglo H, Hofman M. From lamination to assembly modelling the seine musicale. *Fabricate: Rethinking Design and Construction* 2017;3. <https://doi.org/DOI: 10.2307/j.ctt1n7qkg7.39>.
- [25] Single Span Beam – Karamba3D n.d. <http://www.karamba3d.com/examples/simple/single-span-beam/> (accessed December 1, 2019).
- [26] Preisinger C, Heimrath M. Karamba—A Toolkit for Parametric Structural Design. *Structural Engineering International* 2014;24:217–21. <https://doi.org/10.2749/101686614X13830790993483>.
- [27] Archicad – Rhinoceros – Grasshopper Connection n.d. <https://www.graphisoft.com/archicad/rhino-grasshopper/> (accessed October 15, 2019).
- [28] Poinet P, Nicholas P, Tamke M, Thomsen M. Multi-Scalar Modelling for Free-form Timber Structures 2016. <https://doi.org/10.5281/zenodo.2611218>.
- [29] Svilans T, Poinet P, Tamke M, Ramsgaard Thomsen M. A Multi-scalar Approach for the Modelling and Fabrication of Free-Form Glue-Laminated Timber Structures. In: De Rycke K, Gengnagel C, Baverel O, Burry J, Mueller C, Nguyen MM, et al., editors. *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, Singapore: Springer Singapore; 2018, p. 247–57. https://doi.org/10.1007/978-981-10-6611-5_22.
- [30] Poinet P, Tamke M, Thomsen M, Scheurer F, Fisher A. Schema-Based Workflows and Inter-Scalar Search Interfaces for Building Design. 2018.
- [31] Aish R, Pratap A. Spatial information modeling of buildings using non-manifold topology with ASM and DesignScript. na; 2012.
- [32] Aish R, Jabi W, Lannon S, Wardhana N, Chatzivasileiadi A. Topologic: tools to explore architectural topology. *Advanced in architectural geometry* 2018, Gothenburg: 2018.
- [33] Learning – Topologic n.d. [//topologic.app/learning/](http://topologic.app/learning/) (accessed December 16, 2019).
- [34] Morpheus Hotel, Macau – a paradigm shift in computational engineering - Piermarini - 2018 - Steel Construction - Wiley Online Library n.d. <https://onlinelibrary.wiley.com/doi/abs/10.1002/stco.201810025> (accessed June 20, 2019).
- [35] Rhino.Inside n.d. <https://www.rhino3d.com/inside> (accessed October 15, 2019).
- [36] van der Heijden R. *Parametric Building Information Generation for Design and Construction*, 2015.
- [37] Elefront. Food4Rhino 2013. <https://www.food4rhino.com/app/elefront> (accessed June 20, 2019).
- [38] Regulations concerning the degrees of Philosophiae Doctor (PhD) and Philosophiae Doctor (PhD) in artistic research at the Norwegian University of Science and Technology (NTNU) - Lovdata n.d. <https://lovdata.no/dokument/SFE/forskrift/2018-12-05-1878?q=ntnu%20phd> (accessed November 20, 2019).
- [39] Aasheim E. Glulam Trusses for Olympic Arenas, Norway. *Structural Engineering International* 1993;3:86–7. <https://doi.org/doi:10.2749/101686693780612394>.
- [40] Aasheim E. Development of timber bridges in the Nordic countries. *Proc. of the 6th World Conference on Timber Engineering*, Citeseer; 2000.
- [41] Abrahamsen R, AS ML. Mjøstva arnet-Construction of an 81 m tall timber building. *International House Forum*, 2017.

- [42] Conceptual Structural Design - Department of Structural Engineering - NTNU n.d. <https://www.ntnu.edu/kt/research/csdg> (accessed January 19, 2020).
- [43] Dynamo vs. Grasshopper: Comparison. n.d.
- [44] Grasshopper Assembly for v5 - Visual Studio Marketplace n.d. <https://marketplace.visualstudio.com/items?itemName=McNeel.GrasshopperAssemblyforv5> (accessed May 23, 2019).
- [45] Hejlsberg A, Wiltamuth S, Golde P. The C# programming language. Adobe Press; 2006.
- [46] Goldberg A, Robson D. Smalltalk-80: the language and its implementation. Addison-Wesley Longman Publishing Co., Inc.; 1983.
- [47] Build software better, together. GitHub n.d. <https://github.com> (accessed May 14, 2019).
- [48] Brown K. What Is GitHub, and What Is It Used For? How-To Geek n.d. <https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/> (accessed May 14, 2019).
- [49] Mork JH, Toda Y, Izumi B, Luczkowski M, Hillersøy Dyvik S. Reindeer 0.5. Zenodo; 2019. <https://doi.org/10.5281/zenodo.3567051>.
- [50] Stehling H, Scheurer F, Roulier J. Bridging the gap from CAD to CAM: Concepts, Caveats and a new Grasshopper Plug-in. Fabricate 2014: Negotiating Design & Making. DGO-Digital original, UCL Press; 2017, p. 52–59.
- [51] Front Inc. Front Inc n.d. <https://www.frontinc.com/> (accessed June 20, 2019).
- [52] Hadid Z. The Morpheus Hotel: From Design to Production: Live Webinar on Vimeo n.d. <https://vimeo.com/203509846> (accessed April 30, 2017).
- [53] Smart Clustering - Smart Solutions Network n.d. <http://www.smart-solutions-network.com/page/smart-clustering> (accessed June 20, 2019).
- [54] Tamke M, Thomsen M. Digital wood craft. T. Tidafi and T. Dorta (eds) Joining Languages, Cultures and Visions: CAADFutures 2009, PUM, 2009, pp. 673- 686, 2009.
- [55] Tamke, Tamke M, Thomsen R, Thomsen M. Designing Parametric Timber. Architecture in Computro [26th eCAADe Conference Proceedings / ISBN 978-0-9541183-7-2] Antwerpen (Belgium) 17-20 September 2008, pp. 609-616, 2008.
- [56] Poinet P, Nicholas P, Tamke M, Thomsen MR. Multi-scalar modelling for free-form timber structures. Proceedings of IASS Annual Symposia, vol. 2016, International Association for Shell and Spatial Structures (IASS); 2016, p. 1–10.
- [57] E W. Principles of Multiscale Modeling. 1 edition. Cambridge, UK; New York: Cambridge University Press; 2011.
- [58] Svilans T, Tamke M, Thomsen MR, Runberger J, Strehlke K, Antemann M. New Workflows for Digital Timber. Digital Wood Design 2019:93–134. https://doi.org/10.1007/978-3-030-03676-8_3.
- [59] Speckle: Data Platform for AEC. Speckle Is the Open Source Data Platform for Architecture, Engineering and Construction n.d. <https://speckle.works/> (accessed June 16, 2019).
- [60] Masuda H. Topological operators and Boolean operations for complex-based nonmanifold geometric models. Computer-Aided Design 1993;25:119–29. [https://doi.org/10.1016/0010-4485\(93\)90097-8](https://doi.org/10.1016/0010-4485(93)90097-8).
- [61] Chatzivasileiadi A, Wardhana NM, Jabi W, Aish R, Lannon S. A Review of 3D Solid Modeling Software Libraries for Non-Manifold Modeling. Manuscript Submitted for Publication 2018.
- [62] Robeller C. Integral Mechanical Attachment for Timber Folded Plate Structures n.d.:181.
- [63] Timber Plate Structures (TPS). Food4Rhino 2017. <https://www.food4rhino.com/app/timber-plate-structures-tps> (accessed July 3, 2019).
- [64] Robeller C. Timber Plate Shell Structures: A Digital Resurgence of Traditional Joining Methods. Digital Wood Design, Springer; 2019, p. 1117–1133.

- [65] Loveridge RA, Gugger H. Process Bifurcation and the Digital Chain in Architecture. EPFL; 2012.
- [66] Robotics HAL. Software. HAL Robotics | Versatile Robot Programming & Simulation Solutions n.d. <https://hal-robotics.com/> (accessed July 3, 2019).
- [67] Association for Robots in Architecture | KUKA|prc n.d. <https://www.robotsinarchitecture.org/kuka-prc> (accessed July 3, 2019).
- [68] design2machine - the data transfer interface for wood constructions n.d. <https://design2machine.com/> (accessed March 11, 2019).
- [69] Thomsen M, Tamke M. Narratives of making: thinking practice led research in architecture. 2009.
- [70] Van Rossum G. Python Programming Language. USENIX annual technical conference, vol. 41, 2007, p. 36.
- [71] Ladybug Tools | Home Page n.d. <https://www.ladybug.tools/index.html> (accessed May 14, 2019).
- [72] Verbeke J. Research by Design is up and running. Revista Lusófona de Arquitectura e Educação 2012;0:110–119.
- [73] Candy L. Practice Based Research: A Guide. Creativity and Cognition Studios Report 2006;1.
- [74] How do reindeer find enough food in the tundra? HowStuffWorks 2008. <https://animals.howstuffworks.com/mammals/reindeer-find-food.htm> (accessed April 8, 2019).
- [75] Staub-French S, Poirier PeEA, Calderon F, Chikhi I, Arch M, Zadeh P, et al. Building Information Modeling (BIM) and Design for Manufacturing and Assembly (DfMA) for Mass Timber Construction 2018.
- [76] Scheurer F, Stehling H, Tschümperlin F, Antemann M. Design for Assembly–Digital Prefabrication of Complex Timber Structures. Proceedings of IASS Annual Symposia, vol. 2013, International Association for Shell and Spatial Structures (IASS); 2013, p. 1–7.
- [77] Guy B, Shell S, Esherick H. Design for deconstruction and materials reuse. Proceedings of the CIB Task Group 2006;39:189–209.

Alphabetically sorted books/articles

Aasheim E. Development of timber bridges in the Nordic countries. Proc. of the 6th World Conference on Timber Engineering, Citeseer; 2000. [40]

Aasheim E. Glulam Trusses for Olympic Arenas, Norway. Structural Engineering International 1993;3:86–7. <https://doi.org/doi:10.2749/101686693780612394>. [39]

Abrahamsen R, AS ML. Mjøstva arnet-Construction of an 81 m tall timber building. International House Forum, 2017. [41]

Aish R, Jabi W, Lannon S, Wardhana N, Chatzivasileiadi A. Topologic: tools to explore architectural topology. Advanced in architectural geometry 2018, Gothenburg: 2018. [32]

Aish R, Pratap A. Spatial information modeling of buildings using non-manifold topology with ASM and DesignScript. na; 2012. [31]

Aish R. First build your tools. Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design 2013:36–49. [8]

- Aish R. From intuition to precision. *From Intuition to Precision* 2005. [14]
- Barbosa F. Reinventing construction: A route to higher productivity. McKinsey&Company; 2017. [1]
- Birol F. Towards a zero-emission, efficient, and resilient buildings and construction sector. *Global status report 2017*. UN Environment; 2017. [2]
- Candy L. Practice Based Research: A Guide. *Creativity and Cognition Studios Report* 2006;1. [73]
- Carpo M. *The alphabet and the algorithm*. MIT Press; 2011. [4]
- Carpo M. *The Second Digital Turn*. Mit Press; 2017. <https://doi.org/10.2307/j.ctt1w0db6f>. [6]
- Chang T-C, Wysk RA. *Computer-aided manufacturing*. Prentice Hall PTR; 1997. [23]
- Chatzivasileiadi A, Wardhana NM, Jabi W, Aish R, Lannon S. A Review of 3D Solid Modeling Software Libraries for Non-Manifold Modeling. *Manuscript Submitted for Publication* 2018. [61]
- Dynamo vs. Grasshopper: Comparison. n.d. [43]
- E W. *Principles of Multiscale Modeling*. 1 edition. Cambridge, UK ; New York: Cambridge University Press; 2011. [57]
- Goldberg A, Robson D. *Smalltalk-80: the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc.; 1983. [46]
- Guy B, Shell S, Esherick H. Design for deconstruction and materials reuse. *Proceedings of the CIB Task Group* 2006;39:189–209. [77]
- Hejlsberg A, Wiltamuth S, Golde P. *The C# programming language*. Adobe Press; 2006. [45]
- Jabi W, Soe S, Theobald P, Aish R, Lannon S. Enhancing parametric design through non-manifold topology. *Design Studies* 2017;52:96–114. <https://doi.org/10.1016/j.destud.2017.04.003>. [17]
- Loveridge RA, Gugger H. *Process Bifurcation and the Digital Chain in Architecture*. EPFL; 2012. [65]
- Makkonen M. Stakeholder perspectives on the business potential of digitalization in the wood products industry. *BioProducts Business* 2018:63–80. [22]
- Masuda H. Topological operators and Boolean operations for complex-based nonmanifold geometric models. *Computer-Aided Design* 1993;25:119–29. [https://doi.org/10.1016/0010-4485\(93\)90097-8](https://doi.org/10.1016/0010-4485(93)90097-8). [60]
- Menges A, Schwinn T, Krieg O, Willmann J, Gramazio F, Kohler M, et al. *Advancing Wood Architecture: A Computational Approach*. 2016. [20]
- Mork JH, Luczkowski M, Manum B, Rønquist A. *Toward Mass Customized Architecture. Applying Principles of Mass Customization While Designing Site-Specific, Customer-*

- Inclusive and Bespoke Timber Structures. *Digital Wood Design* 2019:221–249. https://doi.org/10.1007/978-3-030-03676-8_7. [7]
- Mork JH, Toda Y, Izumi B, Luczkowski M, Hillersøy Dyvik S. Reindeer 0.5. Zenodo; 2019. <https://doi.org/10.5281/zenodo.3567051>. [49]
- Oxman R. Thinking difference: Theories and models of parametric design thinking. *Design Studies* 2017;52:13. <https://doi.org/10.1016/j.destud.2017.06.001>. [15]
- Poinet P, Nicholas P, Tamke M, Thomsen M. Multi-Scalar Modelling for Free-form Timber Structures 2016. <https://doi.org/10.5281/zenodo.2611218>. [28]
- Poinet P, Nicholas P, Tamke M, Thomsen MR. Multi-scalar modelling for free-form timber structures. *Proceedings of IASS Annual Symposia*, vol. 2016, International Association for Shell and Spatial Structures (IASS); 2016, p. 1–10. [56]
- Poinet P, Tamke M, Thomsen M, Scheurer F, Fisher A. Schema-Based Workflows and Inter-Scalar Search Interfaces for Building Design. 2018. [30]
- Preisinger C, Heimrath M. Karamba—A Toolkit for Parametric Structural Design. *Structural Engineering International* 2014;24:217–21. <https://doi.org/10.2749/101686614X13830790993483>. [26]
- Robeller C. Integral Mechanical Attachment for Timber Folded Plate Structures n.d.:181. [62]
- Robeller C. Timber Plate Shell Structures: A Digital Resurgence of Traditional Joining Methods. *Digital Wood Design*, Springer; 2019, p. 1117–1133. [64]
- Robeller CWM. Integral Mechanical Attachment for Timber Folded Plate Structures. PHD. ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2015. [19]
- Scheurer F, Stehling H, Tschümperlin F, Antemann M. Design for Assembly–Digital Prefabrication of Complex Timber Structures. *Proceedings of IASS Annual Symposia*, vol. 2013, International Association for Shell and Spatial Structures (IASS); 2013, p. 1–7. [76]
- Scheurer F. Digital craftsmanship: from thinking to modeling to building. *Digital Workflows in Architecture* Birkhäuser, Basel 2013:110–129. <https://doi.org/DOI:10.1515/9783034612173.110>. [5]
- Scheurer F. Materialising complexity. *Architectural Design* 2010;80:86–93. [10]
- Skullestad JL, Bohne RA, Lohne J. High-rise Timber Buildings as a Climate Change Mitigation Measure – A Comparative LCA of Structural System Alternatives. *Energy Procedia* 2016;96:112–23. <https://doi.org/10.1016/j.egypro.2016.09.112>. [18]
- Staub-French S, Poirier PeEA, Calderon F, Chikhi I, Arch M, Zadeh P, et al. Building Information Modeling (BIM) and Design for Manufacturing and Assembly (DfMA) for Mass Timber Construction 2018. [75]

- Stehling H, Scheurer F, Roulier J, Geglo H, Hofman M. From lamination to assembly modelling the seine musicale. *Fabricate: Rethinking Design and Construction* 2017;3. <https://doi.org/DOI: 10.2307/j.ctt1n7qkg7.39>. [24]
- Stehling H, Scheurer F, Roulier J. Bridging the gap from CAD to CAM: Concepts, Ceaveats and a new Grasshopper Plug-in. *Fabricate* 2014: Negotiating Design & Making. DGO-Digital original, UCL Press; 2017, p. 52–59. [50]
- Svilans T, Poinet P, Tamke M, Ramsgaard Thomsen M. A Multi-scalar Approach for the Modelling and Fabrication of Free-Form Glue-Laminated Timber Structures. In: De Rycke K, Gengnagel C, Baverel O, Burry J, Mueller C, Nguyen MM, et al., editors. *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, Singapore: Springer Singapore; 2018, p. 247–57. https://doi.org/10.1007/978-981-10-6611-5_22. [29]
- Svilans T, Tamke M, Thomsen MR, Runberger J, Strehlke K, Antemann M. New Workflows for Digital Timber. *Digital Wood Design* 2019:93–134. https://doi.org/10.1007/978-3-030-03676-8_3. [58]
- Tamke M, Thomsen M. Digital wood craft. T. Tidafi and T. Dorta (eds) *Joining Languages, Cultures and Visions: CAADFutures 2009*, PUM, 2009, pp. 673- 686, 2009. [54]
- Tamke, Tamke M, Thomsen R, Thomsen M. Designing Parametric Timber. *Architecture in Computro [26th eCAADe Conference Proceedings / ISBN 978-0-9541183-7-2]* Antwerpen (Belgium) 17-20 September 2008, pp. 609-616, 2008. [55]
- Thomsen M, Tamke M. Narratives of making: thinking practice led research in architecture. 2009. [69]
- Usai S, Stehling H. La Seine Musicale. In: De Rycke K, Gengnagel C, Baverel O, Burry J, Mueller C, Nguyen MM, et al., editors. *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, Singapore: Springer Singapore; 2018, p. 201–9. https://doi.org/10.1007/978-981-10-6611-5_18. [16]
- Van der Heijden R. *Parametric Building Information Generation for Design and Construction*, 2015. [36]
- Van Rossum G. Python Programming Language. *USENIX annual technical conference*, vol. 41, 2007, p. 36. [70]
- Verbeke J. Research by Design is up and running. *Revista Lusófona de Arquitectura e Educação* 2012;0:110–119. [72]
- Woodbury R. *Elements of parametric design* 2010. [9]

Alphabetically sorted Web-References

An Overview of GenerativeComponents - GenerativeComponents Community Wiki - GenerativeComponents - Bentley Communities n.d. https://communities.bentley.com/products/products_generativecomponents/w/generative_components_community_wiki (accessed June 8, 2019). [11]

Archicad – Rhinoceros – Grasshopper Connection n.d. <https://www.graphisoft.com/archicad/rhino-grasshopper/> (accessed October 15, 2019). [27]

Association for Robots in Architecture | KUKA|prc n.d. <https://www.robotsinarchitecture.org/kuka-prc> (accessed July 3, 2019). [67]

Brown K. What Is GitHub, and What Is It Used For? How-To Geek n.d. <https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/> (accessed May 14, 2019). [48]

Build software better, together. GitHub n.d. <https://github.com> (accessed May 14, 2019). [47]

Conceptual Structural Design - Department of Structural Engineering - NTNU n.d. <https://www.ntnu.edu/kt/research/csdg> (accessed January 19, 2020). [42]

Construction and demolition waste - Environment - European Commission n.d. http://ec.europa.eu/environment/waste/construction_demolition.htm (accessed June 26, 2019). [3]

Design2machine - the data transfer interface for wood constructions n.d. <https://design2machine.com/> (accessed March 11, 2019). [68]

Dynamo Studio | Computational BIM Design | Autodesk n.d. <https://www.autodesk.com/products/dynamo-studio/overview> (accessed July 3, 2019). [13]

Elefront. Food4Rhino 2013. <https://www.food4rhino.com/app/elefront> (accessed June 20, 2019). [37]

Front Inc. Front Inc n.d. <https://www.frontinc.com/> (accessed June 20, 2019). [51]

Grasshopper Assembly for v5 - Visual Studio Marketplace n.d. <https://marketplace.visualstudio.com/items?itemName=McNeel.GrasshopperAssemblyforv5> (accessed May 23, 2019). [44]

Hadid Z. The Morpheus Hotel: From Design to Production: Live Webinar on Vimeo n.d. <https://vimeo.com/203509846> (accessed April 30, 2017). [52]

How do reindeer find enough food in the tundra? HowStuffWorks 2008. <https://animals.howstuffworks.com/mammals/reindeer-find-food.htm> (accessed April 8, 2019). [74]

Krymsky Y. The Architecture Software Revolution: From One Size Fits All to DIY. ArchDaily 2015. <http://www.archdaily.com/778619/the-architecture-software-revolution-from-one-size-fits-all-to-diy> (accessed January 18, 2017). [12]

Ladybug Tools | Home Page n.d. <https://www.ladybug.tools/index.html> (accessed May 14, 2019). [71]

Learning – Topologic n.d. [//topologic.app/learning/](https://topologic.app/learning/) (accessed December 16, 2019). [33]

Morpheus Hotel, Macau – a paradigm shift in computational engineering - Piermarini - 2018 - Steel Construction - Wiley Online Library n.d. <https://onlinelibrary.wiley.com/doi/abs/10.1002/stco.201810025> (accessed June 20, 2019). [34]

Regulations concerning the degrees of Philosophiae Doctor (PhD) and Philosophiae Doctor (PhD) in artistic research at the Norwegian University of Science and Technology (NTNU) - Lovdata n.d. <https://lovdata.no/dokument/SFE/forskrift/2018-12-05-1878?q=ntnu%20phd> (accessed November 20, 2019). [38]

Rhino.Inside n.d. <https://www.rhino3d.com/inside> (accessed October 15, 2019). [35]

Robotics HAL. Software. HAL Robotics | Versatile Robot Programming & Simulation Solutions n.d. <https://hal-robotics.com/> (accessed July 3, 2019). [66]

Single Span Beam – Karamba3D n.d. <http://www.karamba3d.com/examples/simple/single-span-beam/> (accessed December 1, 2019). [25]

Smart Clustering - Smart Solutions Network n.d. <http://www.smart-solutions-network.com/page/smart-clustering> (accessed June 20, 2019). [53]

Speckle: Data Platform for AEC. Speckle Is the Open Source Data Platform for Architecture, Engineering and Construction n.d. <https://speckle.works/> (accessed June 16, 2019). [59]

Timber Plate Structures (TPS). Food4Rhino 2017. <https://www.food4rhino.com/app/timber-plate-structures-tps> (accessed July 3, 2019). [63]

Woodpecker. Food4Rhino 2013. <https://www.food4rhino.com/app/woodpecker> (accessed June 16, 2019). [21]

IV. APPENDICES

PAPERS

Paper I

Toward mass customized architecture. Applying principles of mass customization while designing site-specific, customer-inclusive and bespoke timber structures

Paper II

JointSearch: Efficient parametric detailing preparation through user-defined and property-based joint type filtering

Paper III

Parametric timber toolkit: A timber tailored approach

TOWARD MASS CUSTOMIZED ARCHITECTURE. Applying principles of mass customization while designing site-specific, customer-inclusive and bespoke timber structures

John Haddal Mork*, Marcin Luczkowski, Bendik Manum and Anders Rønnequist

* Correspondence: PhD-candidate, john.h.mork@ntnu.no

Abstract: Mass customization is established in many industries, but are not yet integrated in architecture and the building industry. This article presents a parametric timber toolkit under development. A flexible toolkit for parametrically designed timber structures, a toolkit that simplifies and substantiates a continuous digital workflow from global shape to digital fabrication and assembly - A toolkit that requires parametric thinking, not only parametric modelling skills. The toolkit proposes solutions to four recurrent workflow-related challenges that limit efficiency and quality while designing timber structures. A series of built case projects are used to exemplify and explain the toolkit. An important finding discussed in the end of the article is that parametric modelling, and partly the toolkit, changes our conception of what is considered a similar structure.

Keywords: "Parametric design and fabrication strategies, CNC and Woodworking Technology, Parametric timber engineering"

1. Introduction

Is it possible to integrate the benefits of mass customization (MC) into architecture? How can architects apply the principles of MC while designing site-specific, customer-inclusive and bespoke timber structures? Mass customization, a term introduced by Stanley Davis in 1987 (Davis, 1997), was later redefined by Andreas M. Kaplan and Michael Haenlein as follows: "Visionary traditional MC in a strategy that creates value by some form of company-customer interaction at the design stage of the operations level to create customized products, following a hybrid strategy combining cost leadership and differentiation." (Kaplan & Haenlein, 2006)

A challenge that distinguishes architecture from other professions is the large design space that has to be offered. Naval architects always make variations of a ship with a hull, an engine, and a variety of equipment. Designers in the footwear industry always deal with feet of different sizes and customers with different uses and styles. In contrast, architectural design projects range from working on a bridge to a high-rise to small furniture. Johanna Daaboul stated that MC can be offered either via product variability or process variability (Daaboul, Cunha, Bernard, & Laroche, 2011). Offering both, a wide range of process variability and product variability, is a core service of being an architect and makes mass customization especially complex. First, every project site is unique and includes the functional requirements, locally available materials, climate and building culture of a given location to be unique. Second, the user and the user's needs and functional demands are unique. Adapting projects to their contexts is what makes architecture, architecture. Furthermore, architecture is a highly subjective task, partly a poetic process from a vision to a built structure, and mass customization must not remove such an important quality.

The use of Computer Numerical Controlled (Rapp & Johnson, 1980) (CNC) Machines has become widespread in the building industry (BI). CNC-machines are used for a large range of manufacturing operations, such as cutting, drilling and milling, and are digitally controlled. One of the biggest benefits of using such tools is that the production process is automated and uses the same amount of time manufacturing a set of unique components, such as a set of identical components. However, automated manufacturing of unique building components does not implicate automated, manufacturing ready computer drawings. In contrast, unique building components require unique drawings and demands instead of an automated strategy to be economically sustainable.

Computer-Aided Design (Burns, 1986) (CAD) has existed since the 80s, but in reality, it is more or less a digital, augmented variant of manual drawing. Later, Building Information Modeling (Kymmell, 2007) (BIM) was introduced. From dumb geometry, 3D geometry suddenly represented a building component—a building component aware of where it was, what function it had, optional data about cost, manufacturer, etc. According to Jared Banks, BIM enabled designers to spend more time designing and less time documenting and coordinating (Banks, 2016).

Digital parametric workflows, which widely appeared a decade ago, represent a dramatic new way of applying digital power while designing our built environment. Instead of drawing geometry, parametric

workflows let the user define the design as a series of decisions, systems and relationships (Krymsky, 2015). Hence, the digital design process has the potential of becoming more flexible than existing, partly standardized, BIM processes. Robert Aish described it as the computation era and argued that the objective of design computation is to overcome many of the limitations of BIM (Aish, 2013). We see a computational approach and parametric thinking as the next, decisive step to reach mass-customized architecture and BI.

Parametricism (P. Schumacher, 2008) is a buzzword that has been around for some years and is often associated with organic, prestigious buildings made by architects such as Zaha Hadid or Frank Gehry. However, parametric workflows have greater potential than creating signature architecture and are thoroughly described in Architectural Design's special edition, "Parametricism 2.0". Patrik Schumacher claims following: "In order to reverse the current marginalisation of Parametricism, it is necessary to relaunch it in a self-critical redirection as Parametricism 2.0. Parametricism is architecture's answer to contemporary, computationally empowered civilisation, and is the only architectural style that can take full advantage of the computational revolution that now drives all domains of society." (Patrik Schumacher, 2016)

Great examples of such methodology can be seen in the works of Shigeru Ban and Achim Menges. Shigeru Ban's famous Nine Bridges Country Club gridshell is realized with the help of DesignToProduction (consultant company) parameterization and automated manufacturing (Scheurer, 2013). Another outstanding example is Achim Menges, who applies parametric tools throughout the entire process while making his pavilions and structures. Nevertheless, the mentioned projects are more customized than those that are mass customized. Large budgets and partly research-funded projects make such projects happen.

What is mass customization in architecture? Parametric workflows are extremely efficient when established, but establishing such workflows in each project may not be economically justifiable. Hence, the goal must be to create a continuous parametric workflow from sketch to fabrication that is applicable to a wide range of projects. Through designing and building a series of timber structures according to an established parametric approach, we identified four recurrent workflow-related challenges that limit efficiency and quality while designing timber structures. These are also the challenges that often cancel a continuous workflow from sketch to fabrication and are the motivation to create the parametric timber toolkit:

1) Parametric Detailing: Finding a robust way to sort and identify geometric data that are time-consuming in each individual project. Among other reasons, data is sorted to be able to design the different details in a structure.

2) Tectonic Architecture: Processing timber components that are subtractive, meaning that the material is subtracted from a stock. In contrast, digital design is largely additive, e.g., extrusions, revolves, sweeps, etc. Thus, a recurrent design challenge focuses on how a component is physically manufactured.

3) Manufacturing Descriptions: Concerning timber building design, architects and structural engineers tend to output geometry that the manufacturer has to redraw. Geometry that does not include information about the manufacturing process. The result is redundant work and increases the chances of modeling errors.

4) Structural Analysis: Since the timber is orthotropic material and the most demanding phase in structural analysis is the connection design, the faster the structural analysis is implemented, the greater the chance of achieving a feasible project. Similarly, in manufacturing conversion, a challenge is to automatically convert geometry modeled by an architect to structural analysis.

"First build your tool" is the title of the abovementioned Aish's article (Aish, 2013). The objective of the ongoing research project has been to develop a flexible toolkit for parametrically designed timber structures, a toolkit that simplifies and substantiates a continuous digital workflow from global shape to digital fabrication and assembly. A toolkit that requires parametric thinking, not only parametric modeling skills.

The toolkit is based upon the process of designing timber structures, but many of the principles are general and apply to other materials. This article thoroughly describes the system and principles of using elements and nodes as a basis for most types of parametrically designed architectural structures. It describes how architects, engineers and manufacturers consider an element, and a shared solution is proposed. Furthermore, the article describes the algorithm that sorts details into detailing groups. A series of built case projects are used to exemplify and explain the toolkit. An important finding discussed in the end of the article is that parametric modeling, and partly the toolkit, changes our conception of what is considered a similar structure. A structure can be similar in parametric description but can vary much in function, form, scale and all other attributes. What appears as visually and radically different structures from an architectural point of view is very similar from an algorithmic point of view.

2. The computational workflow

2.1 Software platform

The toolkit is implemented in Grasshopper 3D (McNeel, 2015), an add on to Rhinoceros 3D (McNeel, 2009). The visual programming software was introduced in late 2007 and accelerated the architectural software revolution. (Krymsky, 2015). Currently, the toolkit is an in-house beta-version, but is planned to be released as an open-source plugin early 2019. Using C# and Visual Studio, the toolkit is now developed as a proper plugin. Earlier development, as described in the case-studies, have been a combination of Grasshopper and IronPython scripting. The current toolkit is based on C# classes, which corresponds to the detailing groups described in chapter 4.1. The export from Rhinoceros is based on Building Transfer Language (BTL), and will be described in chapter 4.4

Figure 1 and Figure 2 shows a simple example of the concept described in this paper. A few bars are connected in a node. If the bars are shorter than 925 mm, they are trimmed with an angle. The left side of Figure 1 shows the result in Rhinoceros. The right side of the figure shows the manufacturing ready BTL-export. Figure 2 shows the script in Grasshopper.

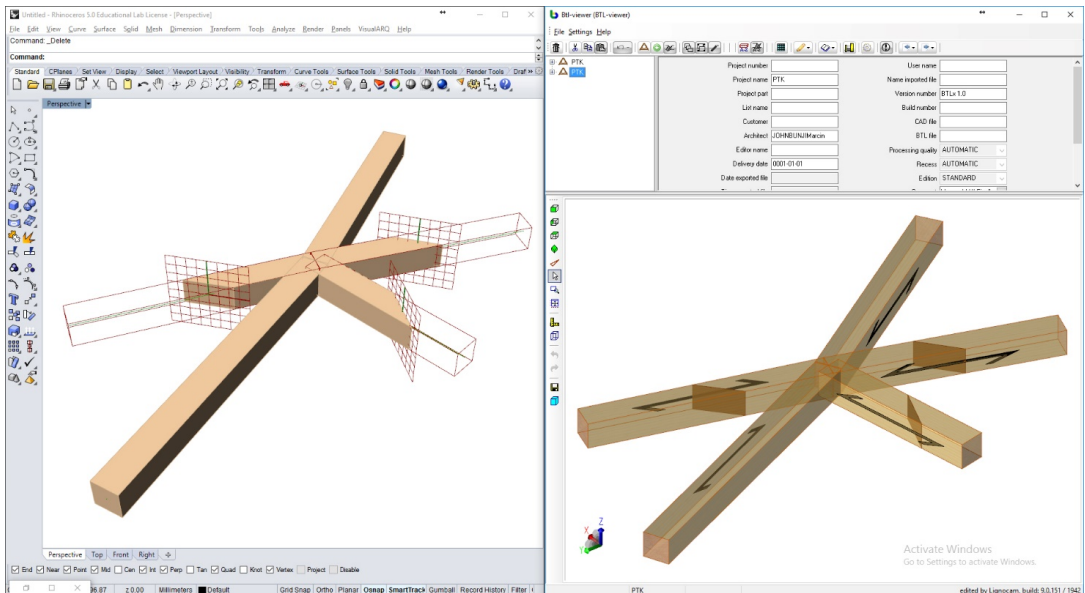


Figure 1: Rhinoceros to the left, BTL-viewer to the right

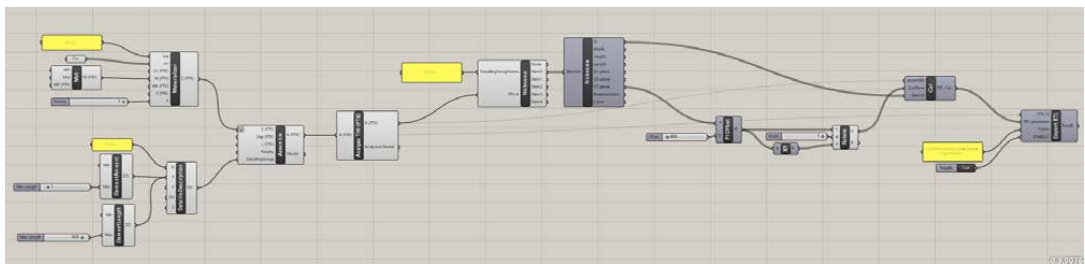


Figure 2: The grasshopper script required to create the geometry illustrated in figure 1

2.2 Workflow

To use parametric modelling as an efficient method, it is crucial to ensure a continuous flow from overall geometry, via detailing and structural analysis to manufacturing output. Furthermore, it is crucial to sustain a workflow that suits both architects, engineers and manufacturers. This chapter explains briefly the chosen computational workflow. The following chapters explains in-depth how the introduced challenges are solved and implemented.

The components and workflow are described in Figure 3, and the procedure is as follows:

Overall geometry

- 1) Centerline-geometry is generated with the help of Rhinoceros or conventional Grasshopper-components
- 2) The centerline-geometry is fed into one or multiple element components. Here, the centerline is materialized.
- 3) In the element components, the cross-section, cross-section orientation and material are defined.
- 4) The property description tools describe each detailing group. Here, the node-properties are also assigned.
- 5) The loads are defined.
- 6) The loads, the elements and the descriptions of the detailing groups are attached to the assembler. The assembler generates relations between the nodes and elements, generates the details and assigns the details to its detailing groups. This step relates to challenge 1 and are thoroughly described in chapter 4.

Structural analysis

- 7) Finite element analysis is performed using Karamba (Preisinger & Heimrath, 2014). However, the pre-processing and post-processing of the analysis is included in the toolkit. To integrate the structural analysis and the architectural design, the structural FE objects are being made simultaneously with definition of the geometrical elements and its components (material, cross section). More over the results from FE analysis are automatically send for post-processing. The components are checking the elements and joints according to the EC5 criteria and are informing user (designer) about utilization ratio and which combination of failure state (compression, tension, bending, combinations) is crucial.

Detailing

- 8) The respective detailing groups are extracted by a component. Here, the user can extract each member of the detail.
- 9) Properties from the members of the details are further extracted.
- 10) Subtractive tools detail the timber structure. This step relates to challenge 2 and 3. The Subtractive tools are described in chapter 5.

Output

- 11) The outputs from the subtractive operations can be used for visual inspection, FEA or BTL-export.

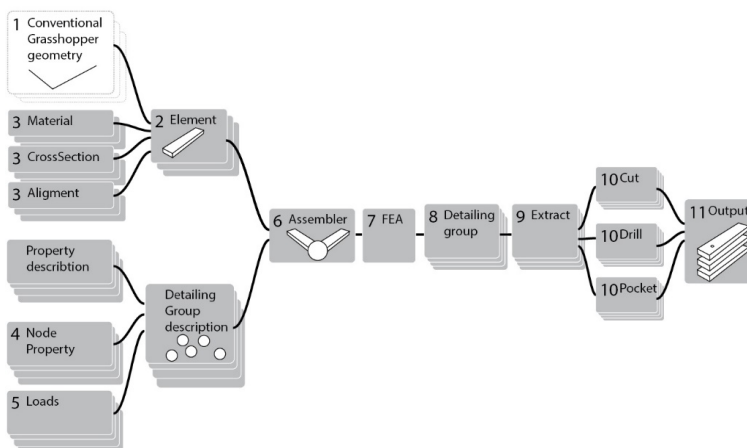


Figure 3: the computational workflow

3. Establishment of a shared parametric approach for architects, structural engineers and manufacturers

An established method of modeling digital structures is to start with the center-line geometry, modeling the theoretical center-line of any building component. Columns, beams, chords and bars are described by two points —the start and end points that construct a line. If two lines share the same points, the lines are connected neighbors. The shared point will then be a connection node. With only points, lines and geometric properties such as cross section and materials, a majority of the different kinds of structure can be described in architecture or engineering.

As in many other applications, the parametric timber toolkit is based on a system of curves and points. Curves can represent any building component with one dominant geometrical direction. In this article, these components will be described as elements. In buildings, elements are mostly attached by a physical connector, but digitally, they can be described as single points where elements start, end or intersect (hereby called nodes). See Figure 4.

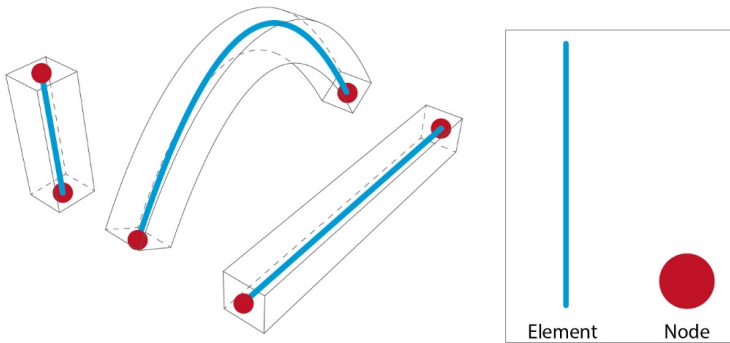


Figure 4: The article and toolkit's definition of elements and nodes

As a general description of a topology, the overall shape of the structure, the description of using curves and points, works very well. However, when architects, engineers and manufacturers are going into detailed design, analysis and manufacture of the structure, the described system is not optimal. The following is a generalized description of how geometry generation methods are preferred according to each of the building-process participant:

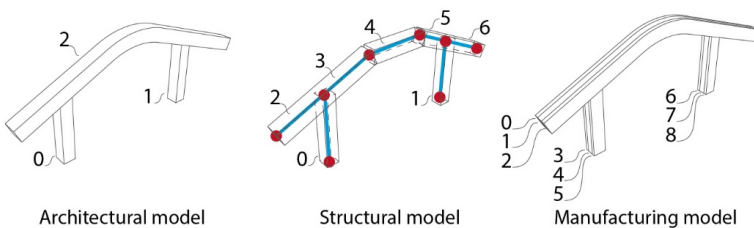


Figure 5: How architects, structural engineers and manufactures define an element

3.1 Architect's conception of an element

What you see, is what you get. For an average architect, it is enough to consider an element as a building component. The physical component ordered from the manufacturer is what is being considered as one element. The example in Figure 5 shows two columns and an arched beam. Depending on the detail-level, further detailing of the node and even the refinement of the elements might be required. Regardless, the architect can consider the illustrated structure as three elements.

3.2 Structural engineer's conception of an element

The structural engineers view the objects from two perspectives, physical and numerical. A numerical model always aims to simulate the physical behavior of the structure or at least its part (here called element). Currently, the numerical models are mostly built using the finite element method (FEM).

In general, (the physical model) one element is understood by the structural engineer as an object with material continuity that allows continuous stress distribution. The engineering simplification of the objects to the beam element is very intuitive and allows operating on force and deformation for estimating the element capacity. Two rules are essential:

- 1) Commonly, the beams are represented by the linear finite element; for representing curves, we divide it into a sufficient number of smaller finite elements.
- 2) The connection/continuity between elements is described by the nodes, and the finite elements have to begin or end in the nodes.

With the rules in mind, we see that the architectural model on the left side (Figure 5) does not fulfil the requirements for a structural analysis. First, the arch must be segmented into a polyline and fragmented into linear elements. Second, the beam must be divided where the columns are attached. That is, for a structural engineer, the structure consists of seven elements.

3.3 Manufacturer's conceptions of an element

The architectural conception of an element is also often sufficient for a manufacturer. One element is one building component. However, sometimes, an element consists of multiple sub-elements joined together as one element. There are many reasons contributing to this phenomenon. For glulam-manufacturers, the reasons to split the element can be due to size limitations or being able to mill a complex detail.

For such purposes, the element must be divided into multiple elements, but not likely in the same manner as for the structural engineer.

If the structure in the example were detailed and slotted in plates and dowels as nodes, the manufacturer would likely prefer to construct each element from at least three sub-elements. In this way, space for the plates could be easily milled, thereby being glued together as one element. The result is seen in Figure 5

3.4 A solution that integrates the three conceptions

How is this problem solved? How does one make a model that satisfy these three approaches? Owing to object-oriented programming (Goldberg & Robson, 1983), a class-structure is developed to contain the three ways of describing an element. The master element is similar to the physical component delivered to the building site. However, a subclass is also storing one or more structural elements within the main element. Similarly, one or more manufacturing sub element is stored in the main element. The system is illustrated in Figure 6. Note that the structural sub element is subdivided transversal while the manufacturing sub element is subdivided longitudinal.

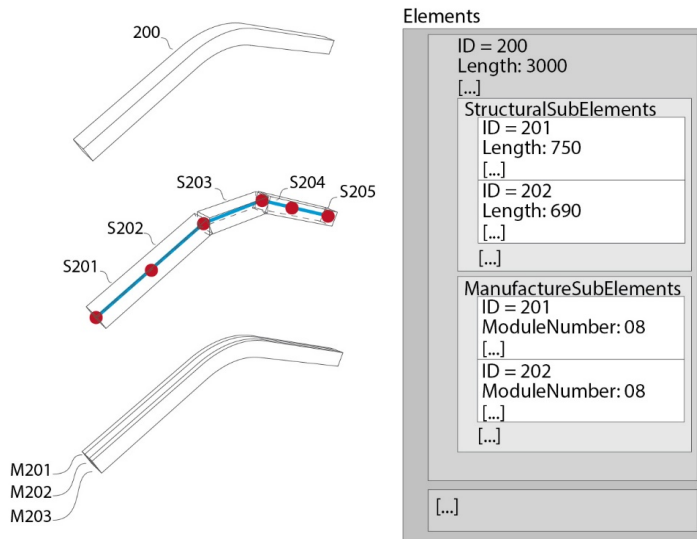


Figure 6: Object-oriented programming gathers architects, structural engineers and manufacturers' conceptions of an element in one master element.

In addition, geometric representations of the element and a series of other data are generated and stored in the element object. First, geometric data such as length, height, width and local planes and vectors are stored. Second, cross-sectional data, material properties, manufacturing data and other metadata are stored in the element object, making it trivial to extract relevant data when detailing or analyzing a structure.

4. Rethinking parametric detailing: Introducing property-based detailing groups

The beauty of parametric modeling is the ability to create a limited amount of parametric details that become valid for numerous instances in a given structure. The challenge is to create a robust sorting algorithm that updates any geometry correctly. The first level of complexity is to make the sorting work for a given topology; a more complex sorting is to make it work for any kind of topology, allowing the topology of a structure to change fundamentally, but the details are updated and distributed correctly. Through the research project, a general and flexible sorting algorithm for parametric modeling has been developed. The method is surprisingly simple.

However, to describe the concept properly, let us start from the beginning. What is a detail? When detailing a node, one is also detailing the ends of the connected elements. Hence, a detail of the node influences the node, and due to, for example, holes in the element, the detail changes the element size. Mutually, when detailing an element, the connected nodes may be affected. Thus, the system defines a detail as a coupled system of elements and nodes, that is, either a node and its elements or an element and its nodes. The principle is shown in Figure 7.

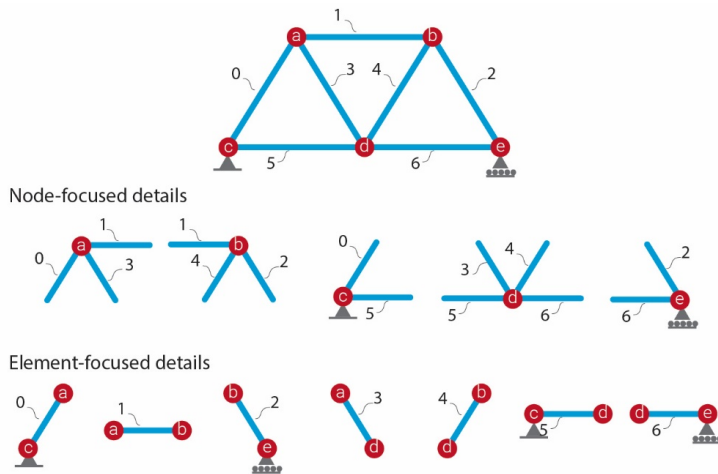


Figure 7: Node-focused details and element-focused details. A node has elements and an element has nodes.

Every type of structure, both parametrically and conventionally designed, contains a set of details. How the column meets the ground, how the members in a gridshell are connected, and how the top-chord is connected to the two bars are examples of details. However, there is a big difference between parametric and conventional detailing:

Parametric detailing principle

One parametric recipe for a detail generates specific geometry for all instances of the details in the structure. A good parametric recipe of a detail is able to handle a wide range of variations. The more general the recipe is, the bigger algorithm has to be.

One model – several possible detail instances

Conventional detailing principle

If there are no variations, then one detail applies to all. However, if there are variations within the same detailing principle, one either has to draw several variations or describe only one in detail. The rest depends on the builders to replicate. Since each detail needs a new model in this approach, the time of design and the probability of making a mistake by the designer increase.

One model – one detail instance..

4.1 Detailing Groups

How a detail is crafted is decided by a series of considerations. Architectural, structural and manufacturing considerations are taken into account, but all of the other disciplines included in a building project influences a design. Are the details of the structure load-bearing? Is the detail geometrically planar? Is the connection hinged or fixed? Will the connectors be visible? Do the details need fire-protection? What class of materials will be applied? How will the components be assembled on the construction site? The possible considerations are almost endless.

Figure 9 shows two structures. An experienced human can intuitively point out what is similarly detailed. The challenging part is to precisely describe the logics in a language that a computer understands. The method used in proposed algorithm is surprisingly similar to the children's game called "Guess who?". The purpose of the game is to use as few possible descriptive questions to ask of whom the competitor is thinking. It is not allowed to ask topology-based questions, such as "is it a person in the bottom-left corner?" See Figure 8.



Group1: A man with glasses who does NOT have hair on the top of his head



Group2: A woman with a hat

Figure 8: "Guess who?" Property descriptions filter the different groups

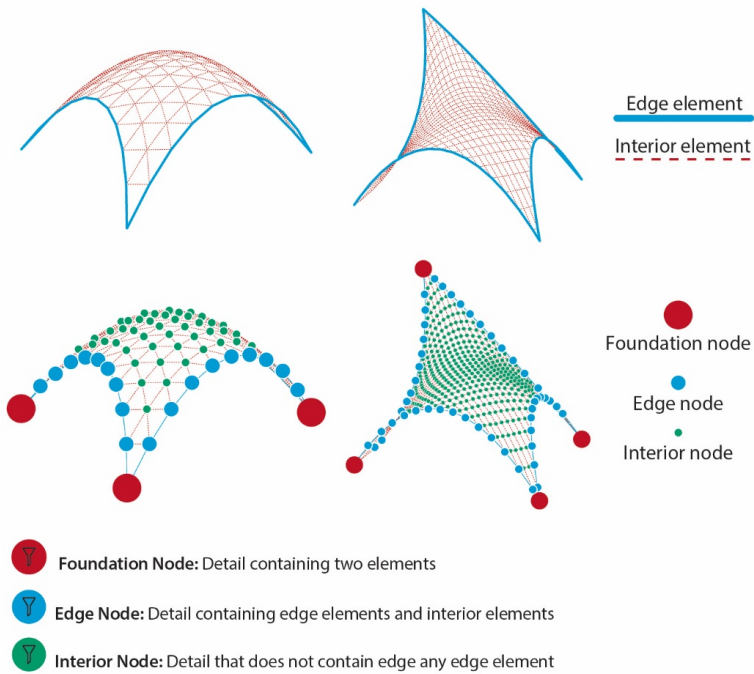


Figure 9: Similar to "Guess who", the detailing groups can be described by property descriptions.

A similar approach is suitable to identify details in a structure. Figure 9 shows two structures. By using property descriptions, the same descriptions can be used on two different structures. In figure 9, all details that contains two elements becomes a foundation node.

The key advantage of the described sorting principle is not to use descriptions relative to the topology and space. Rather, descriptions of local parameters, only influencing each individual detail, are used to identify the details. Hence, a change in the topology does not disturb the sorting system.

The sorting principle is implemented in the toolkit by a series of components regarding property description. Detailing groups are defined by telling whether a property description is true or whether a property description is false. When the detailing groups are defined, each detail is analyzed to determine of which detailing groups they belong. A diagram of the system is described in Figure 10. Table 1 presents a list of property descriptions that have been or will be implemented in the toolkit:

Name	Description	Option
Element length	Defining Min/Max Length	Valid for one or all
Element angle	Max/Min angle between elements	Valid for one or all
Element parameter	Defining Min/Max parameter where the node is connected to the element	Valid for one or all
Element amount	Min/Max amount of elements	
Node position	Checks if node is inside defined bounding box	
Element Names	Checks if Detailing group contains defined names	Including at least or exactly defined elements
Element direction	Defines min/max angle deviation from defined vector	Valid for one or all
Normal Force	Defines min/max Normal Force value	Valid for one or all
Moment Force	Defines min/max Moment Force value	

Table 1: Property descriptions that have been or will be implemented in the toolkit

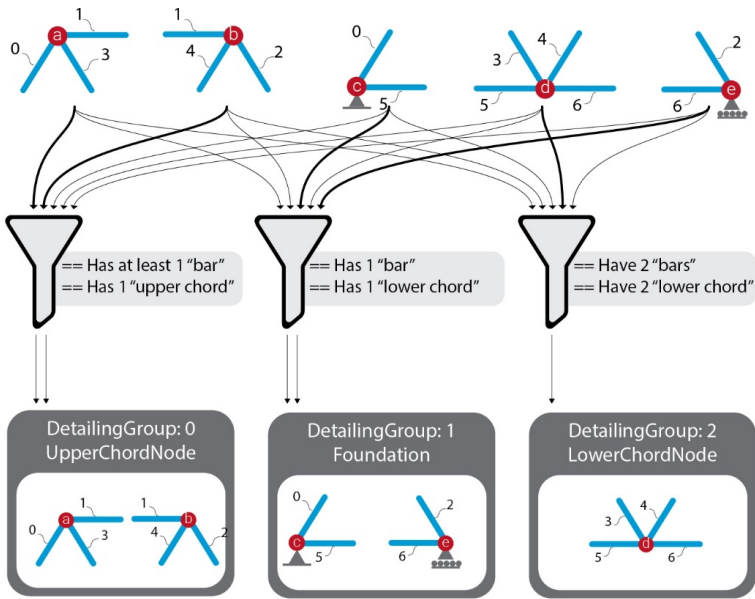


Figure 10: Diagram of the sorting algorithm. The algorithm checks each detail and determines if it fits one or several detailing groups.

The result from the described sorting system is that one may end up with details present in either zero or multiple detailing groups. If that is not the intention of the specific project, then the detailing groups must be further described. However, there are cases in which details connected to zero or multiple detailing groups are relevant. First, being connected to zero detailing groups implies that no detailing is required, the stock does not need any refinement. Second, there might be sub-details that overlap in some details. An example is the chord connections of the Follo Bridge shown in Figure 11. The upper chord and lower chord are primarily similar, but the bottom chord node detail includes a suspended connection to the secondary structure. Hence, the connections in the top chord only belong to the ChordNodeDetail, while the connection in bottom chord also belongs to the SuspensionDetail

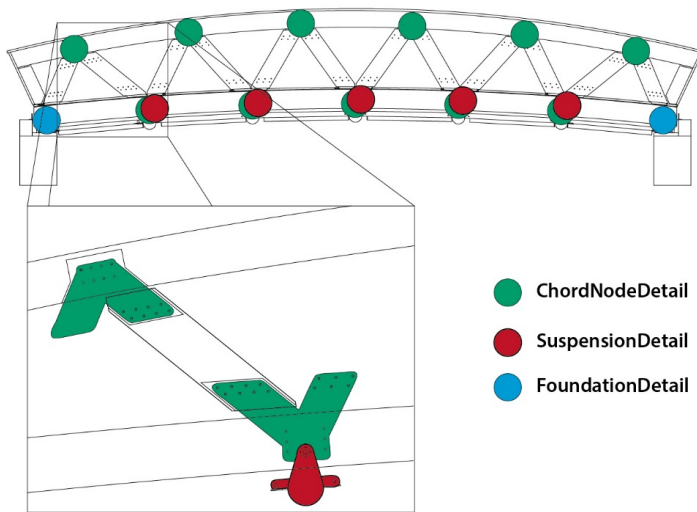


Figure 11: The bottom chord detail belongs to the two detailing groups

4.2 Node Properties

The properties of the elements can be created simultaneously by defining the elements. Assigning properties to the nodes is a slightly more complicated process because the node geometry is a consequence of the intersection of the elements. The features of the node can be changed, added or limited according to changes in the global geometry. The system assumes that all nodes within one node-based detailing group have shared properties. Hence, the desired properties are connected to the property description component.

Generating a sufficient node plane is crucial for efficient parametric detailing. As previously explained, the node is generated from the end points or the element intersections and is by default just a point with no normal direction. However, when detailing a spatial object around this point, the orientation of a local coordinate-system, or a local node plane, must be clearly identified relative to the global coordinate-system of the overall structure. How a node plane is oriented is highly dependent on the type of structure and how the node is detailed, but this orientation is valid for all parametric detailing procedures; a consistent and logical node plane generation is crucial for efficient detailing.

What is a consistent and logic generation of a node plane? The concept can be well-explained in a truss bridge example. By default, a node plane is an XY-plane (Figure 12 A) but is most likely not suitable for the nodes in a bridge. If the bridge is straight and parallel to a global axis, an XZ-plane or a YZ-plane is usable, but it is not flexible. A more consistent rule would be to say that the plane is planar to the truss plane (Figure 12 B). Then, the structure can be oriented in any direction. However, the logics of the planes are still not consistent. The normal direction may be flipped, and the X-alignment may not be defined. By stating that the normal direction of the plane always faces from or to the secondary structure and the X-axis is parallel to the bottom chord, a fully consistent and logical plane-generation is established (Figure 12 C).

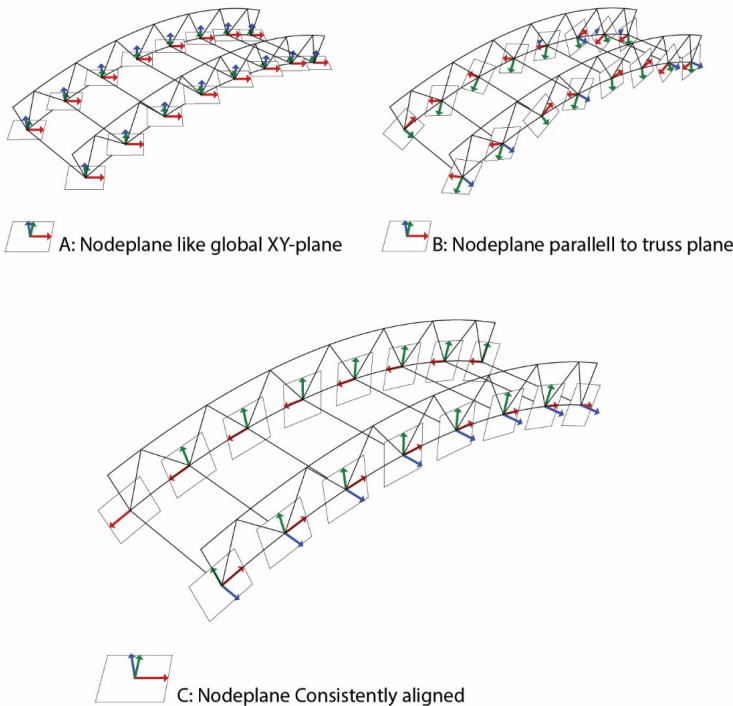


Figure 12: Different strategies of node-plane generation. Figure 9 C shows a consistent and logical plane-generation

These rules apply well for a truss bridge, but the logic does not necessarily apply for all other structures. For example, a gridshell structure is easier to detail if the normal axis of the node plane is parallel to the normal surface. Thus, the toolkit should supply various node plane alignment components.

5. Digital Subtractive tools

There are many purposes of creating a 3D CAD-model. The way a model is made influences the model output possibilities. A sketched volume model is relevant to understand the space to be created but most likely is not adequate for fabrication purposes.

Most of the big manufacturers in the timber industry use CNC tools, such as Hundegger Speedcutter (Hundegger, 2001). These tools have two shared characteristics: they are controlled by numerical data, and they subtract material. These features must be taken into account when designing timber structures digitally. Since manufacturing timber is highly specialized and tool-dependent, a design must often be redrawn by the manufacturer. Redundant work is done, but it also increases the chances of human modeling errors. To solve this challenge, the modeling purposes and geometrical outputs of the architects, structural engineers and manufacturers were investigated. The timber toolkit aims to create an effective solution that fits all three professions. The following is a brief description of these three professions' objective for 3D-detailing timber structures. The description focuses on the part that influences all three professions, namely, the load-bearing system.

5.1 Architect:

While some architects leave the load-bearing system to structural engineers, other architects base the architectural expression on how a structure is made. They use architecture to communicate how the forces in a building work and how a building has been crafted. To design such architectural expressions, it is suitable to make a detailed 3D-model, including the bolts and brackets. Indeed, the architect does not dictate the dimensioning but can hold a great influence on how the structural load-bearing system and detailing are composed. Thus, the architectural geometrical output is primarily visual. Secondary output such as cost, manufacturer, and volume may be generated.

4.2 Structural engineer:

The timber structures due to natural imperfections in the material (orthotropic, not linear grain angle, rods..) are very sensitive to imprecise detailing. Even the best calculations and perfectly chosen static schemes can be ruined because of bad explanations of the joint fabrication/production (the detail).

The issue that is characteristic of timber engineering is finding the connection stiffness. Most of the connections are designed to be rotation-free hinges. Finding the real characteristic of the joint behavior is left either to experimental studies or sophisticated experiments.

Two computational approaches are described—one which is simple and fast, and one which is advanced and more CPU-demanding. The simple detail analysis is made by applying analytical equations (e.g., from Eurocode 5). These equations can check a whole structure in real time. The advanced analysis is made based on sub-modeling. Critical details from global analysis are sent to local analysis. Finite element analysis is performed with volume-objects and more precise material descriptions, and it includes eventual steel connectors such as dowels, bolts and screws.

At first glance, the geometry needed for advanced analysis is very similar to an architectural model. However, the major difference is that how the mesh of the analysis model is built up is critical to make the analysis work.

4.3 Manufacturer:

In regard to digital fabrication, the manufacturer's output is preferably an assembly-ready building component, but the input is a primitive stock. Thus, one of the roles of a high-tech manufacturer is to create instructions for a CNC machine. Two-dimensional cuts performed by lasers or mills are possible to import from a standard CAD-model, but more complex operations have to be defined as virtual timber processing operations.

4.4 Finding a shared detailing solution

The three described methods and geometric outputs look surprisingly similar, but the difference is how the geometry is being made. There are some existing tools that are bridging the gap between the disciplines, but assuming a user-controlled parametric workflow, the options are limited. CadWork and HSBCAD are tools that are tailored for timber structures and connect architecture and manufacturing. Further, Woodpecker, an innovative tool developed by Lignocam and Designtoproduction, enables the export from Grasshoper 3D to any CAM-software. This export occurs through an open-source format called Building Transfer Language.

However, Woodpecker is not a design tool; rather, it is more of a geometry conversion tool (STEHLING, SCHEURER, & ROULIER, 2017).

The solution developed in this toolkit aims to bridge the gap between all mentioned disciplines while modelling parametrically. The concept is to allow the user to refine timber components exclusively based on subtracting material from a digital stock. Thus, the designer is forced into thinking about physical processes while designing virtually. When the architect applies a subtractive operation to a stock, three outputs are automatically generated. A 3D preview geometry, an FEA-ready mesh and a BTL-description. (In the current version of the toolkit, the FEA mesh is not yet implemented). The concept is illustrated in Figure 13.

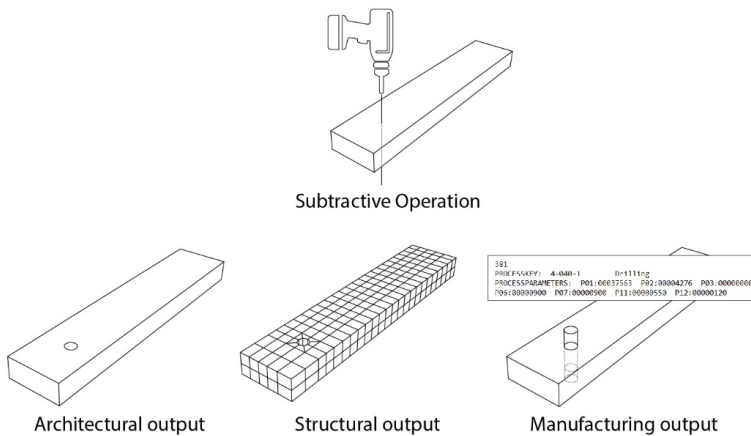


Figure 13: Different outputs from one subtractive operation

5. Case Studies

A series of case projects have been built to test the capability and flexibility of the toolkit. The following sub-chapters shows how the toolkit was used to parametrically design two timber bridges, a freestyle water ramp, a log house and a shelf. The structures are visually and fundamentally different but surprisingly similar when applying the ideas of elements and nodes. The toolkit have been developed while designing the case-studies. Mock-ups of the toolkit have been developed in both Grasshopper-scripts, Iron-python and C#. Table 2 shows what elements of the toolkit have been used designing the structures.

Project	Script Language	Detailing groups	Subtractive tools	FEA	BTL-Export
Orkla Bridges	Iron Python	X	X	X	X
Water Ramp	Iron Python	X	X	X	
Log House	GH-components	X			
Book-shelf	GH-components	X			

Table 2: The table shows which elements of the toolkit have been used to design and build the structures.

5.1 Orkla Bridges

The bridges are designed for a pathway in a park and cross two small rivers. The distance between the bridges are approximately 1 km; hence, it was natural to create two bridges that had a similar architectural expression. However, the boundary conditions at the sites were different. The first bridge, Follo Bridge (shown in Figure 17) had a span of 10 meters and on height-differences but had to be arched to allow the potential ice

drift through. The second bridge, Evjen Bridge (shown in Figure 16) had a span of 15 meters and had a relatively large height difference. Furthermore, the bridge needed curved platforms on both sides to connect to the path and fulfill the requirements of a maximum 1:20 rise.

The different boundary conditions gave a nice opportunity to build a parametric model with a large enough design-space to be adaptable for both bridges. The architectural expression is based on a well-established system of dowels and slotted-in-plates as a connection system (Mork & Luczkowski, 2017). Due to relatively heavy loads, a secondary steel structure had to be used, but the rest of the structure is glulam-based. While the smallest bridge has a classic arched shape, the larger bridge has a more organic appearance with a doubled curved railing.

Using the principle of detailing groups as outlined, only four different detailing groups had to be developed in the small bridge: the bottom chord, the top chord, the foundation points and the zero-force node. Figure 14 shows the rules applied







-  **Top ChordNodeDetail:** Details containing two bars and one top chord
-  **Zero-forcnodeDetail:** Details containing one bar, and one top chord
-  **FoundationDetail:** Details containing one bar, one bottom chord and a secondary beam
-  **BottomchordNodeDetail:** Details containing two bars, one secondary beam and a bottom chord

Figure 14: Detailing groups in Follo Bridge. The same rules were applicable while designing Evjen Bridge

The bottom chord node is the most complex detailing group. The structural bars are connected to the bottom chord, and the secondary structure is suspended from the metal-plates. Due to the scale of the bridge, the suspension connections were integrated inside the chord, making a more compact detail. As previously shown, the basis for the detailing is stocks that are not processed. Subtractive tools are used to design the timber elements. In addition, steel components were designed using conventional parametric tools in the grasshopper environment. In the bottom chord detailing group, these processes were performed (Figure 15):

- Cutting the bars parallel and with an offset half the height + 30 mm from the tangent direction of the bottom chord center curve.
- A grid of holes for the dowels. The amount of dowels was calculated based on the force in the bar/chord, and the size of the grid was calculated based on the required edge distances.
- A pocket milling to make space for the metal-plates. Both the plate that connected the bars to the chord and the plate for the suspended secondary structure.

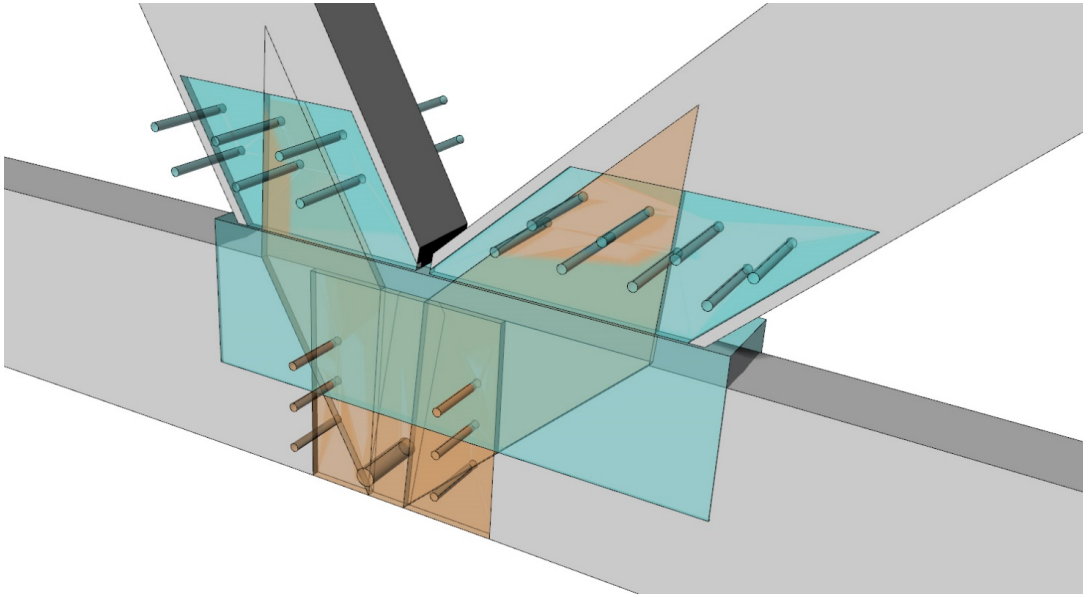


Figure 15: Subtractive operations to achieve wanted detail



Figure 16: Evjen Bridge. Photo: Arnfinn Sæthre



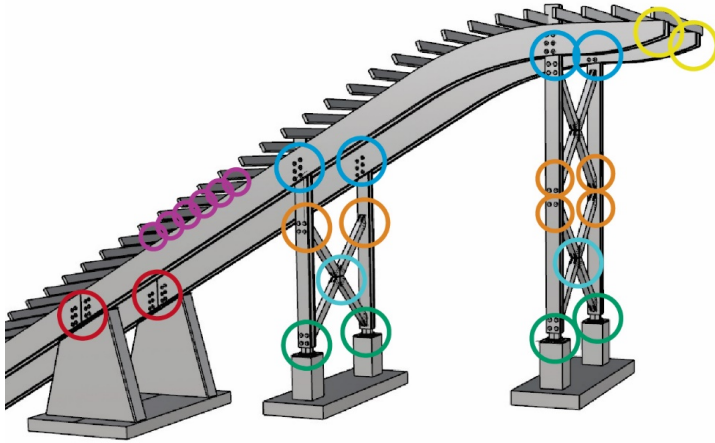
Figure 17: Follo Bridge. Photo: Arnfinn Sæthre



Figure 18: The sauna (mostly L-nodes) on the left side and the water ramp with secondary connection and primary extension nodes.

5.2 Freestyle water ramp

A freestyle water ramp is a ski jump that ends in a lake and is used for summer training. This structure was a simpler design than those of the two bridges. Due to low loads, a beam-column structure was feasible. However, the principle of using dowels and slotted-in metal plates applied in this structure was used as well. Surprisingly, the water ramp had more details to develop than those of a bridge. The detailing groups can be seen in Figure 19, and Figure 18 shows the ramp under construction



- ④ BeamColumnDetail: The detail contains one column and one primary beam
- ④ Secondary Connection: The detail contains one primary beam and one secondary beam
- ④ PrimaryExtensionDetail: The detail contains two primary beams
- ④ Crossing-end: The detail contains a crossing and a column
- ④ Crossing-center: The detail contains four crossings
- ④ Crossing-center: The detail contains only one primary beam
- ④ FoundationDetail: "The detail contains a column and a crossing, and the node is at the end of the column"

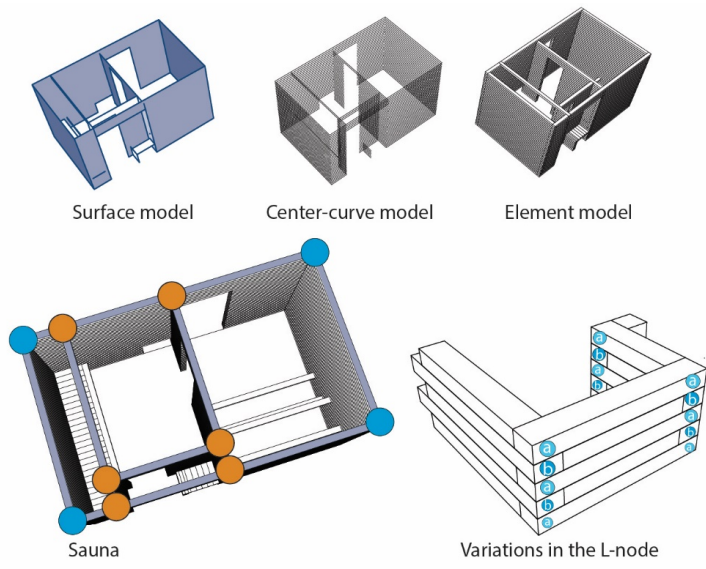
Figure 19: Relatively high amount of detailing groups. Note how the yellow detailing group can be used to trim building components.

5.3 Log house sauna

Until now, the examples have shown rules based on element names. However, there are many cases in which element names cannot be used. Often, the elements are nameless or a set of similar elements. In addition, previous examples have described spacious structures. Is it possible for the principle to apply a more homogeneous structure, containing only one type of element?

An algorithm was developed to be able to model the wall's surfaces. The algorithm then splits the structure into layers with a center-line geometry. What is challenging is to generate the actual length of each individual element. Some elements are extended, and some elements are trimmed to become an overlapping system.

Looking at the system, there are four kinds of connection types, i.e., four types of detailing groups—the X-node, the T-node, the L-node and the I-node. These are easy for a human to determine, but what rules can be applied in the toolkit to distinguish the nodes? An X-detail contains only two laths, but that is the case for the L and T-detail, as well. In this case, information regarding where the node is relative to the lath was used to distinguish the details. Descriptions of the detailing groups are shown in Figure 20 and the final result is shown in Figure 18.



- ✕ **X-Node:** two laths are connected between the start and ending point
- └ **L-Node:** two laths are not parallel and are connected at either the ending or starting point
- | **I-Node:** One lath is connected to the node
- ∖ **E-Node:** Two laths are parallel and connected in a node
- ┌ **T-Node:** One of the laths is connected at the ending or starting point, and the other lath is connected between the starting and the ending points

Figure 20 Detailing groups applying the positioning of the node relative to the element to distinguish the details.

With such rules, all the different types are separated, but there is one more condition needed to generate a log house, namely, the shift in which element is extended. Thus, one more condition to all the detailing groups is if the group is on an even or odd layer. If the T-node is on an odd layer, the element connected to the end will be extended, and the opposite will happen if the node is on an even layer. This principle is shown in the bottom-right of Figure 20.

5.4 Plywood shelf

Finally, the principle has been tested on furniture: a plywood shelf prefabricated using a 3-axis CNC-mill. Both visually and in function, a shelf is something extremely different from a truss, a ski jump and a log-house; there are different scales, different appearances, and different functions. However, algorithmically, a shelf is identical to one layer of the log-house principle. The same type of detailing group applies to the X, T, L and I-node. Figure 21 shows the nodes and Figure 22 shows the detailing of the X-node.



Figure 21: Same property descriptions as the log-house, but different design of the details.

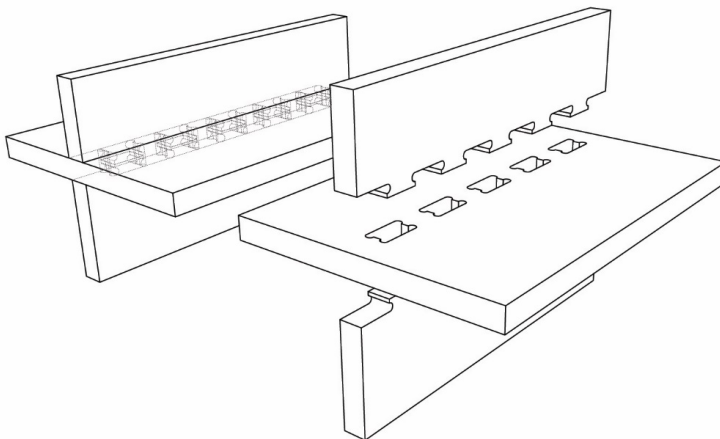


Figure 22: Detailing the X-node. The parts are slotted to simplify assembly

6. Discussion and conclusion

This article presented a flexible approach designing structures parametrically. The aim was to investigate how to apply the benefits of MC while designing site-specific, customer-inclusive and bespoke timber structures. The main finding is that when abstracting any structure to elements and nodes, most of the structures can be designed through a similar set of rules.

Architects are trained to design unique projects. Reusing elements from a completed project almost feels like cheating. Such mindset stands in contrast to the idea of mass customization. However, learning from the described study and case-projects, one conclusion is particularly prominent: To be able to achieve Mass Customization in architecture, we must standardize projects. Thanks to parametric modelling, we do not have to standardize the physical result. Rather, we can standardize the workflow, the process and the interfaces between the project stakeholders. This paper has exemplified a standardized, still flexible, building system in the age of parametric architecture.

A structure can be similar in a parametric description, but can vary much in function, form, scale and all other attributes. What appears as visually and radically different structures from an architectural point of view is very similar from an algorithmic point of view. The log-house and the shelf exemplify this issue very well. In contrast, two structures that are functionally similar are fundamentally different from an algorithmic perspective. With such a view, a truss bridge is more similar to a free-form gridshell than a beam bridge. A tendency in described case-projects, is that one parametric workflow outputs a much larger design space than in ordinary, manually drawn CAD-projects. By parametric thinking, we are able to shift from standardized products to standardized, still scalable, parametric building systems.

The toolkit made in fully parametric design environment allows the designer (architect or engineer) to observe and control the design in a holistic way. From digital conceptual draft to the production codes, every important decision can be programmed, followed in real time and changed according to the need of hour. The toolkit does not introduce such methodology, but simplifies and substantiates a continuous digital workflow from global shape to digital fabrication and assembly. This by offering a series of tools that reduces technical complexity of parametrically designing timber structures. Especially the detailing groups based on property descriptions and the digital subtractive tools have drastically reduced the time needed to set up a parametric model of a project.

According to R.Aish [9] in computation design: "The designer is no longer directly modelling the building; instead he develops a graph or script whose execution generates the model". The timber toolkit presented here allows the designer to go out from strict BIM design, where everything have to be described "a priori", to described by R. Aish definition called computation design. The variety of presented cases done with one approach and one toolkit is a conclusive evidence of applicability of it in building design and industry.

One can definitely state that the Orkla Bridge project fulfils the idea of mass customization. The first bridge, Follo Bridge were extremely time-demanding, but designing the Evjen Bridge was much more efficient. Updating the shape of the bridge required only 60-90 seconds, including a simple FEA, regenerating all details and not the least outputting BTL-instruction. However, these bridges were relatively similar. How about the other structures? The already-designed bridges contributed to a more efficient detailing of the water ramp. The slotted-in metal plates for the detailing algorithm were reused when modeling the foundations for the columns. They are different shapes but based on the same knowledge-based engineering (KBE)-rules.

The flexibility of a building system is often dependent of the hours spent developing the system. A flexible and highly reusable parametric building system is lightly not to pay off if thinking one project a head. However, if developing the building system is considered as an investment in a series of future projects, we strongly believe that such mindset is economically sustainable. Further, to achieve Mass customization in architecture, we must continue shifting our mindset to better understand the possibilities of applying parametric tools and computation.

At last, does the timber toolkit ensure site-specific, customer-inclusive and bespoke timber structures? The author's answer is, luckily, that the toolkit does not make great architecture. At best, the toolkit reduces the amount of work spent on redundant work. However, the hours earned can be spent making better architecture. Architecture is still made by architects; tools are just tools simplifying the creative process.

References

- Aish, R. (2013). First build your tools. *Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design*, 36–49.
- Banks, J. (2016). Why BIM is still bankrupting your firm. *Shoegnome* 9.12.
- Burns, W. E. (1986). Computer-Aided Design (CAD). *Industrial Education*, 75(6), 10–11.
- Daaboul, J., Cunha, C. D., Bernard, A., & Laroche, F. (2011). Design for mass customization: Product variety vs. process variety. *CIRP Annals*, 60(1), 169–174. <https://doi.org/https://doi.org/10.1016/j.cirp.2011.03.093>
- Davis, S. M. (1997). *Future perfect*. Basic Books.
- Goldberg, A., & Robson, D. (1983). *Smalltalk-80: the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc.
- Hundegger, H. (2001). Woodworking machine.
- Kaplan, A. M., & Haenlein, M. (2006). Toward a parsimonious definition of traditional and electronic mass customization. *Journal of Product Innovation Management*, 23(2), 168–182.
- Krymsky, Y. (2015, December 11). The Architecture Software Revolution: From One Size Fits All to DIY. Retrieved January 18, 2017, from <http://www.archdaily.com/778619/the-architecture-software-revolution-from-one-size-fits-all-to-diy>
- Kymmell, W. (2007). *Building Information Modeling: Planning and Managing Construction Projects with 4D CAD and Simulations (McGraw-Hill Construction Series): Planning and Managing Construction Projects with 4D CAD and Simulations*. McGraw Hill Professional.
- McNeel, R. (2009). Rhinoceros 3D. Retrieved Jan, 15.

McNeel, R. (2015). *Grasshopper 3D*.

Mork, J. H., & Luczkowski, M. (2017). One algorithm, two timber bridges built in Orkdal, Norway. In *Forum Wood Building/Nordic Trondheim 17*. Faculty of Architecture and Design Trondheim.

Preisinger, C., & Heimrath, M. (2014). Karamba—A Toolkit for Parametric Structural Design. *Structural Engineering International*, 24(2), 217–221. <https://doi.org/10.2749/101686614X13830790993483>

Rapp, E. J., & Johnson, G. A. (1980). *Computer numerical control machine tool*. Google Patents. Retrieved from <https://www.google.com/patents/US4199814>

Scheurer, F. (2013). Digital craftsmanship: from thinking to modeling to building. *Digital Workflows in Architecture*. Birkhäuser, Basel, 110–129.

Schumacher, P. (2008). *Parametricism as Style-Parametricist Manifesto*.

Schumacher, Patrik. (2016). Parametricism 2.0: Gearing Up to Impact the Global Built Environment. *Architectural Design*, 86(2), 10. <https://doi.org/10.1002/ad.2018>

STEHLING, H., SCHEURER, F., & ROULIER, J. (2017). BRIDGING THE GAP FROM CAD TO CAM: CONCEPTS, CAVEATS AND A NEW GRASSHOPPER PLUG-IN. In *Fabricate 2014: Negotiating Design & Making* (DGO-Digital original, pp. 52–59). UCL Press. Retrieved from <http://www.jstor.org/stable/j.ctt1tp3c5w.10>

JointSearch:

Efficient parametric detailing preparation through user-defined and property-based joint type filtering

Authors:

John Haddal Mork

Marcin Luczkowski

NB: The following article is an update version compared to the version sent to the reviewers. After the review process, this article was accepted to be published in International Journal of Architectural Computation. The journal reviewers recommended minor changes to the demonstrator and conclusion chapter. Hence, the following article is a print of the online, published article.



JointSearch: Efficient parametric detailing preparation through user-defined and property-based joint type filtering

International Journal of
Architectural Computing
1–15

© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1478077120943176
journals.sagepub.com/home/jac



John Haddal Mork¹  and Marcin Luczkowski²

Abstract

Detailing joints are important when designing structures. In this design process, a structure is divided into different joint types. Digital fabrication and algorithmic aided design have changed the conceptions and requirements of joint detailing. However, parametric tools that can efficiently identify joint types based on the solution space are not available. This article presents a methodology that efficiently generates topological relations and enables the user to assign joint instances to joint types. A series of property-based search criteria components is applied to define the solution space of a joint type. Valid joints are coherently filtered, deconstructed and outputted for detailing. The article explains both the methodology and programming-related aspects of the joint type filtering. The article concludes that the developed methodology offers the desired flexibility and may be suitable for other materials and applications.

Keywords

Digital planning, parametric detailing, algorithmic aided design, geometric relations, topology

Introduction

An important aspect of designing structures is to detail the joints – the intersection of building elements. The detailing of a joint is dependent on a series of constraints, including architectural, structural and manufacturing constraints.¹

Digital fabrication and algorithmic aided design (AAD) have evolved and will continue to evolve the building industry (BI): Digital fabrication and computer numerical control (CNC) machines expand the

¹Faculty of Architecture and Fine Art, Norwegian University of Science and Technology, Trondheim, Norway

²Faculty of Engineering, Norwegian University of Science and Technology, Trondheim, Norway

Corresponding author:

John Haddal Mork, Faculty of Architecture and Fine Art, Norwegian University of Science and Technology, Alfred Getz vei 3, 7491, Trondheim, Norway.

Email: john.h.mork@ntnu.no

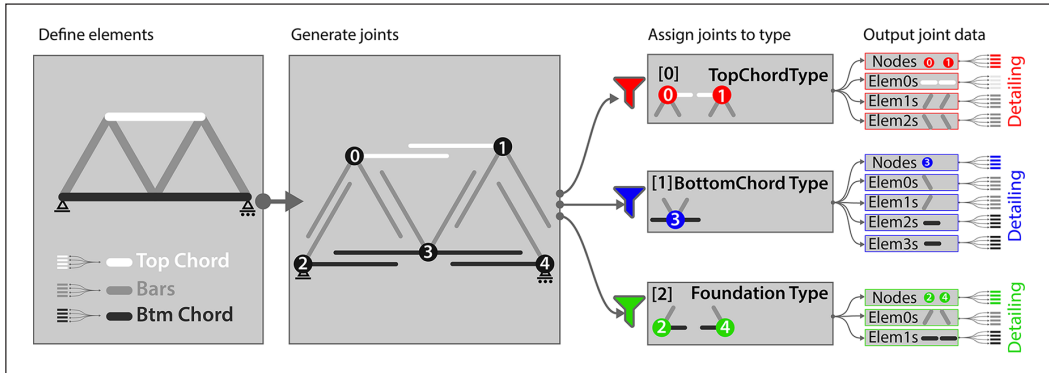


Figure 1. Four steps for preparing a parametric detailing.

domains of rational manufacturing. Not dependent on jigs and manual labour, CNC machines are universal² and can produce variations at no extra cost.³ This mode of production is referred to as nonstandard seriality⁴ and substantiates a mass-customised BI.⁵ Similarly, AAD is crucial for expanding what is rational to digitally design. The digital aspect of nonstandard seriality is achieved by building algorithms that generate geometry based on custom inputs.

In parallel, visual programming software, such as Grasshopper3D⁶ and Dynamo Studio,⁷ has democratised algorithmic modelling and enables designers with limited programming knowledge to utilise the power of both parametric modelling and digital fabrication. Arguably, digital tools have changed the conceptions and requirements of a detail.⁸

- All variations of a fabricated building element need to be accurately modelled when applying digital fabrication: A digitally fabricated structure requires a digital twin. This requirement contradicts notational drawings that are aimed at manual labour, where principal details are sufficient. Although a structure consists of thousands of unique joints, the amount of principally different joint types is likely to be manageable. La Seine Musicale, which is a timber gridshell, had 2798 unique joints but only 8 main joint types.^{9,10} By making an algorithm-aided parametric system for each joint type, the types' instances can be automatically generated. The size of a joint type's possible solution space is dependent on the developer. Most joint type's parametric system will allow some degree of geometric flexibility.

Regardless of the AAD, the logic that separates a set of joints into joint types varies from project to project. The previously mentioned joint types in La Seine Musicale were constrained by the beam types connected in the joint (two diagonals, a diagonal and a horizontal beam (referred to as ring)).⁹ Similarly, a beam that is connected to a column is likely to be detailed differently than a secondary beam that is connected to a primary beam. Thus, these two cases belong to two joint types. In other cases, other geometric parameters are determinant.

An important task of preparing a parametric detailing is to define a logic that defines elements, generates all preliminary joint relations, assigns all joints to their joint type parametric system and outputs a joint's geometric data stream in the most possible coherent manner (refer to Figure 1). The purpose of this task is to prepare sufficient input data for each joint type's parametric detailing system.

CITA has developed solutions for creating and storing information to generate joints.^{11,12} Similarly, but more universal, Wassim Jabi and Robert Aish are developing Topologic, which is a kit for non-manifold

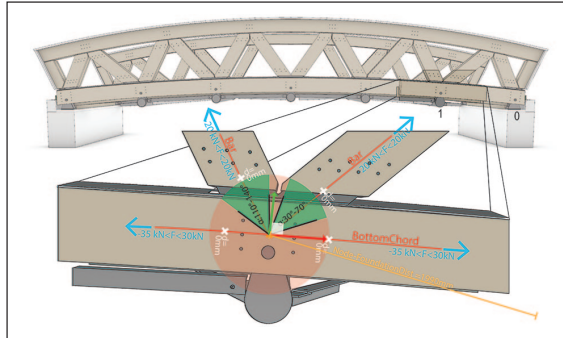


Figure 2. The timber truss bridge and the BottomChord joint type's solution space.

topologies (NTMs).¹³ This toolkit is open-source and allows the user to build topologic relations that combines vertices, lines, wires, faces, shells and cells.

If a structure is logically clear, a small sorting algorithm may be used to filter the joint types. However, the algorithm is likely to be dependent on a fixed topology. Complex structures require other methods, and in the case of La Seine Musicale, a spreadsheet interface was employed to assign the correct type to each joint.⁹ A different strategy is to write a custom algorithm that filters joints based on defined properties – Buro Happold identified similar connections in the Morpheus Hotel exoskeleton by applying this strategy.¹⁴ Rhino.Inside for Revit enables the user to filter geometry using predefined geometric rules.¹⁵ Front Inc. has developed a toolkit named Elefront, which enables the user to filter geometry based on custom key-values that are assigned to a geometry.¹⁶

As introduced above, methods for assigning joints to joint types exist. However, these methods of identifying joints are dependent on a fixed and distinct global topology or require advanced parametric thinking. The result is a fragile parametric model or a time-demanding process of preparing a model for detailing.

In all types of detailing, properties and detailing-specific constraints define a joint type's solution space. Thus, a tool that enables the user to intuitively transfer these properties and constraints to an algorithm is needed.

This article presents a methodology that efficiently generates topological relations and enables the user to assign joint instances to joint types. A series of property-based search criteria components is used to define the solution space of a joint type. Valid joints are coherently filtered, deconstructed and outputted and customised for detailing. The result is a universal method that turns joint type identification independent from its global topology.

The tools are prototyped in a Grasshopper toolkit named Reindeer. The toolkit is under development and customised for detailing fabrication-ready timber structures. Other articles explain the overall framework⁵ and timber-specific tools.¹⁷

This article explains the methodology and programming-related aspects of the developed tools. Brief examples and a built case structure demonstrate and evaluate the tool. A brief conclusion is provided.

Proposed methodology

As most projects invent a new joint type variation, the number of possible joint types is almost infinite. Therefore, predefining all joint type solution spaces is not possible. Rather, the designer should be free to define and update a joint type's solution space while designing. This chapter proposes such a methodology. To explain the principle, a BottomChord joint type in a timber truss bridge is employed. See Figure 2.

Any joint type has an intended purpose, such as structurally connecting two bars, a chord and a secondary beam. Thus, only valid joints must be detailed according to their joint type.

A well parametrically detailed joint enables some flexibility in its solution space⁹ and is likely to have some threshold values that cause the joint to belong to another joint type. In this context, a solution space is an arbitrary space that capsules all possible variations of a joint type. If a joint instance is inside the joint type's solution space, it belongs to this joint type. The BottomChord joint type has the following property criteria that define the solution space. Note that the following property criteria correspond to both the joint and its elements.

Contains (a) BottomChord, (b) bar and (c) bar

The BottomChord is continuous through the node (can be subdivided for structural analysis)

The angle between the BottomChord and the bars are between 30 and 70 or between 110 and 140 degrees

The normal force of the BottomChord is between -35 and 30 kN

The normal force of the bars is between -20 and 20 kN

The joint is planar

The joint is not near a foundation

If a joint fails one of the property criteria, it does not belong to the BottomChord joint type. For example, if the BottomChord is not continuous, the joint is located at the end of the bridge. In this case, these joints must be defined as another joint type.¹⁸ Some properties allow a domain of allowed values, while other properties are Boolean. In the example, the combination of elements is fixed, while the angle between the bars and the chord allows a variation. In theory, all thinkable parameters are definable in a joint type. But if no joint instances are close to exceeding the threshold value, there is no practical function for defining it. In the example, the length of each bar is a redundant solution space parameter to define. A joint type's solution space can be visualised using parallel coordinates. Figure 3 shows one property for each column and the size of the allowed domain. The figure compares Joint0 and Joint1 against the BottomChord joint solution space. Joint1 is valid, and Joint0 is invalid.

The outcome of defining a joint type solution space in this way is dual. Primarily, correct joints are assigned to correct joint types. However, a second application can be equally relevant. If assigned joints are visualised in real time, the designer can adjust the geometry until all joints belong to a detailed joint type. In this way, the method can also be applied for optimisation purposes.

This example showed how a BottomChord joint type's solution space was defined. However, this method should work for all types of joints, regardless of material or function. To generalise the method, the solution space is modularised. Each module equals a property, and each property should have an adjustable domain. In this way, the user can easily add the properties that are relevant for defining a specific solution space. In some cases, one property is sufficient; for other joints, multiple properties are needed. Figure 4 shows how two sets of property criteria can filter two bridges and three shells.

This methodology is likely to be applicable for NTMs¹³ and can be used to filter any kind of joint. A joint can be point based, edge based or even face based. The only requirements are that the defining solution space must cover both the object and its neighbours and the structure's elements must have sufficient property information for evaluation. By example, a joint type search can be performed in a cross-laminated timber structure. If the plates are idealised as faces, and the faces are attributed with properties such as name and width, the same solution space method can be employed. The methodology is expected to work universally, regardless of industry, material or topology.

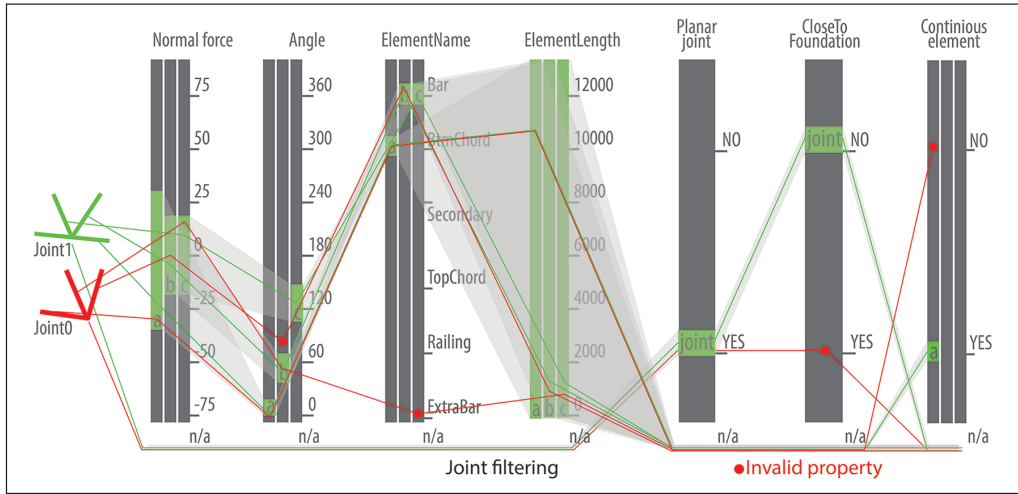


Figure 3. BottomChord joint type’s solution space visualised in parallel coordinates. Joint0 and Joint1 tested if belong to BottomChord joint type. Joint0 fails the test.

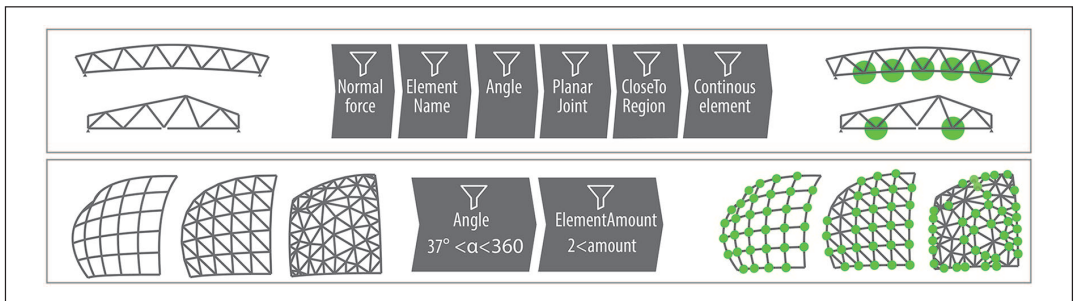


Figure 4. Modularisation of property criteria. Two bridges and three shells are analysed to find two types of joints.

Implementation

To test the proposed methodology, a prototype is developed and implemented as a plugin for Grasshopper3D. To limit the scope, one-dimensional (1D) timber elements and seven property criteria were determined to be sufficient for implementation.

The following workflow is adapted to timber structures and are the steps used when applying Reindeer. Timber-specific content, such as material, cross-sections, detailing principles and manufacturing technology, can be placeholders for other materials.

0. Generate global model – A global model is defined using (centre) curves. Step 0 is the main input to Reindeer
1. Define elements – The user creates timber elements based on curves from step 0. These elements are customised by defining the element name, cross-section(s), local alignment, global alignment and material properties.

2. Assemble: Generate joints – Generates nodes and joints based on the structure's elements.
3. Structural analysis of the structure – Optionally, a structural analysis may be run in this step.
4. (a) Joint search and (b) output joint data – The user defines property-based search criteria, filters the joints and outputs data required for detailing. A user perceives searching and outputting as one step; from a programming perspective, it is two steps.
5. Detailing the elements using TimberProcessingTools – Using Reindeer, timber elements are solely detailed using tools that mimic physical timber processing. Reindeer 0.5 includes cutting, drilling, pocketing and tenon/mortise.¹⁷
6. Processing elements – In this step, processing instructions from step 5 are used to output a detailed preview and a BTLx-file. The detailed preview is non-uniform rational basis spline (NURBS) geometry.¹⁷

Step 4a is a prototype of the described methodology. To ensure that this step works and has a practical purpose, steps 1–3 and 4b are equally important. These four steps are chronologically presented in the next section.

Defining elements

Elements are modularly defined by multiple components. Material, cross-section, local alignment, composite and global alignment components enable the user to customise an element. Highly detailed figures, related to this article, can be found online at Zonedo.¹⁹ For definition of elements, see Mork and Luczkowski¹⁹ (p. 1).

Defining joints

In Reindeer, a joint consists of a node and its 1D element(s).¹⁷ Nodes are not specified by the user, but a result of elements that intersect.⁹ From the structural analysis perspective, elements can only intersect at the ends of a 1D element. From a detailing perspective, however, one element can meet another element in the middle.⁵ The bridge example illustrates this concept. Thus, two events generate a node:

The starting point or endpoint of an element

Elements intersect within the domains of both elements.

The steps of defining elements and assembling the structure have organised a data model that is built for both search and detailing. The joint contains elements and nodes, and the elements and nodes contain geometric data. The following section explains joint generation and Reindeer's class structure.

Joint generation algorithm. Allowing elements to intersect anywhere, including the endpoints, increases the complexity of the programming strategy: The algorithm is required to check if elements intersect somewhere on the curve's domain. To make the algorithm work, a two-step process was developed. The first step is conventional. When an element is added to the assembly, the code checks if the endpoints exist as nodes. If the node exists, the element is added to its joint. Otherwise, the endpoint is applied to create a node, and the node and element form the basis for creating a joint. The second step takes into account the probability of intersection. All elements must be checked to determine if they intersect with any other elements in the structure. This procedure is simple if the structure is small, but the calculation time exponentially increases when additional elements are added to the assembly.

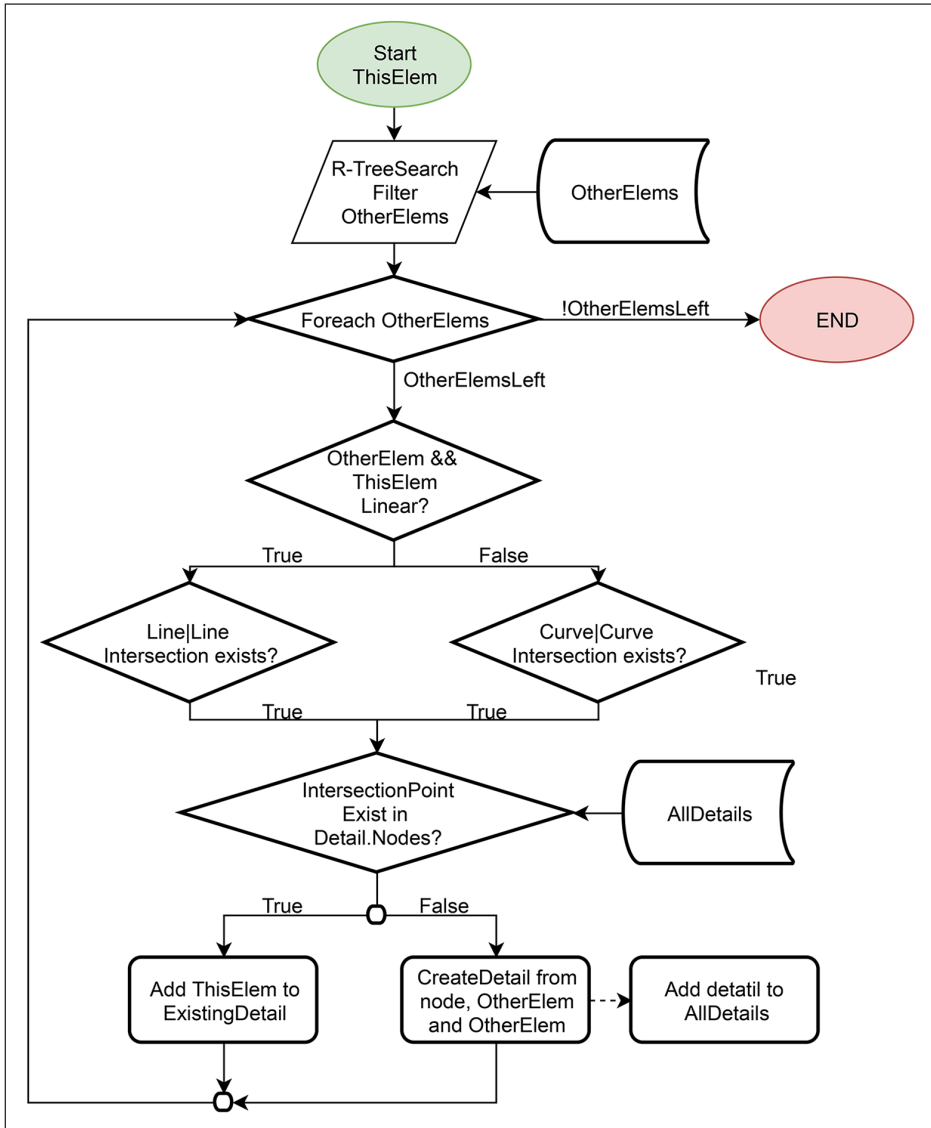


Figure 5. Finding intersection nodes.

Figure 5 explains the developed strategy. When an element is added to the assembly and tested for an intersection, the code starts by using an R-tree search, which is a spatial search structure, to filter elements that plausibly intersect. The R-tree stores the element’s bounding sphere in a tree structure and turns searches more targeted towards plausible hits. Visually speaking, an intersection is plausible if the bounding spheres of two elements intersect.

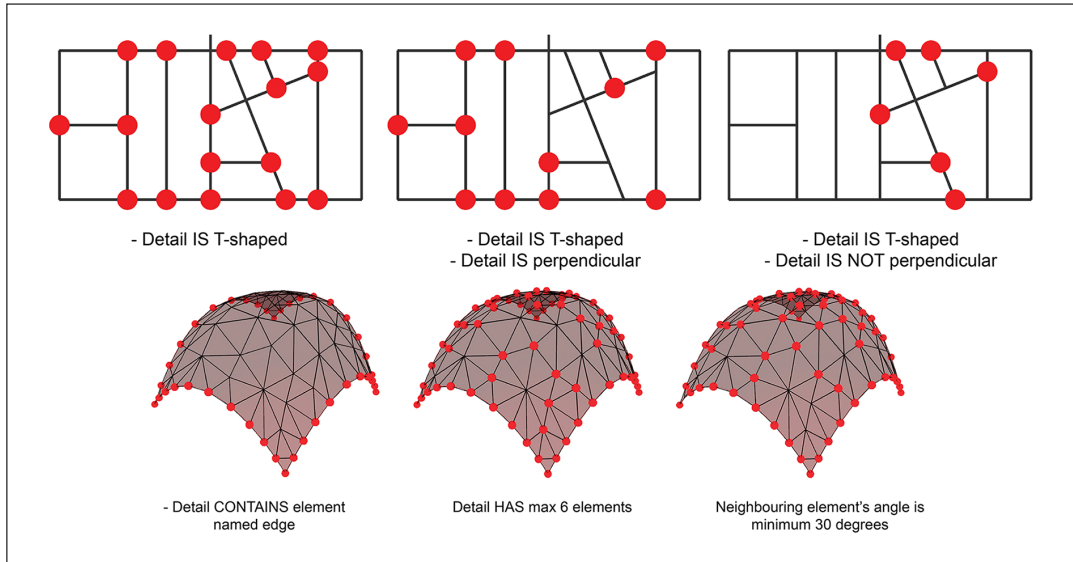


Figure 6. JointSearch criteria work, regardless of shape or surface pattern.

Plausible other elements (OtherElems) are checked if they intersect the new element (ThisElem). The algorithm checks whether the intersected point already exists as a node. If the node exists, the ThisElem is added to the joint (detail). Else, a new node is created, and the node, ThisElem and OtherElem are applied to generate a joint. The joint is then added to the list of AllDetails, and the loop may be repeated.

Class structure. The class structure consists of a main assembly class. The assembly class contains a node class, element class and detail class (see Mork and Luczkowski,¹⁹ p. 2). A detail and a joint are synonyms in this article: to synchronise the vocabulary with other related researchers, the word joint is used as the relation between elements and nodes. Initially, the word detail was employed and is still applied in the code and the schemes that explain the code.

Identifying joints

The developed solutions let the user define the joint types' solution space by using one or multiple search criteria components. Conceptually, one component equals the columns in the parallel coordinate diagram. These search criteria filter all joints and pass only the joints that fulfill all search criteria. Mork and Luczkowski¹⁹ (p. 3) shows two trusses and illustrates how joints are identified using Reindeer.

As the approach does not apply sorting rules but solely depends on the filtering based on the joint's geometric properties, a parametric detail becomes algorithmically independent from the global topology. If the global topology changes during a design process (What is a sign of a healthy design process?), valid joint instances will be identified if they fulfill the solution space of the joint type.

The power of the search method is revealed when detailing structures without repetitive logic. Figure 6 illustrates a shelf and shell. The shelf shows three different detail types, where the first detail type only requires the detail to be T-shaped. The two other detail types additionally require the angle between the elements to be perpendicular or not perpendicular. A practical example related to this separation is that the

Table 1. Search criteria components.

Name	Inputs	Checks
Element length	Min length Max length	Checks lengths of elements and returns true if all elements are within the allowed domain
Element angle	Min angle Max angle	Checks angle between each pair of elements and returns true if pairs are within the allowed domain
Element amount	Min amount Max amount	Checks the number of elements and returns true if the amount is within the allowed domain
Element name	Element names Mode	Checks if the detail's element's names correspond to the names of the element. The mode input determines the strictness of the checker. Mode0: The detail's element's names must contain one of the inputted names Mode1: The detail's element's names must contain at least all inputted names Mode2: The detail's element's names must contain all inputted names and no other names Mode3: The detail's element's names must correspond exactly to the inputted names
Detail topology	Mode	By choosing a mode, the checker finds details that correspond to typical shape types. The following modes are included: Mode0: L-node: Two elements connected in an endpoint Mode1: T-node: One element is connected at the end, and one element is connected in the middle of the element (not at the ends) Mode2: X-node: Two elements connected on the elements (not at the ends) Mode3: End-node: Nodes with a single element connected Mode4: Star-node: Three or more elements connected at the end Mode5: Planar: All elements can be placed on a single plane Mode6: Orthogonal: All angles between elements in detail are 0, 90 or 180 degrees
Node in region	Region Max offset distance	Checks if the point of the node is inside an inputted region. The inputted max offset distance determines if the point must be on the region or can be offset in the normal direction of the region
Element force components	Max force Min force	Checks if all elements are within the force domain specified by the user. Individual components for Torsion MX, Bending MY, Bending MZ, Compression FX, Tension FX, Shear FY, and Shear FZ.

perpendicular details can be sawed using a three-axis machine but the angled detail is likely to require a more advanced manufacturing technique.

The figure shows the same shell that is shown in Figure 4. On the left side, the name of the element is used to find the details that contain an edge. The two other joint searches distinguish the complexity of the joint. Reducing a joint's maximum number of elements or defining the minimum angle between the elements is likely to reduce the complexity of the node. Table 1 presents the implemented search criteria components.

As shown in Figure 7, each search criterion has one component. The JointSearch is the component that performs the filtering. The criterion is defined and initiated in the SearchCriterion component, sent to the JointSearch component, added to the search criteria list and is performed. The programming strategy is presented by, first, explaining the definition and initialisation of a search criterion, and second, how the criteria are filtering the joints.

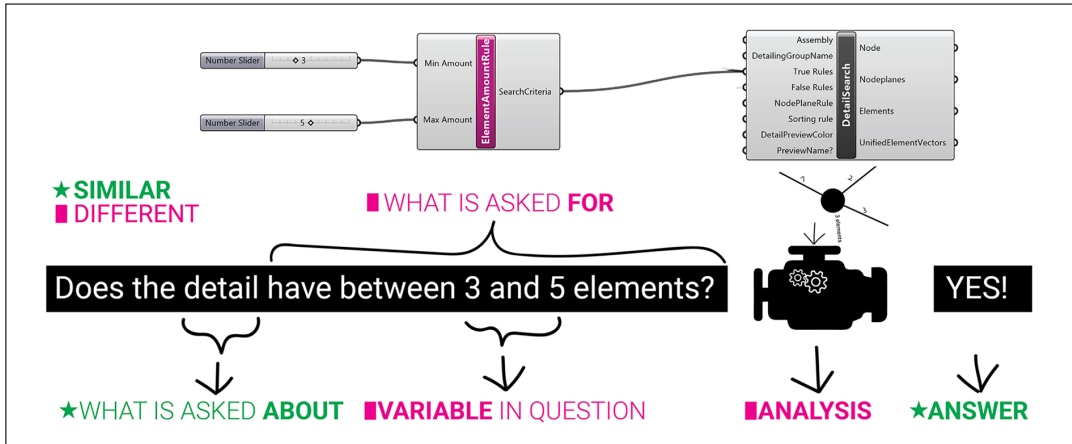


Figure 7. Elements of a search criterion. The top half of the figure shows an ElementAmount criterion.

Initiating search criteria. To be scalable, all search criteria follow a strict format that provides sufficient flexibility. The following points define the format (refer to the bottom of Figure 7):

Questions can only be asked about the joint

The question that is asked is free – the answer must be true or false

The question allows user-defined variables

The method returns a true or false value, and the only input parameter is the joint. As the method is inside the class, all fields are accessible. Figure 8 shows the ElementAmount class and illustrates how the question from Figure 7 is turned into an algorithm. The method checks the number of elements in the joint. If *elements.Count* is smaller or equal to *minAmount* AND *elements.Count* is smaller or equal to *maxAmount*, then the method returns a true. Otherwise, the method returns a false. The strictness of the search criteria is attributed to the notion that the method is not initiated immediately but passed to the JointSearch component. The SearchCriteria component does not have information regarding which joint to analyse. Thus, a delegate method is defined with a bool return and a detail as the parameter input. This makes it possible to store methods in a list and run them when appropriate.

When a search criterion object has been initialised, the delegate method is wrapped in a general Rule object and outputted from the component. The Rule object standardises the search criterion and ensures that the input data to the JointSearch is coherent. As a result, new developers can easily code a new search criterion if they construct a method that adheres to the specified delegate format and wraps the constructed method inside a rule object.

In addition to inputting the assembly, the JointSearch component inputs True Criterion and False Criterion. Both inputs are classified as the Rule object type but are stored in lists named *ValidProperties* and *InvalidProperties*. If a rule in the *ValidProperties* list is to be valid, it must return a true. Oppositely, if a rule in the *InvalidProperties* list is to be valid, it must return a false.

Figure 9 shows the function that filters the search's valid joints. Each joint is tested against the rules in the *ValidProperties* list and *InvalidProperties* list. However, if a rule reveals the joint to be invalid (red field), the loop is broken. Only if the joint passes all rule tests, it is added to *ApprovedDetails* list (green field).

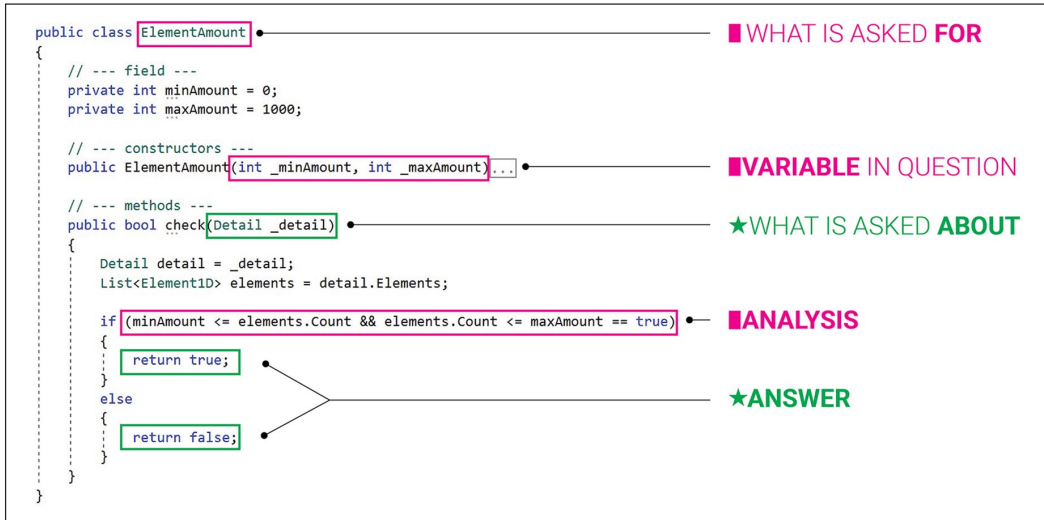


Figure 8. Back-end: class and method. Each rule has a unique class containing field, constructor and method. The fields contain the variables in the question; the constructor initialises the variables to the object, and the method is the logic that answers the question.

Outputting joints

When detailing a joint, information about the related nodes and elements is required. The more consistently streamed data, the easier the detailing process becomes. An inconsistency often demands codes that handle exceptions from a distinct logic. Three functionalities are implemented to increase the consistency of the data stream: (1) Node plane generators, (2) Unified element vectors and (3) Element sorting rules.

Node plane generators. Defining a node plane is essential in most detailing cases. If a node plane is consistently generated according to its detail's purpose, the detailing process becomes easier to manage. However, the logic of a node plane varies from case to case. Currently, the Reindeer plugin provides five different methods for generating the node plane. The methods offer various ways for defining the Z- and X-axis of the plane, and the node point determines the location of the plane. Similar to the search criteria component, delegate methods enable the node plane component to grant access to data from the detail.

Unified element vectors. An element has a fixed starting point and endpoint. Thus, a detail will consist of a mix of elements that have their starting point or endpoint at the node's location. For detailing purposes, utilising vectors that are parallel to their elements but consistently directed from or towards the node is convenient. Thus, the JointSearch component automatically generates and outputs unified element vectors.

Element sorting rules. The order of the elements in detail is initially unstructured. To increase the consistency, the first, second, third and nth element in each detail instance must correspond. However, a sufficient ordering logic is highly case specific. The current version enables the user to sort (1) clockwise relatively to the node plane, (2) based on structural priority or (3) alphabetically based on element names. Figure 10 exemplifies the importance of these sorting functionalities. The first detail is part of a gridshell structure. Thus, sorting the output of the elements clockwise is probably convenient. The second detail consists of a column,

```

public DetailingGroup GenerateDetailingGroup(List<Detail> _details)
{
    List<Detail> ApprovedDetails = new List<Detail>();
    List<Plane> NodeGroupPlanes = new List<Plane>();

    foreach (Detail detail in _details)
    {
        bool ValidDetail = true;

        foreach (CheckGroupDelegate TrueProp in ValidProperties)
        {
            if (!TrueProp(detail)) //Testing for false. If false, the detail does not contain in the group
            {
                ValidDetail = false;
                break;
            }
        }

        foreach(CheckGroupDelegate FalsePrope in InValidProperties)
        {
            if (FalsePrope(detail)) //Testing for true. If true, the detail does not contain in the group
            {
                ValidDetail = false;
                break;
            }
        }

        if (ValidDetail)
        {
            ApprovedDetails.Add(detail);
            NodeGroupPlanes.Add(PlaneRule.NodeGroupDelegate(detail));
        }
    }
}

```

Figure 9. The method that filters the joints.

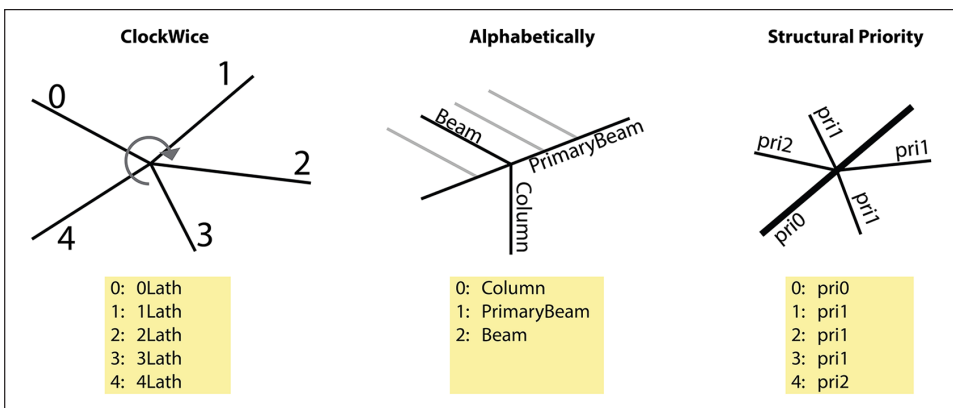


Figure 10. Three different sorting rules for outputting the elements.

primary beam and secondary beam. In this case, alphabetically sorting the elements will be consistent. The third detail is part of a system with a primary structure, secondary structure and tertiary structure. In this case, sorting based on structural priority is convenient.

Example: outputting data for detailing. The hierarchical construction of the joint makes it easy and logic to deconstruct the joint. The JointSearch component outputs nodes, node planes, elements and unified element vectors as a grafted list (one branch for each joint). The order of the elements and unified element vectors is selected according to the applied sorting rule. Further, the element and node are deconstructed into its properties. Finally, the element can be deconstructed into its subelements. What is required data is individual from detailing to detailing. However, the node plane, width and height, base curve, cross-section plane and side planes of the element are frequently required in the design processes. Mork and Luczkowski¹⁹ (p. 4) illustrate the data outputted from a truss' foundation joint.

Example: node in complex shell. A shell, which is shown by Mork and Luczkowski¹⁹ (p. 5), consists of interior elements and edge elements. The scheme shows how the JointSearch identifies three- or four-legged interior details. Only two criterion components are required: a criterion that defines the element names that are not allowed (edge) and a criterion that specifies the number of elements. The left detail in the figure in Mork and Luczkowski¹⁹ (p. 5) shows how the JointSearch outputs geometry by default. However, the default joint plane is not applicable, and the order of the outputted elements is not consistent. The right side of the figure is a better basis for detailing. First, the detail plane component has generated a plane based on the shell surface. The X-axis of the plane is aligned parallel to one of the interior elements. Second, the elements are sorted clockwise according to the detailing plane. Note that the element0 is parallel to the X-axis.

Demonstrators

The BottomChord joint type used as example in this article is implemented in two bridges that were designed and built using an early version of Reindeer. These bridges, a ski jump and a log-house, are documented in other articles.^{5,20} In the case of the bridges, the joint types were defined using only element name combinations. The joints were filtered based on their properties, and the algorithm was designed to be easily adaptable to both bridges.

Figure 11 shows a parametric timber frame wall (Mork and Luczkowski,¹⁹ p. 6, shows the related Grasshopper definition). Such structure is not complex from a geometrical point of view. However, from an algorithmic and topological point of view, the structure is more advanced. The example is part of an online shed configurator. When a user orders a shed based on individual inputs, the algorithm automatically outputs pre-cut instructions readable by a CNC machine. With such large design space, it is not feasible to develop a universal sorting algorithm to identify the joints. The example illustrates how joint types are identified in a series of walls with various openings. In this example, a combination of 'Element Name', 'Detail topology' and 'Element Amount' criterion components is used to identify eight joint types. In some sheds, all joint types are needed – in simpler sheds, only a few joint types are needed. These joint types are detailed according to the house manufacturer specifications and outputted as BVX/BTLx-files. The purple joints are identified by specifying what element names to be included or not included in the joint, and that the joint shall be T-shaped and orthogonal.

The contrast of the discussed and referred demonstrators indicates that the introduced methodology, implemented in the Reindeer prototype, is applicable for designing a wide range of structures.

Conclusion

This study developed a methodology that efficiently generates topological relations and enables the user to assign joint instances to joint types. The developed prototype, which is based on four steps – defining

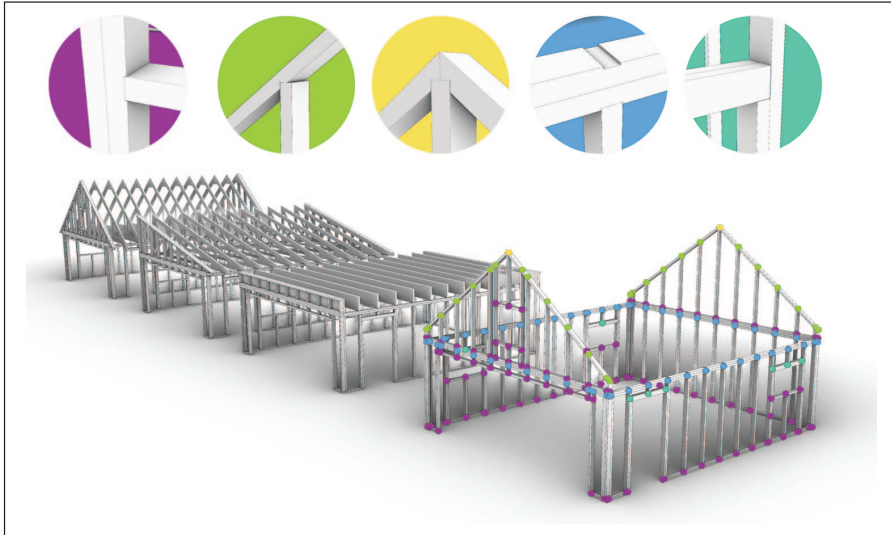


Figure 11. Different shed configurations, but a fixed set of possible joint types.

elements, generating joints, searching for joints and outputting joints – has proven to be flexible despite the simple scheme required to prepare a model for detailing. Other referred tools, such as Elefront, allow efficient geometry filtering. However, these tools lack designated methods for multi-objective filtering of geometric relations. Hence, the presented property-based joint-search methodology is seen as the study's main contribution.

The notion that the parametric detail becomes algorithmically independent of the global topology is an important advantage. This approach simplifies the process of filtering joint types and substantiates reusing/adapting definitions of parametric joints from projects to projects.

The bridge structures illustrate that the method is applicable for advanced, but logically sound, structures. Further, the timber frame wall demonstrates that the methodology is also applicable for structures that have a less clear topological concept.

The results of this study indicate that the joint-search method is a universal method that is applicable for distinguishing joint types in an arbitrary structure but is dependent on a properly established model that contains topological relations. If a structure is built up by 1D elements and nodes, prototyped software is applicable.

To turn Reindeer more relevant, a future software development must also include cross-laminated timber and other plate-based timber materials. Hence, new joint types and search criteria are also required. Introduced Topologic offers a more comprehensive set of elements in its topology, including wires, faces, shells and cells. Implementing such elements will allow a much larger range of structures. Hence, further research will investigate the potential of combining Topologic's topology library and Reindeer's joint-search capabilities.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship and/or publication of this article.

ORCID iD

John Haddal Mork  <https://orcid.org/0000-0003-3256-7370>

References

1. Mork JH, Luczkowski M, Dyvik S, et al. *A parametric toolkit for advanced timber structures*. In: *6th forum wood building Nordic Trondheim 17*. Trondheim: Forum Wood Nordic, 2017
2. Scheurer F. Digital craftsmanship: from thinking to modeling to building. In: Marble S (ed.) *Digital workflows in architecture*. Basel: Birkhäuser, 2012, pp. 110–131.
3. Carpo M. *The second digital turn: design beyond intelligence*. Massachusetts, UK: MIT Press, 2017.
4. Carpo M. *The alphabet and the algorithm*. London: MIT Press, 2011.
5. Mork JH, Luczkowski M, Manum B, et al. Toward mass customized architecture. Applying principles of mass customization while designing site-specific, customer-inclusive and bespoke timber structures In Bianconi F and Filippucci M (eds) *Digital wood design: innovative techniques of representation in architectural design*. Cham: Springer International Publishing., 2019, pp. 221–249.
6. McNeel R. Grasshopper3D, www.grasshopper3d.com (2020, accessed 11 January 2020).
7. Dynamo Studio. Computational BIM design, *Autodesk*, www.autodesk.com/products/dynamo-studio/overview (2019, accessed 3 July 2019).
8. Celani G. Myths in architectural detailing that are changing in the digital age. *ArchDaily*, <https://www.archdaily.com/886741/7-myths-in-architectural-detailing-that-are-changing-in-the-digital-age> (2018, 12 June 2019).
9. Usai S and Stehling H. La Seine musicale. In: De Rycke K, Gengnagel C, Baverel O, et al. (eds) *Humanizing digital reality: design modelling symposium Paris 2017*. Singapore: Springer Singapore, 2018, pp. 201–209.
10. Stehling H, Scheurer F, Roulier J, et al. From lamination to assembly: modelling the seine musicale. In: Menges A, Sheil B, Glynn R, et al. (eds) *Fabricate 2017*. London, UK: UCL Press, 2017, pp. 258–263.
11. Poinet P, Nicholas P, Tamke M, et al. Multi-scalar modelling for free-form timber structures. In: *Proceedings of the IASS annual symposia, IASS 2016 Tokyo symposium: spatial structures in the 21st century – timber structures & environmental compatibility* (eds K Kawaguchi, M Ohsaki and T Takeuchi), Tokyo, Japan, 26–30 September 2016, IASS.
12. Svilans T, Tamke M, Thomsen MR, et al. New workflows for digital timber. In: Bianconi F and Filippucci M (eds) *Digital wood design: innovative techniques of representation in architectural design*. Cham: Springer International Publishing, 2019, pp. 93–134.
13. Aish R, Jabi W, Lannon S, et al. Topologic: tools to explore architectural topology. In: *Proceedings of the advances in architectural geometry 2018*. Gothenburg, 22–25 September 2018, Chalmers University of Technology.
14. Piermarini E, Nuttall H, May R, et al. Morpheus hotel, Macau – a paradigm shift in computational engineering. *Steel Constr* 2018; 11: 218–231.
15. Rhinoceros. Rhino.Inside, <https://www.rhino3d.com/inside> (2019, accessed 15 October 2019).
16. Van der Heijden R, Levelle E and Riese M. Parametric building information generation for design and construction. *Comput Ecol* 2015: 416–429.
17. Mork JH. *Parametric timber toolkit: a timber tailored approach. Again, goldern era of timber*. Seoul, South Korea: WCTE, 2018.
18. Bell K, Grindland O, Gundersen G, et al. *Kollapsen av Perkolo bru – hva gikk galt?* Oslo: Statens Vegvesen, 2016.
19. Mork JH and Luczkowski M. 2020 Additional images, <http://doi.org/10.5281/zenodo.3921987> (2020, accessed 29 June).
20. Mork JH. A parametrized process: design and realization of timber truss bridges. In: *Proceedings of the ICTB 2017 – 3rd international conference on timber bridges* Skellefteå, 26–27 June 2017, ICTB.

PARAMETRIC TIMBER TOOLKIT: A TIMBER TAILORED APPROACH

John Haddal Mork¹, Marcin Luczkowski², Bendik Manum¹, Anders Rønnquist²

ABSTRACT: Timber is not a straightforward building material. Manufacturing is mainly based on subtractive operations, while most 3D-software invites to an additive design approach. Furthermore translating a design into CNC-instructions may be time-consuming and labour intensive. This paper mainly presents an essential component of a parametric timber toolkit under development. The toolkit aims to simplify the process of designing advanced timber structures. The tools presented in this paper, is the subtractive digital tool package. Tools that let the designer develop a timber based design solely on subtracting material from a parametrically defined stock. The papers also presents a case-project, two timber truss bridges recently designed and built using the timber toolkit. The paper sums up by discussing the implications of the tool.

KEYWORDS: Parametric timber, Glulam, Subtractive operations, Timber truss bridge

INTRODUCTION

Timber is not a straightforward building material. It is not isotropic, and the structural properties varies a lot. Among species, but also internally in one stock. Furthermore, it is mainly processed using subtractive tools[1]. Mills, sanding machines, saw-blades and drills are conventional tools that subtract material from the stock and turns it into a component. Even though Computer Numerical Control (CNC)-machines and robots have revolutionized the timber industry, the same tools are being used. Thus, we have to change our mindset from instructing a carpenter to instructing a machine. CNC-machines does not read conventional blueprints.

It is often said that CNC-machines are able to produce thousand unique components in the same speed as thousand copies. However, if drawing thousand components manually, the potential of fast manufacturing is lost. This is where parametric algorithms comes into play.

Digital parametric workflows, which widely appeared a decade ago, represent a dramatically new way of applying digital power while designing our built environment. Instead of drawing geometry, parametric workflows let the user define the design as a series of decisions, systems and relationships[2]. Hence, the digital design process has the potential of becoming

more flexible, than existing streamlined BIM processes. Robert Aish is describing it as the computation era and argues that the objectives of design computation is to overcome many of the limitations of BIM[3]. As the authors see it, the computational approach and parametric thinking is only way to fully utilize CNC-machines.

Design tools will always influence the result of a design. If designing with clay, you are likely to end up with something organic. If designing with boxes, you are likely to end up with a stacked structure. If so, digital tools will also influence the result.

There are some design software tailored for timber structures, e.g. Cadwork[4] and HSB CAD[5]. Few, such as Woodpecker[6] are also developed for parametric software. However, the majority of software are built up on an idea similar to additive design. Typically either by defining geometric primitives or perform operations on curves. E.g. loft, revolves, sweeps or extrusion. Great tools if materialization is performed by casting, 3D-printing or even 2D-cutting. However, if digitally designing timber structures, one must know a bit of manufacturing processes, and the result is often that the manufacturer have to remodel the design for fabrication[1].

This paper mainly discuss and presents an essential component of a parametric timber toolkit under development. The toolkit aims to simplify the process of designing advanced timber structures. The components presented in this paper, is the subtractive digital tool package that unifies design and manufacturing. Tools that let the designer develop a timber based design solely on subtracting material from a parametrically defined stock. In the background, and real-time, a file for Computer Aided Manufacturing[7] (CAM) software are being generated. The paper also describes two timber bridges recently built, and were designed using the

¹ John Haddal Mork, NTNU, Norway, john.h.mork@ntnu.no

² Marcin Luczkowski, NTNU, Norway, marcin.luczowski@ntnu.no

¹ Bendik Manum, NTNU, Norway, bendik.manum@ntnu.no

² Anders Rønnquist, NTNU, Norway, anders.ronnquist@ntnu.no

parametric toolkit. The paper summarize the implications of using such tool and further development.

2 DIGITAL TIMBER

2.1 BUILDING WITH TIMBER

Tacit knowledge is embedded in a carpenter. Knowledge that are in the hands, knowledge that are intuitive and almost hard to explain to others. The authors are not skilled carpenters, but have built a couple of projects either in a workshop or on site. While being in a workshop and building with wood, design constraints and opportunities seems obvious. Your mind focuses on how an object is made. If you need a hole in a beam, you start with a beam and then assess what is the most rational way of achieving a good result. If the hole is small and the shape is circular, a drilling tool will be sufficient. If a large and rectangular hole were to be processed, a milling procedure would be better to consider, but then remembering that a fillet would be left in the corners. Figure 1 is an example where the milling radius is used as input for architectural expression.



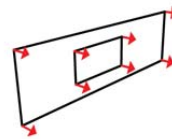
Figure 1: Example of using the milling radius as input for architectural expression

Nevertheless, these are all subtractive methods of refining a stock to a crafted building component. Now, why is such facts being described? The authors strongly believe in a timber architecture driven by carpentry craftsmanship. However, the authors also believe that parametric tools and CNC-machines will be an architect's everyday tools. Thus, bridging the gap between carpentry craftsmanship and digital design is essential.

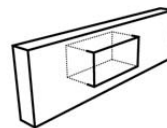
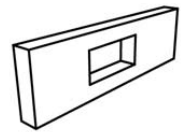
2.2 DESIGNING DIGITALLY

When opening most 3D software tools, one starts with an empty canvas. Points, lines, curves, surfaces and solids are digital geometry used to build up a virtual model. As in the workshop, a skilled designer learn how to design using the premises of the tools. If a rectangular hole in a glulam beam were to be drawn, there would be a series of methods of achieving wanted result. Following is two conventional methods (illustrated in figure 2):

- a) Drawing 2D-lines of the hole and the contour of the beam, and then extrude the lines to a solid.
- b) Making a box representing the beam and a smaller box representing the hole. Then the volume from the smaller box is subtracted from the larger box.



A) Extruded rectangle and circle



B) Box and smaller box

Figure 2: Two methods of modelling a beam with a hole

The second method is more similar to physical making, but still two problems occurs:

- 1) The edges in the hole is not filleted. A problem easy to solve by filleting the edge, but also a problem easy to forget that exists while designing digitally.
- 2) The output from both operations looks like a hole in a beam, but the computer only reads it as boundary representation (B-rep)[8] of surfaces. For a designer, this is perfectly fine, but for a manufacturer and a CAM software, this geometrical information is not directly readable as a hole in a beam.

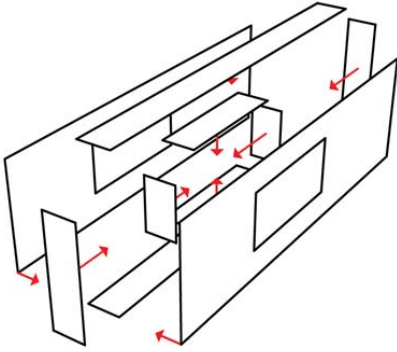


Figure 3: The computer stores the model as a B-rep

A simple case like a hole in a beam can though be solved through feature recognition, meaning that the CAM software analyses the B-rep and understand that a drilling or milling operation is sufficient to achieve wanted shape. However, this is not possible when it comes to shapes that are more complex[9]. Further, method A is logic in a digital environment, but extends the distance to physical making of timber structures, and may lead to designs that are not buildable.

2.3 BUILDING TRANSFER LANGUAGE

Building transfer language (BTL) is an open standard developed by SEMA and CadWork and provide a parametric description of the geometry of wooden building components[10]. The format enables data exchange of timber structures and describes building components as parts (beams, columns, bars laths etc) with processings (drilling, pocketing, dowetails, contours etc). Additionally, the format contains logistic data and timber specific data.

Logistic data is information like package number, module number, designation and annotations, while timber specific data includes grain direction, timber specie and quality grade.

Stehling argues that since BTL is deliberately not machine-specific, BTL makes a good match for the building industry. A firm specific CAM-processor is still needed to generate CNC-files[9].

Stehling and Desigtoproduction have also developed mentioned Woodpecker, a plugin for Grasshopper3D that outputs BTL-files. However, and as they present it, it is more an export tool rather than a design tool.

3 THE TOOLKIT

Fabian Scheuer describes an architect’s creative process as following: “architects very rarely work out a design on site, on a one-to-one scale with their hands dirty from manipulating an actual building material” [11] This is the reality for most architects, and thus digital tools must enhance architects to understand (to a certain level) physical manufacturing constraints while designing digitally.

The toolkit under development aims to bridge the gap from design to manufacturing of advanced timber structures. Not only solve export interfaces between CAD and CAM, but shift the designer to think physically while designing digitally. To solve this, the toolkit includes a series of subtractive design tools. More about these tools in the next chapter.

The toolkit has a series of components that substantiates a continuous automatically updating dataflow from overall geometry, structural analysis, detailing to BTL-export. This means that if the designer changes the shape, a new detailed model shall be ready for BTL-export. The toolkit will not be explained thoroughly in this article, but following is a few definitions and a brief description important to understand the concept.

3.1 ELEMENTS AND NODES

The toolkit generates a virtual timber structure based on center-curve geometry. The user then specifies one or multiple element types by assigning names, cross-section, rules for cross-section alignment and not least material properties. The toolkit focuses on parametric detailing, and a detail is hereby defined as either a node and its elements or an element and its nodes. By defining a series of property description rules, the user can split all details into different custom detailing groups. As for the case study described in the end of this paper, such detailing groups were the upper chord node, the lower chord node, the foundation points and the zero-force nodes. Figure 4 illustrates how the details are sorted.

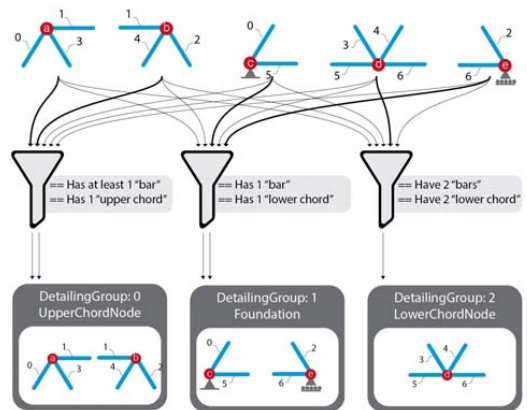


Figure 4: The details are sorted to detailing groups using property descriptions.

3.2 THE TOOLKIT'S DEFINITION OF AN ELEMENT

What is considered as an element is different from architects, structural engineers and manufacturers. While most architect’s conception of an element is a building component, the structural engineer understands an element as a straight line with two end nodes. Further, the manufacturer may split a building component into multiple sub-elements.

Since the toolkit is built with the help of object oriented modelling, an element is defined as an object. The master element is the building component, but additionally, the element object contains a structural sub-elements and manufacturer sub-elements. Figure 5 illustrates the concept.

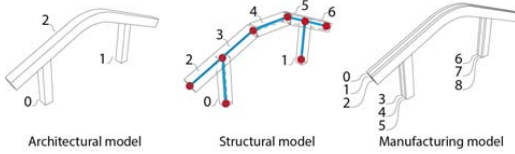


Figure 5: Conceptions of an element

4 SUBSTRUCTIVE TOOLS

The subtractive tools are based on the elements described in previous chapter, and is essential to refine the details. Figure 6 shows the lower chord of the case study bridges before any detailing was performed. The model works as a conceptual model, to understand the shape and interaction with the site.

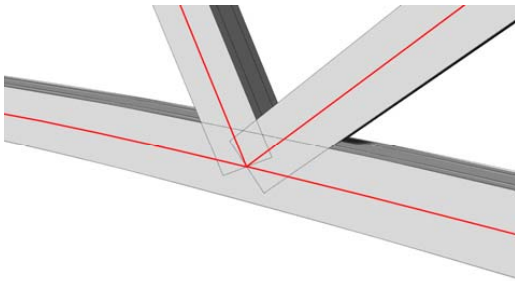


Figure 6: Elements that are not yet detailed. Centerline-geometry is projected on the surface

To refine the details, virtual carpentry tools are used. Drilling, pocket milling, and cutting are the tools that are currently implemented, but the BTL-format offers a series of other processings.

As an example, the input needed for the drilling component is an axis (a line) and a radius. The line are generated in wanted location with help of either global coordinates or local node or element planes.

When a drilling component is added, the algorithm stores two items in the element object:

- 1) The parameters needed to write a BTL processing, local coordinates, local drilling angles and radius
- 2) A Cylinder used to subtract the hole.

The reason why the hole is not subtracted (drilled) inside the drilling component is bifolded:

- 1) Subtractive operations (also called Boolean operations) are CPU-demanding, and many processings will make the software perform badly.
- 2) Most likely, there are more than one processing applied to one element.

For these reasons, the void geometries are stored in the element object. However, this does not mean that geometry can not be previewed. By adding a preview component, the subtractive operations to all elements are executed and a detailed geometry is generated. A detailed version of lower chord can be seen in figure 7.

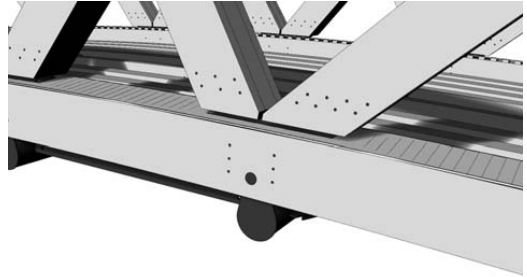


Figure 7: Detailed lower chord

When the user is ready, a BTL-file is also ready to be exported. Further, the authors sees an opportunity to include a third output: A mesh ready for finite element analysis (FEA), but this is not currently implemented.

Figure 8 shows the idea of one operation outputting three types of geometry.

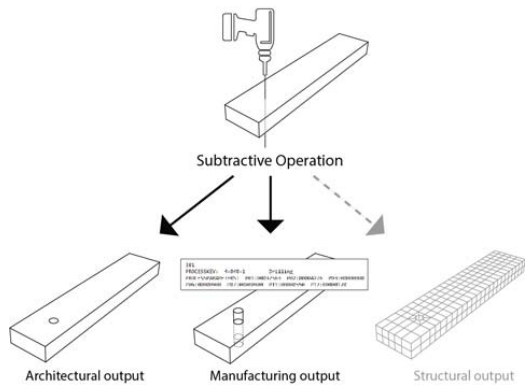


Figure 8: Digital subtractive operations generates a B-rep and manufacturing output (BTL). Future work will investigate the potential of outputting mesh for FiniteElementAnalysis

4.1 Future implementation

As described, the BTL-export is not machine-specific. This makes the export more accessible, but it also have some limitations. Future versions of the toolkit will support input for machine specific and not least firm-specific data. In current toolkit, it possible to digitally design building components that are not buildable. At least not buildable by the chosen manufacturer. If including machine and manufacturer-specific data, the designer can be warned if exceeding the constraints. Following are examples of relevant data important for design decisions:

- Available stock-sizes
- Maximum dimensions to be CNC-mill
- Available milling tools
- Maximum/minimum drilling/cutting angles

5 CASE PROJECT

To validate the relevance of toolkit and the subtractive tools, two timber truss bridges were designed and built. Both bridges were completely parametrized and based on the same parametric model. All slots, drillings, cuts and not least splitting of the glulam were modelled with the help of the toolkit. Additionally, Metal components were parametrically modelled, but prepared for fabrication with a conventional method

Both bridges are mainly for pedestrian purposes. However, the local municipality demanded dimensioning for a 130 kN wheel-loader with an additional 50 kN of load. These are additional similarities:

- Timber truss bridges
- Slotted-in-plates and dowels as connections
- Steel-based secondary structure, pin connected internally in the bottom chord
- Width of 3400 mm (Center-center, trusses)

What distinguishes the bridges are the length, amount of bars, the shape of the bottom/top chord, the rise and the detailing of the endings. Due to height-differences, and demanded path of the road, the biggest bridge has curved platforms on both sides. The platform railing both challenges the geometrical flexibility of the algorithm and gives the bridge a bespoke architectural character

5.1 DETAILING USING SUBTRACTIVE TOOLS

The bottom chord node was the most complex detailing group. The structural bars are connected to the bottom chord and the secondary structure is suspended from the metal-plates. Due to the scale of the bridge, the suspension connections were integrated inside the chord, making a more compact detail. As previously explained, the basis for the detailing is stocks that are not processed. Subtractive tools are used to design the timber elements. In addition, steel components were designed using conventional parametric tools in the grasshopper environment. In the bottom chord detailing group, these processes were performed:

- Cutting the bars, parallel and offset half the height + 30 mm from the tangent direction of the bottom chord center curve.
- A drilling a grid of holes for the dowels. The amount of dowels were calculated based on the force in the bar/chord and size of the grid were calculated based on required edge distances.
- A pocket milling to make space for the metal-plates. Both the plate that connected the bars to the chord and the plate for the suspended secondary structure.

The processings can be seen in figure 9 and the physical result can be seen in figure 10:

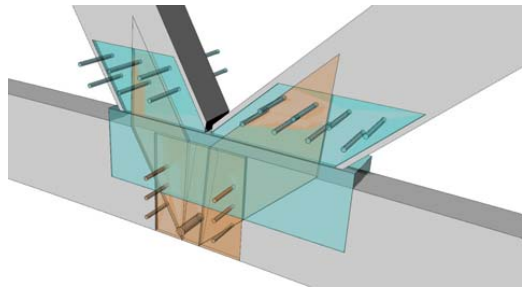


Figure 9: Processings to detail the lower chord (Showing the centric sub-element)



Figure 10: Physical result. The lower chord

5.2 FOLLO BRIDGE

Due to the simple shape and boundary conditions, Follo Bridge was the first bridge to be built. The shape is classically arched and has a span of 10 meters. Follo bridge is shown in figure 11 and 12.



Figure 11: Follo Bridge at night



Figure 12: Follo Bridge

5.3 EVJEN BRIDGE

Universal design with a maximum rise of 1:20, determined the required length of the bridge (including the platform). While the deck is almost 30 meters, the span is only 16.5 meters. The structural part of the bridge is linear in plan-view, but due to relatively large loads, the top chord is arched. The curve of the platform-railings correspond with the top chord and visually merges the structures. Evjen bridge is shown in figure 13 and 14.



Figure 13: Evjen bridge-. Structural bridge in the center, Curved platform on both sides.



Figure 14: Evjen bridge at night.

6 DISCUSSION

As mentioned, designers are influenced by its design tools. The authors strongly believe that the building industry will increasingly be using parametric tools, and therefore better timber tailored tools must be developed.

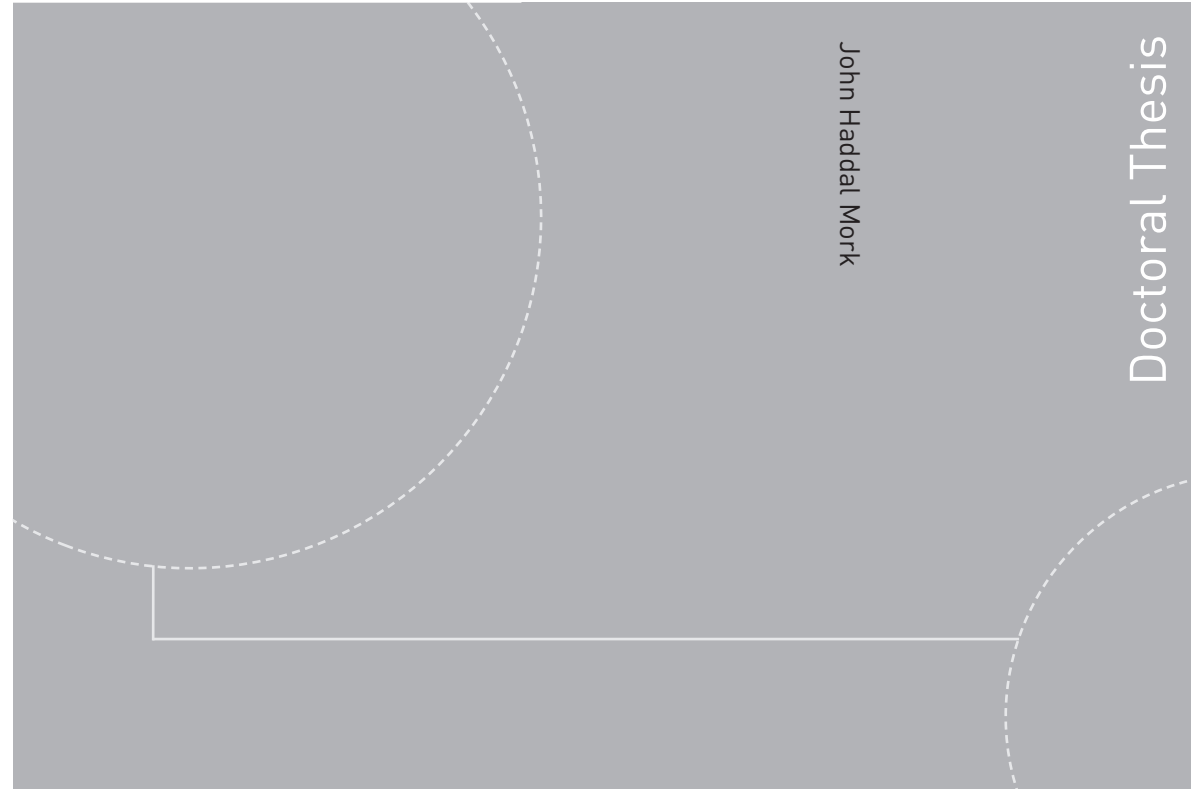
Such parametric toolkit does not replace the value of getting hands dirty. Making objects using physical tools and physical objects will always give a deeper understanding. However, by mimicking the physical process, such tools may at least make the designer remember that the digital design are to be materialized.

Standard or non-standard? As Scheuer beautifully explains: "...But standards does not only make life easier, they also make life simpler"[9]. One of the beauty of parametric modelling is the flexibility. You are not constrained to standards, and you make your own tool while designing. Are the timber toolkit developed constraining the designer? Somehow, the answer is yes. The concept of the toolkit is to push manufacturing constraints to the conceptual phase. In that way, the chances of designing unbuildable structures are reduced.

Described toolkit is still under development, and are not written by computer scientists, but experiences from the case study suggests that such tools are intuitive and makes the design particularly trivial to send to production. While designing, the subtractive tools available limited the design space and led to a rational manufacturing process. Further, the described case projects (except the platform railing) are rather conventional structures, thus more geometrically advanced structures will be developed to further test the toolkit.

- [1] Schwinn T. Advancing Wood Architecture: A Computational Approach, 2016, p. 190–1.
- [2] The Architecture Software Revolution: From One Size Fits All to DIY. ArchDaily 2015. <http://www.archdaily.com/778619/the-architecture-software-revolution-from-one-size-fits-all-to-diy> (accessed January 18, 2017).
- [3] Aish R, Peters B, Peters T. First build your tools. Smartgeometry Expand Archit Possibilities Comput Des 2013:36–49.
- [4] cadwork.com. Cadwork. Cadwork; n.d.
- [5] www.hsbcad.com. HSB CAD. HSB; 2016.
- [6] Stehling H. Woodpecker. n.d.
- [7] Chang T-C, Wysk RA. Computer-aided manufacturing. Prentice Hall PTR; 1997.
- [8] Braid IC. The synthesis of solids bounded by many faces. Commun ACM 1975;18:209–216.
- [9] STEHLING H, SCHEURER F, ROULIER J. BRIDGING THE GAP FROM CAD TO CAM: CONCEPTS, CAVEATS AND A NEW GRASSHOPPER PLUG-IN. Fabr. 2014 Negot. Des. Mak. DGO-Digital original, UCL Press; 2017, p. 52–59.
- [10] design2machine - the data transfer interface for wood constructions n.d. <http://www.design2machine.com/index.html> (accessed April 29, 2018).
- [11] Scheurer F. Digital craftsmanship: from thinking to modeling to building. Digit Work Archit Birkhäuser Basel 2013:110–129.

ISBN 978-82-326-4824-5 (printed version)
ISBN 978-82-326-4825-2 (electronic version)
ISSN 1503-8181



Doctoral theses at NTNU, 2020:238

John Haddal Mork

Parametric Timber Detailing

A parametric toolkit customized for detailing fabrication-ready timber structures

Doctoral theses at NTNU, 2020:238

NTNU
Norwegian University of
Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology

 **NTNU**
Norwegian University of
Science and Technology

 **NTNU**

 **NTNU**
Norwegian University of
Science and Technology