

Aarhus School of Architecture // Design School Kolding // Royal Danish Academy

Enhancing Collaborative Practices in Architecture, Engineering and Construction through MultiScalar Modelling Methodologies

Poinet, Paul

Publication date:
2020

Document Version:
Publisher's PDF, also known as Version of record

[Link to publication](#)

Citation for published version (APA):
Poinet, P. (2020). *Enhancing Collaborative Practices in Architecture, Engineering and Construction through MultiScalar Modelling Methodologies*.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



The Royal Danish Academy of Fine Arts
Schools of Architecture, Design and Conservation

**Enhancing Collaborative Practices
in Architecture, Engineering and Construction
through Multi-Scalar Modelling Methodologies**

Paul Poinet

A thesis presented for the degree of
Doctor of Philosophy

Supervised by:
Professor Mette Ramsgaard Thomsen
Associate Professor Martin Tamke

October 22, 2019

Title	Enhancing Collaborative Practices in Architecture, Engineering and Construction through Multi-Scalar Modelling Methodologies
PhD Thesis	
© Paul Poinet	
Printing and binding	Vester Kopi (Holmen)
Published by	The Royal Danish Academy of Fine Arts Schools of Architecture, Design and Conservation
Industry Partners	Design-to-Production, BuroHappold, Robert McNeel & Associates
Academic Partner	Centre for Information Technology and Architecture (CITA)
Primary Supervisor	Mette Ramsgaard Thomsen
Secondary Supervisor	Martin Tamke
Internal examiner	Daniel Sang-Hoon Lee
External examiners	Jane Burry, Robert K. Otani
Grant	The present thesis is undertaken as part of <i>InnoChain</i> . This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie-Sklodowska-Curie grant agreement No 642877.
Published in	2019



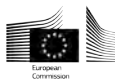
Det Kongelige Danske Kunstakademi Skoler
for Arkitektur, Design og Konservering

CITA



design
to
production

BUROHAPPOLD
ENGINEERING



innochain

To Rick Smith

Table of contents

Acknowledgements	11
1 Introduction	13
1.1 Personal Background and Motivation	13
1.2 Context	14
1.3 Aim and Research Objectives	15
1.4 Methodology	16
1.5 Contribution of the Thesis	17
1.6 Thesis Structure	18
2 State of the Art in Applied Parametric Modelling in AEC	21
2.1 An Overview of GUI Development and Interoperability Strategies for Improving Data Management and Exchange between CAD Software Packages	21
2.1.1 A brief overview of GUI development for Operating Systems (OS)	21
2.1.2 A brief overview of global interoperability concerns	25
2.1.3 GUI and interoperability development for CAD software packages to scale complexity and improve data exchange during the conception of large-scale complex architectural projects	29
2.1.4 Existing UI and UX concepts for visualizing, filtering, querying and exploring complex data sets	35
2.2 Applied Parametric Modelling in AEC: Current Problems and Existing Solutions . .	38
2.2.1 Limitations of DAG-based and integrative workflows in AEC	38
2.2.2 Concerns in parametric modelling: the paradoxical inflexibility of integrative design workflows	40
2.2.3 Dealing with changes during the design process: developing strategies for early design and late stages	41
2.2.4 Towards hybrid workflows: integrated and segregated strategies	42
2.3 Scaling Complexity: Separation of Concerns	44
2.3.1 Staged Models and Activity Networks	45
2.3.1.1 Front's Building Information Generation	45
2.3.1.2 Collaborative dependencies	46
2.3.1.3 Bauke de Vries' Activity Network	47
2.3.2 Design-To-Production's Digital Craftsmanship	48
2.3.3 Early stage modelling at BuroHappold Engineering	50
2.3.4 From Separation of Concerns to interoperability related challenges	51
2.4 Enhancing Interoperability	54
2.4.1 Building Information Modelling	55
2.4.2 Custom-made solutions for improving interoperability from within AEC practices	63
2.4.3 Existing meta-software and neutral formats: from a model-driven to a database-driven approach	79
2.5 Towards Transparency, Concurrent Design and Multi-Scalar Modelling	82
2.5.1 Enabling communication: connections rather than computation	82
2.5.2 Skepticism towards Transparency	84

2.5.3	Collaborative, concurrent design and common interfaces: skeleton models, adapter models and abstract networks	87
2.6	Conclusion	91
3	Towards Multi-Scalar Modelling for Building Design: A Theoretical Framework and Research Methodology	93
3.1	Multi-Scalar Modelling in Architectural Design Research: Origins and Existing Methodologies	94
3.1.1	Learning from the cybernetic precedents: the importance of interfacing, usability and front-loading enabling algorithmic and design thinking . . .	94
3.1.2	Origins of Multi-Scalar Modelling and preceding modelling concepts . . .	99
3.1.3	Multi-Scalar Modelling in architectural design research	100
3.1.4	Existing methodologies for Multi-Scalar Modelling in architectural design research	106
3.1.4.1	Approximating the models: levels, resolutions and uncertainties	106
3.1.4.2	Coarsening and uncoarsening strategies	109
3.1.4.3	Pipelines and workflows	109
3.1.4.4	Discrete Event Simulation (DES): linking levels and resolutions through Multi-Scalar Simulation and optimizations	112
3.1.4.5	Conclusion	115
3.2	Existing Discrepancy between Academia and Practice	116
3.3	Redefining and Expanding the Existing Methodologies of Multi-Scalar Modelling for the Context of Building Design	118
3.3.1	Segregating the levels	121
3.3.2	Encapsulating the workflows: the “Node-to-Code” approach	122
3.3.3	Differentiating “Multi-Scalar Encoding” from “Multi-Scalar Simulation” . .	123
3.3.4	Curating the workflows and enabling feedback	124
3.3.5	Conclusion	125
3.4	Research Methodology	126
3.4.1	Research through Design	128
3.4.2	Evaluating the experiments	130
3.4.3	Mapping the experiments	131
3.4.3.1	Modelling-based experiments	133
3.4.3.2	Schema-based experiments and search interfaces	133
3.4.3.3	Cross-practice collaboration experiments	134
3.4.4	Conclusion	135
4	Modelling-based Experiments: Free-form Timber Structures as a Series of Case-studies	137
4.1	Topological Mapping of Complex Timber Structures	138
4.1.1	From blank modelling to global networks: extracting and navigating through the different levels of resolution	138
4.1.1.1	The modelling techniques developed and employed at Design-to-Production	138
4.1.1.2	A1 BlankMachine: Propagating the blanks	140
4.1.2	Topological mapping in timber assembly	143
4.1.2.1	The spa and health center Solemar-Therme in Bad Dürkheim . .	143
4.1.2.2	Design-to-Production’s expertise	147
4.1.2.3	A2 Topological Mapping: Modelling Lap Joints	149
4.1.3	A3 Topological Mapping: Surface/Projection-based modelling	154
4.1.4	A4 Topological Mapping: Spatial Branching	159
4.2	Stress testing modelling-based strategies	162
4.2.1	WS1 B-MADE Workshop: Fabrication Constraints and Assembly Logistics	163
4.2.2	WS2 LogJam! Workshop: Structural Optimizations and Performance . . .	166
4.2.3	WS3 Exquisite-Timber-Corpse Workshop: Collaborative Practice	168
4.2.4	WS4 Future Wood Seminar: Speculating Design Spaces	170

4.2.5	A5 The Tallinn Architecture Biennale Installation Competition	174
4.2.5.1	Macro-level: global topological mesh and graph modelling	176
4.2.5.2	Meso-level: extracting the fabrication data from the assembly model	180
4.2.5.3	Micro-level: molding, laminating processes and physical prototyping	181
4.2.5.4	Results and remaining challenges	183
4.3	Conclusion: from Modelling to Data Management Concerns	184
5	Schema-Based Experiments and Search Interfaces for Late Stages	185
5.1	From Data Visualization to Adaptive Queries: Inter-linking Scales and Reconfiguring Hierarchies	186
5.1.1	B1 Visualizing Hierarchies: LayerFootprint	188
5.1.2	B2 Navigating across Hierarchies: LayerExplorer	191
5.1.3	B3 Reconfigurable Hierarchies and Adaptive Queries	193
5.1.3.1	Reconfigurable Hierarchies	193
5.1.3.2	Adaptive Queries	197
5.1.4	B4 Exploring Hierarchies: LayerStalker	200
5.1.5	Results, limitations and next steps	203
5.2	Building, Transferring and Receiving Hierarchies	204
5.2.1	B5 Building and Transferring Hierarchies: SchemaBuilder	204
5.2.2	B6 Receiving Hierarchies	209
5.2.3	Conclusion and future directions	212
5.3	Cross-Practice Collaborations: Workshops and Case Studies	212
5.3.1	C1 Sharing Schemas through Speckle and BHoM : a Speculative Design Workflow between Design-to-Production and BuroHappold	213
5.3.2	C2 SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows	218
5.3.3	C3 Piped Assemblies Workshop at CEDIM	221
5.4	Conclusion	227
6	Conclusion and Future Directions: Towards a Multi-Scalar Modelling Paradigm for AEC	229
6.1	Reinstatement of the Context and Research Goals	229
6.2	Summary of Findings	230
6.3	Contribution of the Thesis	234
6.4	Answering the Research Questions	236
6.5	Limitations	238
6.6	Perspectives	240
	Carried Secondments	243
	Bibliography	245
	Illustration Credits	255
	Glossary	259

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to the European Union's funding programme Horizon 2020 for research and innovation which enabled funding for the InnoChain European Training Network within which I undertook the present thesis, under the Marie-Sklodowska-Curie grant agreement No 642877. This fund-raising was made possible by the original initiative taken by both Prof. Mette Ramsgaard Thomsen and Assistant Prof. Martin Tamke at the Centre for Information Technology and Architecture (CITA) in Copenhagen, Denmark. They managed to set up the initial InnoChain network from the ground up and gather the many different academics and practitioners involved within the research programme. I wish to thank both Prof. Mette Ramsgaard Thomsen and Assistant Prof. Martin Tamke not only for such great effort, but also for the supervision of my thesis and their valuable guidance throughout the programme.

I would also like to thank my academic host, the Royal Danish Academy of Fine Arts Schools of Architecture, Design and Conservation (KADK), and all my colleagues at my host lab, CITA. It was a great experience to evolve in such an inspiring environment along with Assistant Prof. Paul Nicholas, Assistant Prof. Phil Ayres, and my colleagues, current or former Ph.D. Fellows and Research Assistants at CITA: David Andres Leon, Kasper Ax, Esben Clausen Nørgaard, Emil Fabritius Buchwald, Sebastian Gatz, Mary-Katherine Heinrich, Anders Holden Deuleuran, Henrik Leander Evers, Scott Leinweber, Ayoub Lharchi, Yuliya Sinke Baranovskaya, Tom Svilans, Vicki Thake, Ida K. F. Tinning.

I am also very grateful to both of my industrial partners: Dr. Al Fisher, Director and Computational Development Lead at BuroHappold Engineering in London, and Fabian Scheurer, Managing Partner at the consultancy practice Design-to-Production in Zürich. Their respective guidances and insights were extremely valuable, not only from a industrial perspective but also from an academic point of view. During my stays at BuroHappold, I also interacted with several members of the Computational Collective to whom I would also like to thank for their very valuable technical help and inputs: hats off to Eduardo Pignatelli, Puria Hesari, Isak Näslund and Arnaud Declercq. During my secondments at Design-to-Production, I also had the chance to meet all the involved consultants, to which I would also like to express my gratitude for their help and guidance: many thanks to Hanno Stehling, Johannes Kuhnen, Julian Höll, Michele Semeghini, Gianluca Tabellini and Sylvain Usai.

I also had the chance to pursue a secondment at McNeel Europe at Roger de Flor in Barcelona, during which I had the great opportunity to play Counter-Strike with Dimitrie Stefanescu (Ph.D. Fellow at InnoChain and initiator of the open-source Speckle Works platform) and get great technical support from both him and Luis Fraguada (Third party developer and support at McNeel Europe). I would like to thank both of them.

I would also like to thank all the persons involved within the InnoChain network for making this adventure possible and more particularly all my colleagues and Early Stage Researchers for the many different gatherings and inspirational moments we spent together across Europe: Zeynep Aksöz, Kasper Ax, Efilena Baseta, Giulio Brugnaro, Stephanie Chaltiel, Angelos Chronis, Ayoub Lharchi, Arthur Prior, Saman Saffarian, Evy Slabbinck, James Solly, Vasily Sitnikov, Dimitrie Stefanescu, Tom Svilans, Helena Westerlind.

Last but not least, I am more than grateful to my family, my parents and my friends David Andres Leon and Belen Torres who have been supported me through all these years.

Chapter 1

INTRODUCTION

This first introduction chapter is divided into six sections. The first section elaborates on the personal background of the author and his motivation to carry the present research work. The second section informs on the current context and explains why the carried research is relevant in such context. The third section declares the principal aims and research objectives. The fourth section introduces the research methodology that has been used to carry the different experiments composing the thesis. The fifth section clearly states the contribution of the thesis. Finally, the sixth and last section gives an overview of the general structure of the thesis by detailing the content of each chapter and how they relate to each other.

1.1 Personal Background and Motivation

During my master studies within the *Integrative Technologies and Architectural Design Research* program which I undertook from September 2013 to October 2015 at the University of Stuttgart, I have been involved in the conceptual development and construction of the *ICD/ITKE Research Pavilion 2014/15*, which demonstrated the architectural potential of a novel building method inspired by the underwater nest construction of the water spider: through a novel robotic fabrication process, an initially flexible pneumatic formwork has been gradually stiffened by reinforcing it with carbon fibres from the inside. I was mainly responsible for the development of a computational design framework generating the global pavilion's shape, which integrated both fabrication constraints and structural simulation (Vasey et al., 2015). The retained geometry was the last iteration from of a broader design exploration series that investigated many different membrane geometries, opening locations, footprint and boundary conditions. During the design process, important topological changes could affect existing design workflows that were not any more relevant for further modeling and simulation. Keeping the model specific but also flexible enough so it can support changes was definitely a crucial aspect of the design process. An other difficult part was to communicate different pieces of information and data to the different teams, either responsible for fabrication or structural analysis, which resulted in the creation of many residual, duplicated models that were often lost amongst our local files. Manual interventions were also constantly required in order to adjust integrative (primarily thought continuous) design workflows, correct unforeseen mistakes or bridge different software platforms that were not meant to communicate between themselves. For example, we partly used ladders and duct tape to properly stick the impregnated carbon fibers onto the ETFE membrane, instead of making constant use of the robotic arm and its custom made end-effector equipped with a fiber laying system. Such surrogates are explained by the time-frame and deadlines that oblige the project team to come up with fast, cheap and substitutive solutions when the developed system remains too shallow for production. This can be quite frustrating, especially when it is known that interoperability problems are most probably the main current struggle in the AEC industry.

The present research is therefore motivated by the realization that design processes, their respective implementation methodologies and the subsequent interoperability issues that they generate are generally too often overlooked and left aside for the (justified) sake of meeting deadlines and completing rapidly the physical product. I believe however that a deeper understanding of the developed design frameworks through the scope of Multi-Scalar Modelling

could enable the user to obtain much more control and flexibility throughout the whole design process. The present research attempts to verify this claim, and investigate how these mentioned problems could be tackled and solved on a building scale.

1.2 Context

The present research work has been carried within the InnoChain ETN network, which has received funding from the European Union's Horizon 2020 research and innovation program under the Marie-Sklodowska-Curie grant agreement No 642877. InnoChain aims to expand, synthesize and consolidate knowledge into computation-informed building design practice across academia and practice. The thesis inscribes itself as well at the intersection of academia in practice, through its academic supervision at the Centre of Information Technology and Architecture (CITA), and the involvement of two industrial supervisors: BuroHappold (London, UK) and Design-to-Production (Zurich, Switzerland), specialized respectively in engineering consultancy and in the geometrical rationalization of complex timber structures.

Today's digital design workflows in Architecture, Engineering and Construction (AEC) are segregated, as they are composed from multiple actors and stakeholders that work in isolation across multiple software platforms to complete different tasks. This is not a new problem as it was already raised 30 years ago:

"One of the problems concerning the automation of the architectural practice is the large number of small, distinct software packages. This means that the same data is stored at different locations for each of the software packages. This can cause inconsistent data when one omits to perform the modifications systematically in all versions. The maintenance of the data requires more effort than in case of one centralised representation. Moreover the different applications can use different codes for the same construction element. In this case the data has to be converted to the right format to be used by another application." (Mollaert, 1989)

Although initial efforts have focused on establishing common standards and data formats — such as the Industry Foundation Classes (IFC) initiative that began in 1994 — to exchange data across multiple modelling environments, communication remains cumbersome and is only enabled through email and file-based exchange procedures which are still stifling the design process, especially at the latest design stages during which communication becomes more and more crucial. In such context, it is therefore difficult to keep a consistent and continuous modelling workflow across all scales during the design process. This current paradigm leads to wrong modelling practices and misconceptions such as the attempt or idea of designing and representing a whole building project within one single file, the latter acting as the single "source of truth". As explained in chapter 2, such practices are idealized and fail at taking into account the actual fragmented landscape of the design process composed by different scales and the multitude of software packages used in AEC. Since modelling practices differ from one scale to another (as observed in chapter 3), one single environment might therefore not be suitable to model an entire building through all levels of resolution. Instead, the modelling framework needs to be deployed across multiple environments, where each can focus individually on a particular scale. Those different scales could then communicate, update and trigger each other through different simulation and optimization strategies.

In academia, architectural design research has already investigated how design workflows and modelling pipelines could be deployed to communicate information across multiple levels of resolutions, triggering different simulation frameworks that optimize specific tasks at different scales and stages in the design process (Tamke et al., 2014). This paradigm refers to the broader interdisciplinary concept of "Multi-Scalar Modelling", introduced in chapter 3, and transferred to the realm of architectural design research by CITA. Because the academic realm tackles the scale of the prototype or the demonstrator which is often conceptualized, analyzed and built by the same research team, it is possible to establish and deploy these simulation and communication protocols upstream in the design process. The building industry, however, tackles a much larger scale, which implies an heterogeneous network of processes, the involvement of a more decentralized team of

actors and stakeholders, working from remote locations and using a large panel of different software platforms.

The present thesis asks whether or not it is possible to reconcile the two realms described above: the fragmented landscape of design processes in the building industry with the emerging Multi-Scalar Modelling paradigm existing in academia. The relationship between these two domains is constantly explored and challenged through the different experiments that compose the thesis.

1.3 Aim and Research Objectives

The thesis examines the interdisciplinary concept of Multi-Scalar Modelling through the scope of the [AEC](#) domain's requirements to improve the existing design workflows in industry. Where present modelling paradigms consolidate the ambition to interface different design environments through a unified model, such as Building Information Modelling ([BIM](#)), the ambition of the thesis is to articulate how Multi-Scalar Modelling can support the creation of a network of models tuned to interface and communicate information across the design chain.

The present thesis takes as a starting point the Multi-Scalar Modelling framework formulated and established by [CITA](#) through the conception, production and realization of precedent design probes, prototypes and demonstrators (e.g. The Rise, Dermoid, Lace Wall and Stressed Skins). Those demonstrators introduced Multi-Scalar Modelling strategies enabling a direct communication between multiple scales, from material specifications at high resolution to the global design environment. The thesis attempts to extend this theoretical framework by adapting it to the building scale through further inclusion of industry concerns and problematics, provided here by both BuroHappold and Design-to-Production: trying to keep a consistent, continuous design workflow throughout the whole design process, from early design to late stages.

In practice, the current existing workflows in [AEC](#) are segregated, and this can be explained by two main reasons: the first is that trying to maintain a complete continuous digital chain is a wicked problem and is most probably promised to collapse ([Davis, 2013](#)). The second is that the different trades involved in a particular project are using a wide range of different software platforms, which hardly communicate between themselves, beyond traditional export-import practices and email attachment workflows. Therefore, the principle objective of this thesis is to redefine the Multi-Scalar Modelling framework specifically for Building Design and [AEC](#), focusing on the digital infrastructure across different scales and phases of a building project. This framework will demonstrate, through a series of conducted modelling experiments and prototypical interfaces, the possibility of implementing more consistent digital design workflows through the development of custom prototypical application and software interfaces, whose ultimate aim would be to solve some of the current issues faced by the [AEC](#) industry, especially during the conception and realization of a large-scale and geometrically complex architectural project.

The aim of the thesis is therefore threefold:

- **Validating whether or not the Multi-Scalar Modelling framework existing in academia can be transferred into the realm of the [AEC](#) industry.**
- **Establishing new means of simulating and communicating complex data sets throughout the whole digital chain of the building process.**
- **Creating a theoretical framework, acting as a larger software platform or infrastructure, that could host the multiple custom prototypical software applications developed throughout the thesis work.**

Therefore, the different experiments conducted through the thesis aim to facilitate and improve existing workflows by answering different research questions and validate several hypotheses, each one of them helping to draw the global picture of a larger speculative software platform. Each experiment attempts to answer at least one of the following research questions:

- ***How can a Multi-Scalar Modelling framework allow the designer to work across different scales in order to take into account multiple constraints related to material, fabrication and structural performances during both early design and late stages?***

- *How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?*
- *How would an ideal Multi-Scalar Modelling AEC-model look like and which requirements would it have to fulfill all user's requests? How could the multi-scalar model be interacted and which User Interface (UI) and User Experience (UX) concepts would be needed?*

1.4 Methodology

Although the research methodology used to carry the research work is developed in more details in section , it is first introduced here – as part of the introduction chapter – in order to help the reader in gaining initial insights into the present thesis and its related methodology.

In order to answer the above research questions, the present thesis has been conducted through an experiment-based method, built upon 18 different experiments (mapped in figure 3.19). These 18 experiments are design-led (Hauberg, 2012) and have been categorized into three different clusters: the *“Modelling-based experiments”*, the *“Schema-based experiments and search interfaces”* and the *“Cross-practice collaboration experiments”*. Each of these experiments are further classified into three different types: the *“design probe”*, the *“prototype”* and the *“demonstrator”*. These different categories have been introduced into the realm of architectural design research by CITA, which uses them as modes of evaluation and evidence through the institute’s own design and fabrication processes (Ramsgaard Thomsen and Tamke, 2009, 3). While the *design probe* is an hypothetical investigation that defines the design criteria and conjectures further research development, the *prototype* constitutes instead a “[...] *materially-led investigation allowing exploratory testing, of craft and material behaviour. The prototype answers and develops the design criteria of the design probe*” (Ramsgaard Thomsen and Tamke, 2009, 3). Since most of the experiments pursued in the present thesis work remain within the digital realm, the prototypes developed in this thesis are defined as digital “prototypical applications”. Indeed, their development led to the creation of digital tools or applications answering specific design problems. Contrary to an application software that is robust enough to be deployed across multiple machines and operating systems, and ready to be use by different parties, a “prototypical application” remains an experiment, although advanced enough to be tested against different contexts, multiple models or scenarios. Finally, the *demonstrator* deploys the prototype (or prototypical application) and tests it under real world constraints. Indeed, it has been defined as “[...] *an application-led investigation allowing interfacing with real world problems and constraints*” (Ramsgaard Thomsen and Tamke, 2009, 3). In the context of the present thesis, the demonstrator acts as a prototype but further tests the latter against real world data. “Real world” data is defined here as data coming from existing building projects or scenarios, provided here by the industry partners – Design-to-Production and/or BuroHappold Engineering.

Stress testing as an evaluation method

These carried experiments are then evaluated through one or multiple “in-action” methods (Schön, 1983) acting as “stress tests”. In software engineering, “stress testing” is defined as an activity that verifies the robustness of a piece of software by testing it beyond normal operative conditions (Weyuker, 1998). As the present thesis work does not pretend to undertake any robust and industrial software development, the term “stress testing” refers here to the deployment and testing of a prototypical application or demonstrator against multiple models or case scenarios, validating (or not) those same developed prototypes or demonstrators.

Three types of “stress tests” are evaluating the carried experiments: *stress testing the design workflow’s (in)flexibility* attempts to evaluate whether or not the speculative probe or prototype allows design flexibility and the enhancement of design possibilities, *stress testing prototypical applications against existing multiple source models* evaluates whether or not the prototypical application or demonstrator is able to function against multiple source models (modelling environments) or scenarios, and *stress testing prototypical design tools and methodologies through collaborative practices and speculative workflows* evaluates whether or not the experiment is robust enough to “survive” a real world cross-collaborative design workflow.

Means of composing the conducted experiments

Finally, the different series of experiments are framed and inter-linked by different means of composition, identified by Peter Gall Krogh in *“Ways of Drifting – 5 Methods of Experimentation in Research through Design”* (Krogh et al., 2015). In this paper, the author analyses different research-by-design doctoral theses which have been carried through either *accumulative*, *comparative*, *serial*, *expansive* or *probing* logics. Where the *accumulative* and *serial* approaches build knowledge iteratively by learning from the precedent experiment to carry the next one, the *comparative* approach treats the experiments more separately and focuses on comparing their similarities and differences. In a similar way, the *expansive* approach investigates multiple domains parallelly to broaden the research perspective. Finally, the *probing* approach explores *“design ideas as they emerge through design work”* (Krogh et al., 2015). For example, *probing* happens when different findings coming from separate research investigations could complement each other by unlocking new findings leading to a new research investigation area. Through the present thesis, this particular scenario happened when the findings made through the carried **“modelling-based experiments”** and the findings made through the carried **“schema-based experiments and search interfaces”** converged towards the investigation of the last **“cross-practice collaboration experiments”**. More generally, as the thesis work has been shaped through a back-and-forth dialog between academia and practice – therefore necessitating versatile means of conducting experiments to adapt to different contexts – the different means of composing experiments described so far and identified in isolation by Krogh through the examination of different theses, have been combined together when necessary.

1.5 Contribution of the Thesis

The present thesis is contributing to the body of knowledge constituting the existing digital design workflows in AEC by transferring the interdisciplinary concept of Multi-Scalar Modelling to building design. The different undertaken digital prototypes and demonstrators have proven that the existing segregated practices analysed in chapter 2 could be further improved and better communicate by deploying the Multi-Scalar Modelling strategies described in chapter 3 (which curate the transfer of data across multiple scales and levels of resolution), as well as custom software applications and search interfaces investigated in chapter 5 (which curate the transfer and management of data across multiple software platforms). These different deployments have enabled more flexible communication between multiple scales, software environments and stakeholders.

Subsequently, the thesis contributed in reconciling two realms whose respective paradigms usually operate on different levels (as argued in section 3.2): the academic realm in which digital workflows can be fully integrative and multi-scalar – as the scale of the investigated demonstrators in academic research is usually manageable by a small team of researchers – and the AEC realm within which digital design workflows are currently segregated, stifling the design processes at the latest stages.

From the above, the thesis contributed in establishing a theoretical framework that applies the interdisciplinary concept of Multi-Scalar Modelling to the realm of building design. The theoretical framework’s key point lies in the hybridization of workflows and strategies between:

- the **modelling-based techniques** – block modelling, graph modelling, DAGs workflows and skeleton models – investigated through the modelling-based experiments in chapter 4 that enabled the passing of information across multiple scales and at early stages
- the **schema-based workflows and search interfaces** – complex data sets visualization, navigation across hierarchical directory structures, adaptive queries and building schema strategies – investigated through the schema-based experiments in chapter 5 that enabled better management of complex data sets generated at the latest stages in the design process.

These two modes of working – one focusing at early stages on continuous and seamless communication strategies triggering multiple simulation frameworks at different scales, and the other operating at the latest stages in a more discrete manner to manage complex data sets across hierarchical directory structures – are synthesized and hybridized in section 5.3 through the series

of conducted **cross-practice collaboration** experiments. In particular, the demonstrator experiment “C1 | *Sharing Schemas through Speckle and BHoM: a Speculative Design Workflow between Design-to-Production and BuroHappold*” looked the transfer of a data-rich object whose schema is able to adapt to two different modelling paradigm, from Design-to-Production’s modelling strategies to BuroHappold’s structural analysis models. While the schema of the object is used here as a skeleton model (defined in section 2.5.3) to transfer the minimum of information necessary to reconstruct the same object within two distinct modelling environments, the modelling-based strategies could also be deployed to further process the object in parallel within these two environments through DAG modelling and back-propagation behaviours. This hybridization of strategies is illustrated in figures 5.22 and 5.23. Here, the hybridization of strategies is deployed at the level of the object (one architectural component). However, the same hybridization between the modelling-based techniques and the schema-based workflows and search interfaces has also been demonstrated through the experiments “C2 | *SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows*” and “C3 | *Piped Assemblies Workshop at CEDIM*”, within which segregated DAG modelling/simulation pipelines were interlinked through skeleton models and Speckle senders and receivers. A search interface – illustrated in 5.25 and 6.3 – enabled the mapping of the overall workflow on a higher-level, from which the user could retrieve data-rich objects at any stage in the design process.

It is important to remind here that the present thesis does not argue that the specific strategies developed for the above experiments are universal and would solve all the current challenges faced by the AEC sector. However, they should act as strong examples (or role models) in which the hybridization of continuous DAG modelling and discrete schema-based, search interfaces are able to work in harmony to deliver highly curated workflows enabling consistent delivery of a design project across all phases and scales. The expert-user – to which the present thesis is addressed – could take this hybridization of strategies as a basis to establish its own workflows, but the curation of the data itself is left to this same expert-user who has the final responsibility of choosing what interfaces and what strategies should be deployed to answer his/her specific design problems.

1.6 Thesis Structure

The present thesis is divided into six different chapters: the current introduction, a state of the art chapter focusing on applied parametric modelling practices in AEC (Chapter 2), a background chapter introducing both the Multi-Scalar Modelling paradigm as well as the research methodology used to carry out the different experiments present in the thesis (Chapter 3), two chapters focusing on the carried series of experiments (Chapter 4 and 5) and a concluding chapter (Chapter 6).

Chapter 2 focuses on the state of the art in applied parametric modelling practices in AEC. It will first contextualize these practices by giving an overview of both GUI (Graphical User Interface) development and interoperability outside and within the AEC (Architecture, Engineering and Construction) realm. Section 2.2 will then focus on the current practices of parametric modelling in AEC and place the emphasis on the main concerns and challenges encountered by this field. Taking as point of departure these particular challenges, section 2.3 will describe existing proposed solutions developed by progressive and leading architecture, engineering and consultancy practices to both scale up geometrical complexity and improve interoperability. Generally, the adopted strategies are relying on a Separation of Concerns (Dijkstra, 1982) (Pena De Leon, 2014), consisting of segregating the design workflows and activity networks (de Vries, 1995) into smaller processes, called “staged models” (Van Der Heijden et al., 2015) or “design cycles” (Tessmann, 2008). From these observations and insights, section 2.4 will focus on the need for improving interoperability and section 2.5 will argue for the development of more neutral format exchange protocols, transparency, and the development of custom UI (User Interface), UX (User Experience), communication and search interfaces.

Chapter 3 introduces the interdisciplinary concept of Multi-Scalar Modelling paradigm before focusing on its application within the realm of architectural design research. Its origins are discussed through both the review of preceding modelling concepts such as *parametric modelling*, *computational modelling* and *generative modelling*, and existing architectural design research demonstrators which have used Multi-Scalar Modelling within their respective design processes.

Section 3.2 nuances the argument by highlighting the dichotomy existing between the design practices deployed in academia and industry. While the former is able to opt for fully integrative design workflow (due to the small scale of the tackled demonstrator), the latter cannot afford for such strategy as the design landscape is more fragmented, geographically (e.g. architects, engineers and contractors working from remote locations) and technologically (use of multiple software packages). Finally, section 3.3 proposes to redefine the emerging Multi-Scalar Modelling methodologies that currently answers architectural design research needs, so that those methodologies could also be applied on the building scale within [AEC](#).

Chapter 4 carries out an initial series of **modelling-based experiments** which take as precedents the specific modelling practices developed at Design-to-Production allowing the detailed description of each element composing large-scale free-form timber structures. As these existing modelling practices remain fragmented and the design process segregated across multiple 3D models, the modelling experiments carried in this chapter explore Multi-Scalar Modelling methodologies (graph-based modelling, projection-based modelling, coarsening/uncoarsening strategies) and investigate how these deployed methodologies could enhance Design-to-Production's current modelling practices by inter-linking the different levels of resolution through the whole modelling process chain. While section 4.1 develops and investigates these Multi-Scalar Methodologies through a first series of modelling experiments, section 4.2 applies these same methodologies through a second series of modelling experiments during different workshops (taught or attended) and throughout the design process of the Tallinn Architecture Biennale Installation Competition. Through this chapter, it has been found that despite the strongest front-loading strategy ([Smith, 2007](#)) enabled through the introduced Multi-Scalar Modelling methodologies, challenges related to communication bottlenecks and data management concerns remained. Therefore, chapter 5 switched the focus from modelling to schema structures and data management concerns.

Chapter 5 carries out a second series of **schema-based experiments and search interfaces** which investigates different strategies to better navigate through and between data-rich models at the latest stages of a building design process. This series of experiments also takes as a precedent existing data-base management practices employed both at Design-to-Production and BuroHappold Engineering. From this current working practice, different concepts and prototypical applications have been developed in order to both visualize hierarchical schemas, navigate amongst them, unravelling implicit relationships and perform adaptive, diagonal queries. Those developed strategies operate on different levels, from the adaptive schema structure of the object level to the multiscale data-visualization and search interface of the project level.

Section 5.3 synthesizes and merges the knowledge gained through both the series of modelling-based experiments (e.g. graph-based modelling modelling workflows, projection-based modelling, coarsening/uncoarsening strategies, skeleton model interfaces) and through the series of schema-based experiments (e.g. navigating across hierarchies, operating structured and unstructured queries), by carrying a third and last series of **cross-practice collaboration experiments**. These experiments deploy speculative cross-platform workflows between workshop participants acting as fictive stakeholders.

The three series of experiments (**modelling-based experiments**, **schema-based experiments and search interfaces** and **cross-practice collaboration experiments**) described above and carried through the chapters 4 and 5 feed into the conclusion in chapter 6, in which it is argued that the fragmented landscape of design processes existing in the building industry is actually compatible with the current Multi-Scalar Modelling framework established in academia, by pairing, hybridizing the latter with smaller task-oriented automations (e.g. search interfaces) enabling more efficient modelling at the latest stages in the design process.

Chapter 2

STATE OF THE ART IN APPLIED PARAMETRIC MODELLING IN AEC

This chapter will discuss the history and state of the art of both **GUI** (Graphical User Interface) – to better manage complexity – and interoperability – to better share data – outside and within the **AEC** (Architecture, Engineering and Construction) realm. It will then focus more particularly on applied parametric modelling in **AEC**, highlight the current concerns and challenges encountered by this field and describe existing proposed solutions to both scale up geometrical complexity and improve interoperability. These solutions will be illustrated by examples of existing design workflows which are continuously developed and maintained by leading architecture, engineering and consultancy practices. Finally, based on those observations and first insights, the last section will start to draft an initial conceptual framework addressing important topics, such as: the need for a neutral format exchange protocol, the enabling of transparency, the design of common interfaces and their further improvement through meaningful **UI** (User Interface) and **UX** (User Experience) means. After assessment of those different concepts, a first definition of a Multi-Scalar Modelling framework for building design will be given, before being further developed in the following chapter.

2.1 An Overview of **GUI** Development and Interoperability Strategies for Improving Data Management and Exchange between **CAD** Software Packages

In attempting to understand data management and interoperability issues of large-scale and geometrically complex architectural research projects, the history of **CAD** (Computer-Aided Design) in the **AEC** industry needs to be carefully analyzed from a modelling and managerial, interoperability perspective, rather than a purely modelling perspective which is usually the main focus of related historical research. Indeed, there is already a lot of literature on the history and development of **NURBS** (Non-uniform rational B-spline) modelling software focusing on the description of more accurate, cutting-edge modelling techniques helping the user in the creation, modification, analysis, or optimization of a design. Therefore, the exercise won't be repeated here and this section will focus instead on the topics of parametrics, scalability, data management and interoperability. Before talking elaborating on those topics from the perspective of the **AEC** sector, the next couple of sections look at the precedents both in term of **GUI** to enable data management within Operating Systems, and tackle the issue of Interoperability at a global scale.

2.1.1 A brief overview of **GUI** development for Operating Systems (OS)

The Oxford English Dictionary defines a **GUI** as follows: *“an interface in which programs, files, data structures, commands, etc., are represented on screen by graphical symbols (such as icons, menu items, and windows) which can be manipulated or activated directly without the need to learn a command language.”* Susan B. Barnes reformulated this definition, referring more specifically to the manipulation of “objects”: *“Graphical User Interface is a method that allows computer users to see and*

directly manipulate representations of objects on the computer screen, rather than addressing the objects through an intervening command language code.” (Barnes, 1995). In other words, the GUI adds an extra layer of interaction between the user and the manipulated data or objects. This dichotomy, or *separation of concerns* between the interaction layer and the data layer gave birth to the terms “front end” and “back end”, which are still used today to qualify the skills of software developers, which can be referred as “front end developer”, “back end developer”, or both.

Origins

It is difficult to draw a detailed history of GUI development, as it is both quite recent and not very well documented, as software development keeps overriding old GUI versions with the newest ones available on the market. This paragraph will attempt to highlight the historical main points that are the most interesting to consider from a CAD software perspective.

Many authors seem to agree that the interest for GUI found its origin in an influential essay written by Vannevar Bush, director of the U.S. Office of Scientific Research and Development, entitled “As We May Think” (Bush, 1945) in which initial GUI concepts and first ideas that anticipated hypertext were introduced. This particular paper inspired Douglas C. Engelbart, a young naval technician, in developing between 1962 and 1968 the most major features of today’s computer interfaces: the window-style display screens and the mouse, which were presented during a public demonstration held on the 9th of December 1968, in the Augmentation Research Center at Stanford Research Institute in Menlo Park. Around the same time, Ivan Sutherland completed his thesis on his Sketchpad program introducing also innovative GUI concepts (interactive design and 3D modelling) as well one of the first object-oriented paradigms (Sutherland et al., 1963).

Introducing Directory Tree Structures

It is only two decades later after Sutherland’s and Engelbart’s breakthroughs, when the Apple’s Lisa’s UI was released in 1983, that hierarchical data structures were introduced and screen icons could finally “represent all files in the file-system, which could then be browsed through using a hierarchal directory structure where each directory opened in a new window.” (Reimer, 2005).

With the release of Windows 3 in 1990 and Windows 95 in 1995, File Explorer (previously named “Windows Explorer” and “File Manager”) strengthened such hierarchical UI and allowed the user to access very intuitively the file systems organized as a directory tree structure. From this interface, the user is able to perform different operations, such as navigating through the child and parent directories, copy and paste a branch from one location to another, deleting and/or renaming folders and files etc. Such operations seem trivial at first sight since we are used to perform those tasks on an everyday basis, but the sophistication of data-management underlying these operations should inspire us for manipulating geometrical data through different scales (granularity and nesting), an important aspect of the broader topic of Multi-Scalar Modelling for Building Design. However, strictly hierarchical interfaces might also present some negative aspects which will be highlighted in the next paragraph.

Reconfigurable hierarchies

During the development of Apple’s LISA’s GUI, there were quite a few design iterations before finalizing the final on the iconic desktop. One of them (Figure 2.2) proposed a non-hierarchical faceted search as an alternative to traditional directory tree structures:

“We were interested in avoiding a strictly hierarchical filing system. We wanted to free users from having to decide the correct place to file a document and then the converse problem of trying to find where the document was filed. The upper area of our document browser contained various attributes that could be selected to narrow the choice of documents. As attributes were selected, documents with those attributes were displayed in the lower area. In this model, documents could be quickly located by type of document, keyword, author, and so on.” (Perkins et al., 1997, 48)

Even though such non-hierarchical concept seems to be more intuitive than a strict hierarchical search interface in some aspects, it actually presents downsides when it comes to others:

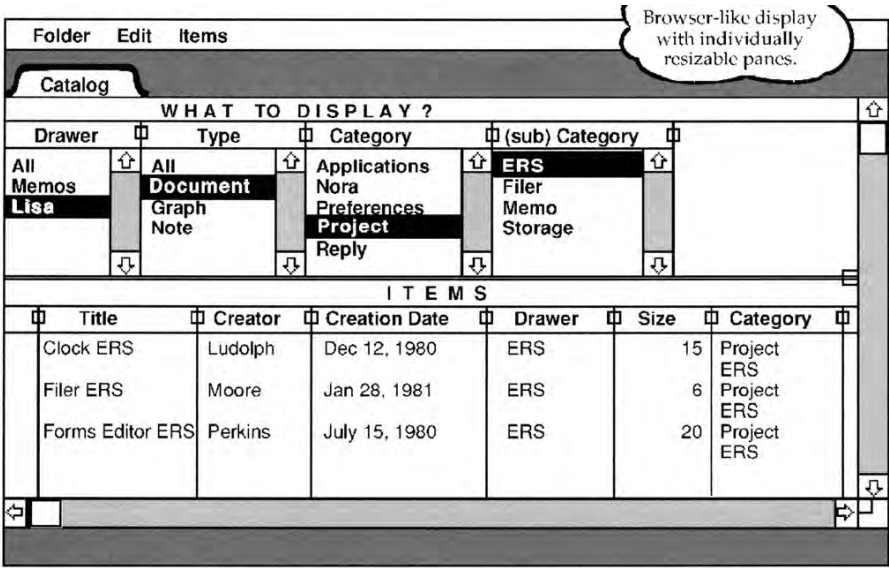


Figure 2.2: Desktop Manager – The Document Browser (December 1980) presenting a non-hierarchical faceted search interface.

In order to overcome such issue, the authors proposed a “Placeless Documents” system, an experimental prototype within which *“properties are the primary, uniform means for organizing, grouping, managing, controlling, and retrieving documents. Document properties are features of a document that are meaningful to users, rather than to the system.”* This approach does not leave aside hierarchical information, which can still be set through the user properties themselves. It is only a decade after the publication’s date of the two previously quoted papers that software companies developed interfaces allowing the user to add tags/properties on the fly. Microsoft Windows exposed this sort of functionality only with the release of Windows 7 in July 2009, which is still working today in Windows 10, but still only for specific types of files (such as .jpeg or word documents).

The need for reconfiguring hierarchies is not a problem that is exclusively related to the Operating System’s GUIs. Such desire or request is quite universal and cross-disciplinary. It could both satisfy social media users, music listeners and DJ professionals (to take only a couple of examples). In the first case, we know that social media makes full use of user-centred properties, most commonly known as “hashtags”. Hashtags basically allow anyone to add any sort of tags and categories to a document (most often a text or a picture) which can be found again by anyone else who would use the same hashtag. Items can be then sorted based on their property value and not on a single centralized hierarchical system. Regarding music, in a recent RA (Resident Advisor) Exchange podcast produced by Kerri Chandler who investigated how DJs organize their music collections, the DJ Avalon Emerson shared the method she classifies her files today¹. Emerson stated that in her first attempt, she would *“simply create normal playlists by dragging files into it”*. After a while, this apparently simple and straightforward approach became problematic because she would *“forget either what the signifier was for those playlists or forget about the playlists entirely, which would therefore mutate and change meaning, becoming finally useless.”* Emerson overcame such frustration by replacing the traditional playlist organization system by assigning different tags and properties on-the-fly after listening to a specific song: *“I just listen to a song and I qualify it based on tags that I created myself. A song can contain multiple tags and none are exclusive. Something can be “breaky”, “acidy” and/or “techno”. Based on all these tags, it is possible to generate intelligent playlists based on a specific combination of all these tags. [...] You can set up a bunch of boolean rules and then everything that fits those rules will show up on your screen.”*

¹The podcast can be listened at this address: <https://soundcloud.com/ra-exchange/ra-exchange-382>. The part about DJ Avalon Emerson starts at 10:18.

Drag and drop models

Besides the development of GUI features, the history of GUIs presents also interesting cases of UX's for managing complex data sets. This is the case of the classical "drag and drop" feature, which was also introduced during the development of LISA's GUI:

"The idea of "drag and drop" was also invented at this time, and the concept of using drag and drop to do file manipulation (for example, selecting a group of files with the mouse and then dragging them to a new folder to copy them) naturally developed from this concept." (Reimer, 2005)

"LISA [...] added several new elements common in today's systems, e.g., the menu bar, trash can, stationery, drag-and-drop, undo, desktop menu, clipboard, interactive tutorials, and exceptional graphics." (Ludolph and Perkins, 1998)

The drag and drop feature has been both further developed within the newest OS of both Mac and Windows. The user is now able with just a few clicks to move or copy-paste a set of folders towards a new location, and to modify the tree-like structure according to its needs.

This very brief overview of GUIs for Operating Systems helps here to understand that both the hierarchical, nested nature of Directory Tree Structures and the more horizontal search interfaces enabling the filtering of objects by category are very important aspects of the UX, as well as the drag and drop feature, when it comes to manipulate objects across scales, between the leaves (horizontally) or through parent-children relationships (vertically). Such interfaces barely exist within the CAD software packages developed for the AEC community, as these software focused more particularly in solving precise modelling problems. Integrating more GUI features within those software packages could be very useful to better manage geometrical data across scales.

2.1.2 A brief overview of global interoperability concerns

Interoperability: a global concern

The Oxford English Dictionary defines a Software Interoperability as follows: *"The ability of computer systems or software to exchange and make use of information."* Interoperability is therefore defined as technical challenge to enable better communication between software and machines. However, before starting to be concerned about the need of enhancing interoperability workflows, one should also ask whether or not such communication bridges are really desired, because of organizational and legislative issues concerning data ownership.

A fragmented, multi-layered landscape

In *"Achieving Interoperability in Critical IT and Communication Systems"*, Desourdis et al. describe governmental interoperability as an existing fragmented landscape in which information circulates between stakeholders across multiple governance layers (Figure 2.3), or levels:

"The local governance level [...], the second governance level [...], the third [...], the national level [...], and the multinational [...] show the complexity of developing an information-sharing architecture that accounts for each organization depending on the chosen scenario." (Desourdis et al., 2009, 94).

The authors observe that each level has its own stakeholder organizations which have their specific ways to operate and exchange information, disorganizing and obfuscating the whole workflow:

The governance layers [...] include overlapping responsibilities, and conflicts with and among disciplines and jurisdictions that will affect information sharing for split-second preparedness and response. These primarily civilian organizations are less regimented, organized, or centrally directed than the military organizations in Washington, D.C. and Pearl Harbor. Each governance level has its own responder-receiver and stakeholder organizations. (Desourdis et al., 2009, 94).

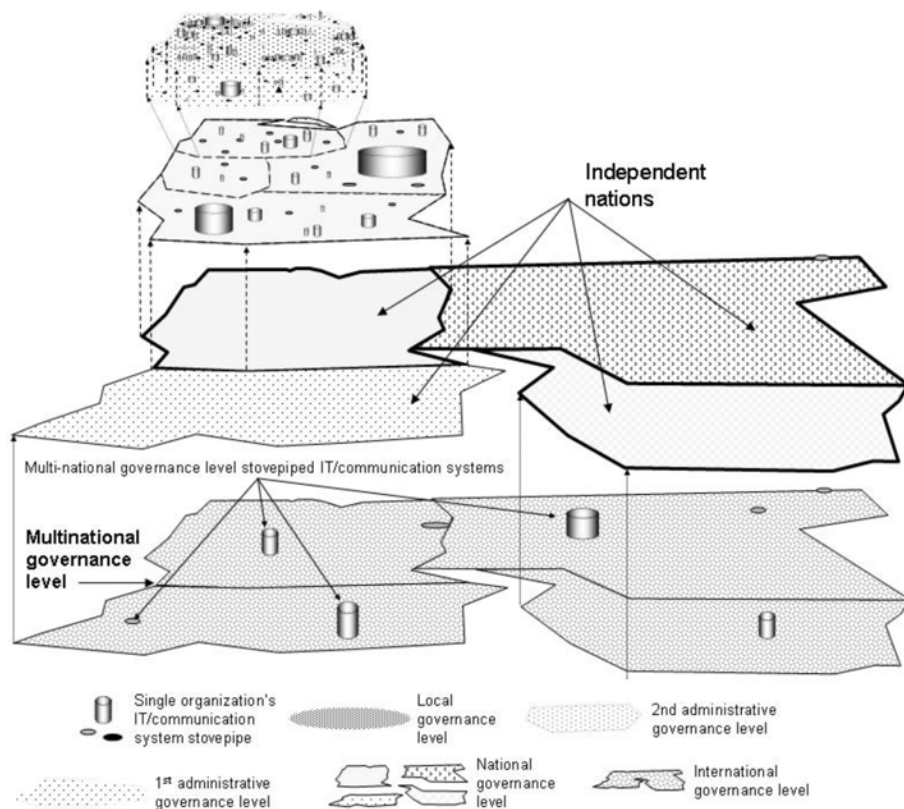


Figure 2.3: National-level stovepipes (Desourdis et al., 2009, 97)

The multi-layer governance model presents a fragmented, disorganized landscape in which information hardly flows between levels. Desourdis et al. suggested that the main cause of interoperability failures might be more of a organizational than technological nature: by analyzing the information flow between the involved governments before the Pearl Harbor attack, the authors concluded that interoperability deficiencies might be caused by other factors than technology itself: organization, assumption, omission, verification and supervision are the five first mentioned interoperability failure factors amongst a list of 25 others (Desourdis et al., 2009, 49).

More particularly, the fragmentation in the **AEC** industry landscape has also been observed and criticized 30 years ago:

"The degree of vertical fragmentation (between project phases, e.g., planning, design, and construction) and horizontal fragmentation (between specialists at a given project phase, e.g., design) in the U.S. AEC industry is unparalleled in any other manufacturing sector. The latest census of the U.S. construction industry reveals that it consists of over 1 400 000 establishments, of which over 930 000 have no employees and the remainder have an average of ten employees. The designers of constructed facilities are similarly fragmented by speciality area and by speciality within speciality. [...] Moreover, the buyers of construction are very fragmented. Individual real estate developers, home buyers, entrepreneurs, and a myriad of state and local agencies constitute a very fragmented customer base." (Howard et al., 1989)

Section 2.3.4 describes a very similar hierarchical, multi-level fragmented landscape amongst the different stakeholders involved in the design and construction project of large-scale and complex architectural projects.

Open vs. closed standards

To unite the fragmented landscape described above, it seems therefore that interoperability needs to be improved both on an organizational and technological level. One of the main answers to that

problem is to use shared, open standards and open-source² code so that seamless interoperability and communication can happen across levels and trades.

In 1995, Jonathan Band and Masanobu Katoh wrote the book *“Interfaces on Trial: Intellectual Property and Interoperability in the Global Software Industry”* (Band and Katoh, 1995), which laid out the principal concerns about the urge of global firms to exercise proprietary control and copyright over their own interface specifications, an attitude that could present a threat to interoperability on a global scale. This attitude was highlighted and illustrated by Pamela Samuelson, reporting on IBM’s decision to exercise full control over its interface specifications:

“Twenty years ago, IBM Corp. was the most vigorous advocate of (very) “strong” intellectual property rights for computer programs. Without strong copyright protection for programs, IBM contended, there would be insufficient incentives for firms to invest in software development. Its senior executives and lawyers contended that [...] interface specifications were among the original elements of computer programs that copyright did and should protect; and that reverse engineering of computer programs for purposes such as achieving interoperability constituted copyright infringement.” (Samuelson and Sherman, 2006)

Such behaviour was against Band and Katoh’s personal views on the matter, pointing out with relief that the United States case law *“indicates [...] that there is no doubt concerning the basic proposition that copyright does not protect [...] interface specifications”* (Band and Katoh, 1995, 165). The definition of “interface” is both defined by the authors and the Oxford English Dictionary as follows: *“A device or program for connecting two items of hardware or software so that they can be operated jointly or communicate with each other.”* Owning full copyrights to an interface thus results in the obstruction of interoperability workflows to the persons that do not possess the appropriate software and/or hardware. At a bigger scale, those obstructions also ensue the multi-layered fragmented landscape described above in *“Achieving Interoperability in Critical IT and Communication Systems”* (Desourdis et al., 2009, 94). In contrast, open standards would prevent such scenario to happen but could present a threat to major software companies, which would prefer to keep their source code closed for economical reasons.

Today, open-source software is less perceived as a threat by the global firms. Indeed, two decades after the publication of their first book *“Interfaces on Trial”*, Band and Katoh wrote a second version: *“Interfaces on Trial 2.0”*, elaborating on the growing interest for open standards, both by the wide community of software developers and by the companies themselves:

“As open-source software has been adopted more widely by major information technology firms and their corporate customers, the dispersed developers working within particular open-source programming environments have coalesced into networks of varying degrees of formality.” (Band and Katoh, 2011, 184)

Through the illustration of an existing legal case, the authors also noticed that *“the Federal Circuit significantly strengthened the remedies available to an open-source licensor against a breaching licensee. In particular, the licensor can seek statutory damages. This, in turn, will discourage a licensee from attempting to “hijack” an open-source product and render it less interoperable.”* (Band and Katoh, 2011, 185). More than being well protected by law, open-source software might presents itself as an economical advantage for large companies such as IBM :

“Among the key advantages of open-source is that the difficulties and costs of software development can be distributed among many contributors. IBM may be contributing \$100 million a year to the development of Linux, but firms such as Nokia, Intel, and Hitachi are making substantial investments as well.” (Samuelson and Sherman, 2006, 5)

Therefore, open-source software can be viewed here as a sustainable business model that could enable better interoperability for the wider community of users and software developers, which could also benefit economically from it. The video game industry is also trying to develop and use open

²According to the Oxford English Dictionary, open-source software is defined as follows: *“denoting software for which the original source code is made freely available and may be redistributed and modified.”*

standards, such as the Pixar’s Universal Scene Description (USD): “*Universal Scene Description (USD) is the first publicly available software that addresses the need to robustly and scalably interchange and augment arbitrary 3D scenes that may be composed from many elemental assets.*”³ USD is an open-source project⁴ developed and maintained by Pixar, allowing the exchange of different components for scene description:

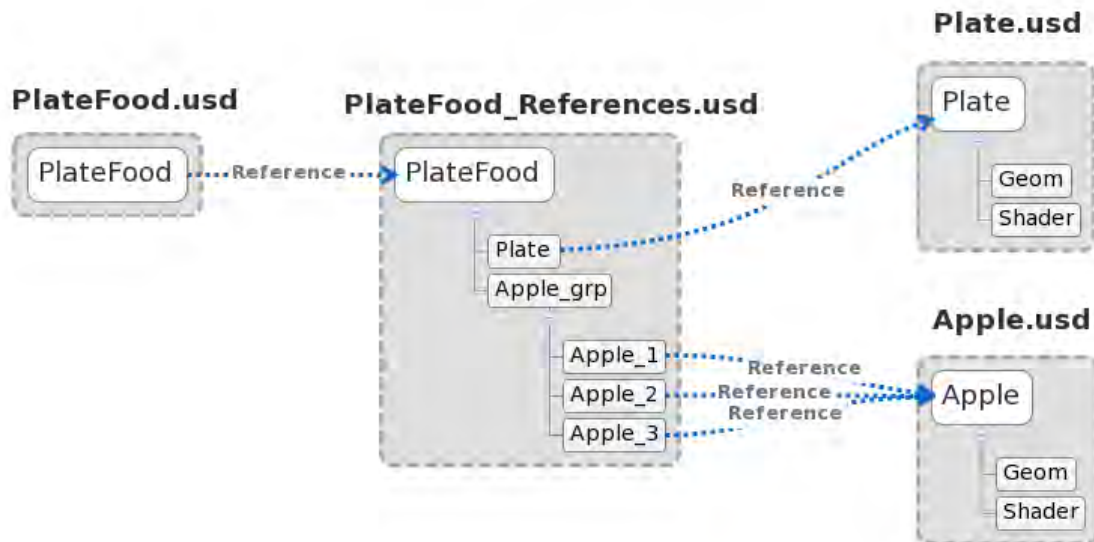


Figure 2.4: Example of an aggregate model using the USD pipeline.

“USD also enables assembly and organization of any number of assets into virtual sets, scenes, and shots, transmit them from application to application, and non-destructively edit them (as overrides), with a single, consistent API, in a single scenegraph. USD provides a rich toolset for reading, writing, editing, and rapidly previewing 3D geometry and shading. In addition, because USD’s core scenegraph and “composition engine” are agnostic of 3D, USD can be extended in a maintainable way to encode and compose data in other domains.”

Different USD components can be combined across multiple models, as in figure 2.4.

Skepticism towards open standards and frameworks

Even if full interoperability is enabled by open standards, it might not necessarily mean that it will be adopted by all. Indeed, seamless interoperability workflows might have a significant impact on business model of the companies involved, raising the primary question of ownership: who owns the data? Do people really want to share everything? In which context? Breaking the data silos would mean to rethink the current ways of manipulating data and would imply an organizational paradigm shift. It is fair enough to ask whether or not the various stakeholders involved in the design process are ready to share their complete data models (see 2.5.2).

³<https://graphics.pixar.com/usd/docs/index.html>

⁴<https://github.com/PixarAnimationStudios/USD>

2.1.3 GUI and interoperability development for CAD software packages to scale complexity and improve data exchange during the conception of large-scale complex architectural projects

The first subsection of this chapter described the rather slow development of GUIs within Operating Systems, and the second subsection highlighted that interoperability protocols and open-source software are quite a controversial topic in the global industry. This section will attempt to elaborate on the same topics, but exclusively from a CAD software perspective applied to the AEC sector's needs. Before doing so, the next paragraph attempts to unravel a research gap that exists both in academia and in software development: the neglected development of GUIs and Interoperability interfaces in CAD software packages to manage complexity and improve data exchange across platforms.

Complex modelling versus managerial concerns: unraveling a research gap

It seems that the advent of the computer within the AEC divided the architectural scene into two categories of practitioners and researchers: the ones interested in the pure managerial aspect of the profession and those interested exclusively in experimenting with design and in novel ways of materializing design.

In the first category, researchers are aiming at improving the current interoperability standards by publishing new research about BIM related protocols and the addition of new dimensions and classes to the IFC standard, and their peer practitioners are usually working as BIM managers (Barison and Santos, 2010) on standard construction projects. By focusing exclusively on interoperability concerns, this first category of researchers and practitioners fails to address the specific need for hybridizing those mentioned managerial aspects with geometrical complex aspects during late stages, so essential to architects working on large-scale and complex architectural projects.

In the second category, researchers are pushing a very specific agenda centered around material experimentation, robotic fabrication and the making of demonstrators (Knippers et al., 2018), and their peer practitioners – involved at an industrial scale in the materialization of complex shapes – are usually pressured by tight deadlines and responsible to deliver consistent architectural products, preventing them of spending too much time in improving and developing consistent, reusable digital design workflows that would better suit their specific needs in the long term (Scheurer, 2012, 130). This second category of researchers and practitioners is therefore overlooking the managerial aspect of the discipline, either voluntarily – in academia, to pursue a very specific agenda – or involuntarily – in industry, for very practical reasons.

Because of this strong dichotomy, it is quite rare to find middle-ground research, literature and historical precedents that focus in depth on the solving of managerial aspects of large-scale and complex architectural projects through the development of specific GUIs and interoperability interfaces for CAD software. This dichotomy of interest within practice might be one of the main explanations behind an existing research gap and poor development of GUIs and interoperability interfaces to manage large-scale and complex architectural projects within existing CAD software packages. Therefore, the latter is also divided into two categories:

- **CAD packages to model complex geometries**

The first type of CAD software is specialized in the modelling of complexity with very poor management interfaces

- **CAD software packages for Building Information Modelling**

The second type of software is allowing a better management of standard building data (BIM) with very poor aptitude to model complexity:

“BIM software limits the amount of direct modeling, using internal algorithms and embedded knowledge about the construction domain.” (Boeykens, 2012, 453)

This parallel dichotomy in the software industry might also be explained by the low ratio of existing complex buildings compared to the very high ratio of standard architectural projects that are tackled on an every-day basis and on a global scale. Indeed, answering practitioners demands who are working on the most challenging projects would be less economically viable for the

software industry than developing more standard, BIM protocols for the wider community of BIM managers working on more regular projects. A more detailed description of those two types of software will be developed in the next paragraphs.

From modelling complexity to the enhancement of software interoperability

To illustrate the first mentioned category of software capable of modelling complex geometries, Rhino3D (a commercial CAD software developed by Robert McNeel & Associates and founded in 1980) will serve here as a main example. This software is specialized in modeling mathematically precise representation of curves and freeform surfaces through NURBS. Besides its modelling features, Rhino3D deliberately aims at remaining as open as possible by delivering a clear API, by initiating and developing an interoperability framework for NURBS geometry with external applications (openNURBS⁵), and by offering the possibility to import/export a large number of file format in order to third party developers to enable communication with its own platform:

“On top of manual modelling tools, Rhino3D offers a multitude of options for users to operate their software from basic scripting to Application Programming Interface and plug-ins. openNURBS is also a Robert McNeel & Associates initiative to enable and encourage interoperability of geometric models.” (Mirtschin, 2011, 2)

Although its openness and ability to process many different file formats, Rhino3D is lacking consistent GUI enabling the user to scale and process building data. Indeed, even though Rhino3D already exposed within its core library user data functions allowing the user to add property-value information types to geometrical objects through the API, the corresponding GUI for operating those same actions on a higher level (closer to the user) was only developed for the release of the software’s 6th version in February 2018, i.e. more than 20 years after the creation of Rhino3D. John Mirtschin summarized both the potential and limits of Rhino3D by highlighting its ability to manipulate complex geometry but its poor capability in processing model data for industry:

“Rhino3D [...] can offer a number of advantages for project design including generative model creation and NURBS shape representations that are easily produced/manipulated and capable of representing advanced forms. However this software is not focused on architectural industry, or any industry in particular. There are many tasks and reasons why designers would need to utilize the model data in external software.” (Mirtschin, 2011, 3)

Improving interoperability workflows in CAD and BIM software

As mentioned above, the major development of CAD software has been primarily focusing on enhancing modelling features and techniques to let the user manipulate as freely as possible geometrical shapes. Therefore, GUI development in CAD software has been more concerned about the way users model the geometry itself (Pena De Leon, 2014, 245), rather than the manner users can attach custom data specifications to that same geometry. This particular divergence of concerns has been already observed 30 years ago by (Howard et al., 1989):

“Existing CAD systems tend to focus on the representation of the design solution and not on the development of the solution.” (Howard et al., 1989, 24)

Furthermore, the author criticized the misuse of CAD packages, that were replicating analogue processes instead of creating new methods to enhance collaborative practices using digital means:

“Over the past 20 years, computers have become very widely employed by specialists in all phases of the AEC process, and yet the AEC industry is still exchanging data and design decisions as it did a century ago, with paper drawings and reports. Introducing computers to the design process has changed the means of generating the paper, but it has not fundamentally changed the methods of sharing data across organizational boundaries. It is not uncommon to find that in one office a designer uses a powerful computer-aided design and drafting (CADD) package to produce a project drawing and then, in another office, a construction estimator uses a digitizer to put the information from that same drawing back into a computer (in effect, unCADD).” (Howard et al., 1989, 20)

⁵<https://www.rhino3d.com/opennurbs>

This resonates with the former distinction made by Kostas Terzidis between “computation” and “computerization” (Terzidis, 2003):

“While computation is the procedure of calculating, i.e., determining something by mathematical or logical methods, computerization is the act of entering, processing, or storing information in a computer or a computer system.” (Terzidis, 2003, 69)

While “computerization” simply digitized analogue processes, “computation” is extending “computerization” and explores complexities through algorithmic procedures that extend human thinking:

“Digitization is the conversion of analogue information into digital information. Computerization, by definition, involves digitization. This is not necessarily the case with computation. Because many mental processes can be analysed, codified, systematized or even synthesized without the use of computers, computational methods do not have to involve digitization. Nonetheless, their implementation on a computer system allows explorations of complexities that extend the limits of human prediction.” (Terzidis, 2003, 69)

In the realm of interoperability workflows in CAD and BIM software, a parallel contrast can be observed between file-sharing processes and the more recent file-less, data streaming paradigm that shapes most of today’s video or music sharing websites.

As mentioned earlier, the second type of CAD software packages enable the management building data sets at late stages in the design process, although without the ability to model complex geometries. The BIM software Revit will serve here as a main example to illustrate this category.

Autodesk’s Revit is a BIM software developed since 2000 allowing all stakeholders involved in the design process of an architectural project to store building data and access it through all the phases, from conception to construction. As explained above, this particular type of software offers poor options to model geometry at the early stage in the design process. Therefore architects are usually using such type software only at late stages, once all the geometry processing has been done beforehand. Computational Design Specialists with expertise about BIM also took a step forward by improving the interoperability experience between BIM and modelling software platform. In an interview given for BIM+, Eckart Schwedtfeger explains how he developed custom tools to improve the workflow between Rhino3D and Revit:

“[...] Revit didn’t support custom freeform shapes efficiently enough. Importing them from our principle design tool Rhino involves about a dozen manual steps, which would have been enormously time consuming to carry out for the many thousands of freeform elements and frequent design changes. I looked at the application programming interface and realised that it was possible to program these steps automatically, so I developed a plug-in that would do the importing with a press of a button. This plug-in is referred to as ZHA BIM and is still heavily used by all BIM projects. It can also seamlessly transfer metadata between platforms, for example details on the material and parameters for characteristics like panel type, finish, colour etc.” (Stephen, 2018)

Section 2.4.2 will describe in more details this particular case study of BIM workflows at Zaha Hadid Architects.

At the end of 2015, Autodesk announced the creation of a new cloud-based platform: the Autodesk’s Forge Platform. Through Forge, Autodesk aims at enhancing interoperability, connecting teams, and building new workflows between the existing Autodesk’s software environments as well as other external applications. With access to a number of open APIs, expert users and developers are able to develop custom workflows for AEC companies, in order to respond to their specific needs and better communicate with external trades.

Similarly to the Forge Platform, McNeel & Associates has also been looking at developing specific strategies in order to facilitate the development of custom application by third party developers and expert users who look at improving data exchange and interoperability with Rhino3D. This has been

the case with the development of Rhino.Inside technology which allows Rhino and Grasshopper to be embedded within other products (e.g. Revit, AutoCAD, Solidworks, Photoshop, Excel, etc.). Rhino.Inside would allow to start Rhino and Grasshopper as a plug-in within another product, and call directly the host's native API (such as Autodesk's Revit) from a Grasshopper or Rhino plug-in. McNeel has also recently released its Rhino Compute Service, an *"experimental project which allows access to the Rhino's geometry library through a stateless REST API via the McNeel cloud. Compute is based on the Rhino.Inside™ technology to embed Rhino advanced geometry calculation inside an online web service."*⁶ Rhino Compute therefore aims at decentralizing the calculations traditionally operated by the software within a local machine by enabling those same calculations to be performed on any other machines connected through the REST API.

Existing tree interfaces & hierarchical paradigms for classifying and manipulating objects

Today's state of the art in CAD software's UIs for manipulating building data across multiple scales makes almost systematically use of tree-like data structures. The latter would either act as a static hierarchical storage, geometrical database (e.g. Rhino3D's LayerTable as illustrated in fig. 2.6, left) in which geometries relate to each other by being simply informed about their parent or child layers (in which other geometrical objects can be found), or as a dynamic Directed Acyclic Graph (e.g. CATIA's Tree Directory as illustrated in fig. 2.6, center) in which geometrical relationships (e.g. boolean operations, intersections, transformation, etc.) persist between objects and propagate along the tree, from the root to the leaves. The Object-Oriented IFC standard also organizes its classes in a hierarchical fashion, in which each IFC object depends on another parent object and/or possesses one or multiple child objects (as illustrated in fig. 2.6, right). Other alternatives for the hierarchical and topological representations of architectural spaces are also being developed. The current Non-Manifold Topology research project led by Robert Aish is specifically looking at linking objects based on their topological relationships to further operate queries across the geometrical data structure: *"Because Non-Manifold Topology maintains topological consistency, a user can query these cellular spaces and surfaces regarding their topological data and thus conduct various analyses."* (Jabi et al., 2018).

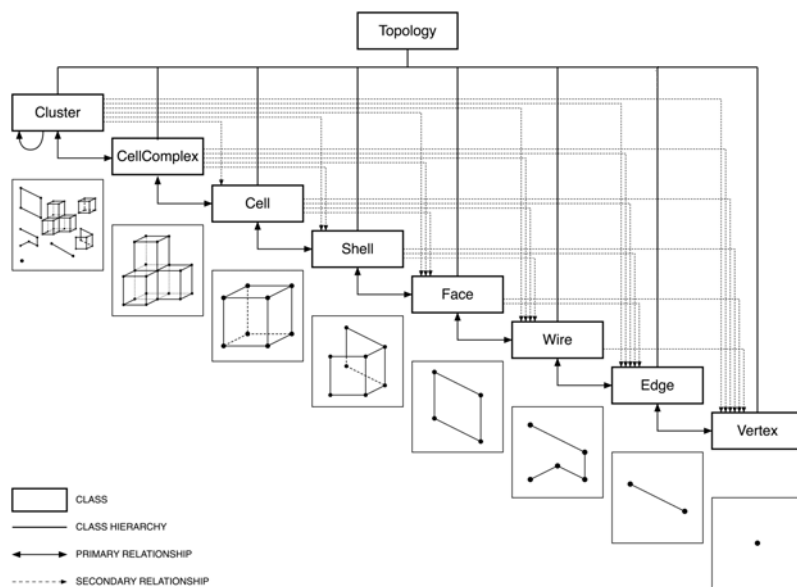


Figure 2.5: Non-manifold topology class hierarchy (Jabi et al., 2018)

Most of these tree-like UIs usually includes some basic UI features, allowing the designer to select specific branches (layers and/or sublayers) within the tree, show/hide the corresponding contained objects, lock specific geometries, navigate across the tree by folding/unfolding branches, etc. Although useful, these existing functions can be sometimes limited, especially when the tree grows

⁶<https://www.rhino3d.com/compute>

and reach a certain scale and intricacy level that is difficult to comprehensively grasp as a whole for the user. For example, unpredicted implicit relationships and cross linked adjacencies might exist between the leaves of the data structure. In such cases, the user would definitely need a more flexible and intuitive UX, in order to search, query, filter and access objects across all the scales offered by the tree. Therefore, better interfaces could be developed to enhance the UX during late stages of a large-scale and complex architectural project.

The next subsection will look at existing UI and UX concepts that are emerging outside the realm of AEC within other disciplines that also need to handle very complex data sets.

Visualizing and exploring complex data sets across scales

Within the interdisciplinary field of genomics that focuses on the mapping and editing of genomes, very large and complex data sets are handled by the genomicists in order to fully understand the manipulated DNA sets. Bioinformatics – a research field that specifically focuses on the development of software tools to better understand biological data – is used here to sequence and analyze the structure of the genomes. It has been pointed out that the manipulation of those complex data sets also need to be combined with efficient data visualization methods and tools, helping the scientist to better understand the former:

“Our growing ability to collect enormous amounts of sequence information to support such studies is arguably out-pacing the rate at which we devise new methods to store, process, analyze, and visualize these data. Any new approaches in data modeling and analysis need to be accompanied with corresponding innovations in the visualization of these data. To mitigate the inherent difficulties in detecting, filtering, and classifying patterns within large data sets, we require instructive and clear visualizations that (1) adapt to the density and dynamic range of the data, (2) maintain complexity and detail in the data, and (3) scale well without sacrificing clarity and specificity.” ([Krzywinski et al., 2009](#), 1639)

From these initial observations, the very popular software tool *Circos* has been developed, enabling a multi-scalar approach to the data visualization of sequenced genomes:

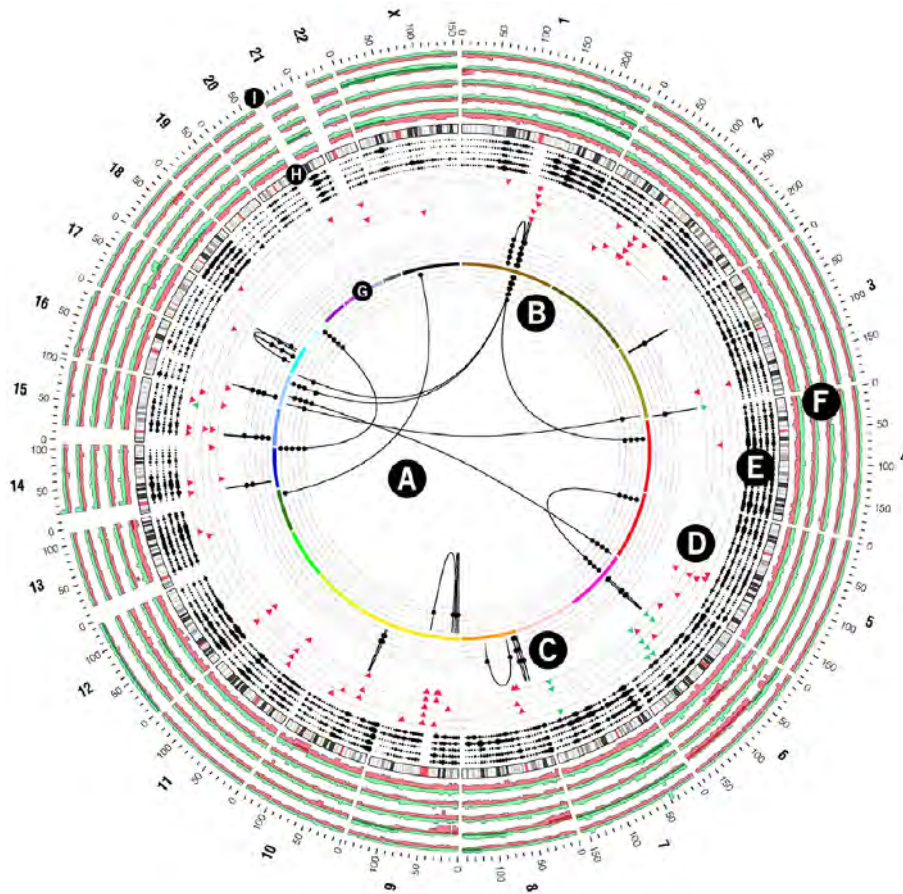
“A unique aspect of Circos is its ability to adjust the global magnification for each ideogram and, furthermore, to smoothly vary the magnification within a region. This kind of local scale adjustment is effective to emphasize fine structure of data in a region while preserving context.” ([Krzywinski et al., 2009](#), 1642)

More advanced research has been undertaken outside the realm of bioinformatics to enhance visualization of intricate, hierarchical data sets that also contain “adjacencies” and implicit, inter-relationships between the leaves of their respective data structure ([Holten, 2006](#)).

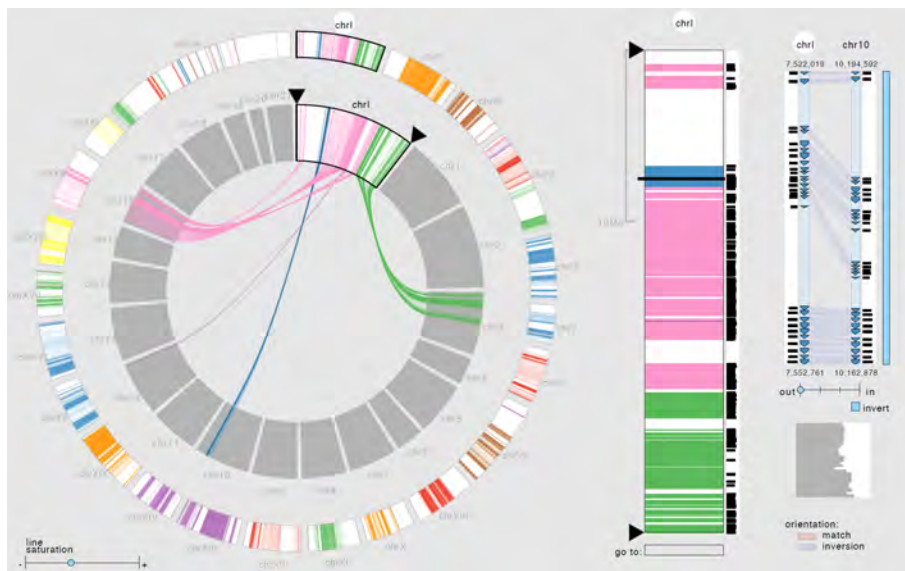
Even though *Circos* has proven to be very useful for representing complex information containing implicit relationships (Figure 2.8a), alternative methods have been investigated and developed to enhance the [UX](#) by improving not only the visualization, but also the interaction with large and complex data sets. *MizBee* (Figure 2.8b) has contributed to enrich such experience by providing “[...] a multiscale syntenic browser with the unique property of providing interactive side-by-side views of the data across the range of scales supporting exploration of all of these relationship types.” ([Meyer et al., 2009](#), 897). This multi-scalar interface enables the display of “[...] linked views across the genome, chromosome, and block levels, allowing the user to maintain context across all of these levels when exploring conserved syntenic data.” ([Meyer et al., 2009](#), 898).

It is quite surprising to observe the slow and belated development of a property-based document management system in the historical development of both Operating System’s and CAD’s [GUIs](#), especially when the [AEC](#) community desperately need such a paradigm shift in order to manage geometrical complexities throughout the whole design process (see section 5.1).

The next section is focusing more particularly in existing challenges faced by the [AEC](#) industry, both in term of interoperability issues and rising complications during the scaling of complex information.



(a) "A whole-genome view of structural changes in five follicular lymphoma tumor samples observed using restriction digest fingerprints and Affymetrix Mapping 500k arrays." (Krzywinski et al., 2009)



(b) MizBee: a multiscale synteny browser (Meyer et al., 2009).

Figure 2.8: State of the art in the development of custom data visualization tools and **GUIs** for the realms of bioinformatics and genomics. Those interfaces attempt to help the researcher in better understanding its manipulated datasets.

2.2 Applied Parametric Modelling in AEC: Current Problems and Existing Solutions

The previous section outlined precedent and current challenges related to interoperability and the scaling of complexity within Operating Systems and CAD software environments. It helped to draft initial insights for improving such platforms by enhancing with appropriate UIs and UX means. Those are necessary to the set up a Multi-Scalar Modelling framework enabling the transfer of data rich objects through a network of models spread across multiple platforms used by different stakeholders during the design process. In this section, starting from a critique of DAG based workflows, the current problems faced by the AEC industry related to both interoperability and the scaling of complexity are discussed. It is finally proposed to hybridize workflows by merging both integrative pipelines with segregated strategies in order to overcome the previously stated issues.

Because CAD Software development aims to cover the needs of a large span of many different industries, its focus has primarily remained on making accessible to its users versatile ways of modelling objects. However, the design of highly complex objects (such as cars, ships, air-planes or infrastructure) shifts the focus from pure modelling concerns to data management and interoperability challenges. In the aerospace and car industry, those are usually tackled by integrating all complexity into one single environment, where all elements and sub-elements are ordered into a neatly organized directory tree structure that maintains and keeps track of the parametric relationships between the objects. In the aerospace industry, one of the first software to handle such complexity is CATIA, developed by Dassault Systèmes, which offers the possibility to segregate the design process into different parts (.CATpart), referencing to the same master model. Contrary to the aerospace or the car industry, the AEC sector does not create mass-customizable objects, but unique architectures, a fact that prevents the person (the *Computational Design Specialist* or *BIM Manager*) in charge of modelling the whole process from spending as much time on a building model as someone would do on an air-plane or car model. Fabian Scheurer specifically pointed out such paradox:

"[...] the one-off design of most buildings makes it a very different procedural problem from the mass production of a jet or car, where the time and effort required to create a comprehensive digital model can be justified through its repeated use." (Scheurer, 2012, 130)

Furthermore, Rick Smith indicates that the typical design process of an airplane differs from an ideal building's design workflow. The former is in fact more fragmented than the latter, and full parametric design does not occur:

"The design process of an airplane is different enough from a building [...]. A plane is broken down into design teams. Each team is responsible for its components and how they interface to the adjacent parts. For example, the landing gear team is not concerned about other aspect of the plane other than what affects the landing gear and where it attaches to the plane. Therefore the parametric models are strictly for much smaller assemblies and subassemblies, as in a landing gear assembly. There is no concept for parametrically controlling an entire airplane. It is simply not done." (Smith, 2007, 5)

Because the Return On Investment (ROI) is much higher for mass-customizable design products than for buildings (and also because of the fact that CATIA is a very expensive product), it seems counter-intuitive to either invest in one of the most expensive software for managing the modelling of complex data or to develop integrative design workflows at an architectural scale.

2.2.1 Limitations of DAG-based and integrative workflows in AEC

In mathematics and computer science, a DAG is a finite directed graph with no directed cycles. In the architecture field, DAGs are often the basis of computational modelling tools and serve to seamlessly establish parametric relationships between three-dimensional objects. The setting up of a DAG-based

workflow often implies a zero tolerance digital chain that has no room for imperfection or radical topological change: *“Parametric design tools are practical but they effectively rely on the definition of specific base infrastructures that are resistant to change during the design process and over the lifetime of a model”* (Tamke et al., 2013). Neil Denari pointed out that this paradigm is incompatible with the reality of the building industry:

“In industrial design, as a point of comparison, the development from the model (the unique thing or prototype) to the serial artifact (the mass-produced thing) assumes an escalation in precision (zero tolerance) and the production of maximum fitness, i.e. that it is fit to be produced and fully resolved within the dual ecologies of use-value and material production. But as a building, being more hand-made than machine-made, is always relegated to the model and realized only once because of its size, location and, most importantly, because of its construction systems, it cannot, in the terms applied to industrial design, reach the level of precision of the serial artifact.” (Denari, 2014, 31)

This observation corresponds to a similar statement made by Jonathan Brandt, criticizing the naive belief of architects in the possibility of setting up deterministic workflows that would perfectly simulate the built reality:

“Determinism in design is fueled either by ego or naiveté. The assumption that an architect, even when augmented with expert consultants and cutting edge technologies, can predict and accommodate all the nuances that occur during the construction and occupation of his or her design, finds evident error. If this were the case, then tolerances would not exist and the intent forever persist; the bolt would always fit in the hole and the tenants would never repaint the walls in a different color.” (Brandt, 2012)

The unscalable and unsustainable nature of Directed Acyclic Graphs (DAGs)

When the DAG scales up in complexity and reaches a certain size, it becomes difficult to manage it, mostly because of its high degree of Cyclomatic Complexity (CC). The Cyclomatic Complexity can be calculated using Thomas McCabe’s formula: $CC(G) = E - N + 2P$, where G = the graph, E = the number of edges within the graph G , N = the number of nodes within the graph G , P = the number of connected components, or independent graphs (parts) within the graph G (McCabe, 1976). Citing McCabe, Daniel Davis (former Director of Research at WeWork and author of the PhD thesis *“Modelled on Software Engineering”*) highlighted the fact that any code with a CC greater than 10 should be restructured:

“Programmers have been required to calculate complexity as they create software modules. When the complexity exceeded 10 they had to either recognize and modularize sub-functions or redo the software.” (McCabe, 1976)

Instead of liberating the expert-user who needs to implement intricate workflows that span from early to late design stages, it *incarcerates* him/her within an overly defined design space, where any radical topological change leads to the ultimate rewriting of the code in which so much effort has been placed. This issue resonates with another statement made by Phil Bernstein (former Vice President Strategic Industry Relations at Autodesk and Associate Dean at Yale School of Architecture) who warned about the mismanagement of continuous workflows:

“If parametric models force designers to explicitly create rule sets, scripts and relationships, then it is also true that, unmanaged, these connections release a cascade of unintended consequences that are liable to become incomprehensible.” (Bernstein, 2012)

It seems therefore that implementing a full DAG workflow from early design stage to late stages is a wicked problem in itself. Therefore, DAG pipelines should be used partially instead of attempting to encapsulate the whole process within one single integrative workflow. The following section criticizes such integrative approach further by analyzing from a software perspective at which particular points full DAG based workflows fail at.

2.2.2 Concerns in parametric modelling: the paradoxical inflexibility of integrative design workflows

The usually tight time-line framing a large-scale and complex architectural project has an important influence on how the team manages, handles and delivers the latter. Hard deadlines require high productivity preventing therefore the possibility to organize all computational processes within a neatly DAG workflow. Indeed, trying to encapsulate the full complexity of the architectural project within one single pipeline would take more time than completing the project itself, especially in an industry that is fragmented through many different software platforms used by a multitude of stakeholders. Therefore, the different trades are currently establishing with their own custom methods, interoperability tools, quick shortcuts and manual interventions in the design process, producing redundant models and files in order to communicate with others and across different software environments.

The five points of parametric modelling

Beyond economical reasons, the limitations are also technological, a dimension well highlighted by Daniel Davis in its thesis *Modelled on Software Engineering : Flexible Parametric Models in the Practice of Architecture* (Davis, 2013), in which he comments and elaborates on the *five major shortcomings with parametric modelling* identified by Rick Smith (a former CAD technician for Lockheed and CEO of Virtual Build Technologies) in a white paper called *Technical Notes from experiences and studies in using Parametric and BIM architectural software* (Smith, 2007). Those five points are the following:

- Parametric models require a degree of front-loading.
- Anticipating flexibility can be difficult.
- Major changes break parametric models.
- Changes can be hard to see visually.
- Reusing and sharing models is problematic.

Based on his own experience, Smith clearly stated that the idea of setting-up an entire parametric model that could solve all problems through all the scales is more of an utopia than reality (and - as he mentions - serves a certain *marketing hype*) if one really takes into account the current technological limitations:

"The concept of creating a fully parametric digital model of an entire project to accommodate variable changes is just not efficient at this time with the current state of technology. This is theory and hyperbole, especially when trying to accomplish such an all encompassing function on a 32-bit PC." (Smith, 2007)

Even though this statement was written more than a decade ago, it remains very actual, as leading contemporary architecture and consultancy practices still make use of a segregated network of models (developed in detail in section 2.3) rather than integrative, seamless design workflows. The computers processors may all have upgraded to 64-bit, but the problem remains essentially the same, and might be more of an epistemological, methodological nature than purely technological. Indeed, even the automotive industry witnessed similar issues and warned about the risk of integrating too many parametric relationships between objects:

"Because product models contain numerous geometrical characteristics that could be connected, the complexity of parametric-associative CAD models increases rapidly, which leads to a decrease in their stability in the case of modifications. To avoid complexity problems, the implementation of parent-children relations should be planned carefully, and the geometrical linkage between components should be kept to a reasonable level. The lower the number of interlinks, the lower the model complexity, which normally leads to a higher model stability." (Hirz et al., 2013, 302)

Therefore, some basic rules have been defined in order to limit the number of parametric associativity in some cases:

“In automotive engineering, some guidelines prescribe the application of non-parametric geometry in the case of geometry-based interlinks [...] between different parts in order to reduce the model complexity and to avoid unwanted dependencies in complex model structures. In such cases, all external geometry references are performed with no parametric-associative linkage.” (Hirz et al., 2013, 302)

Those statements (coming from different industry-related domains) echoes with has been previously discussed about the non-realistic and nearly impossible idea of organizing the whole computational workflow within a single DAG. The following section aims to demonstrate that maintaining a full DAG workflow from early to late stage is not sustainable at an architectural scale.

2.2.3 Dealing with changes during the design process: developing strategies for early design and late stages

It has been highlighted above that one of Rick Smith’s main observations is that major changes break parametric models. One of the classical proposal to solve such issue is to concentrate the design effort as early as possible in the design process, in order to avoid changes (and subsequent costs), that would “break” the supposedly seamless digital design chain. Davis traced back to 1976 the first instances of this *earth-shattering revelation*⁸ (the idea that the cost of design changes increases proportionally with the time-line of the project), both in AEC (Paulson, 1976) and computer science (Boehm, 1976). In the construction industry, this notion is mostly known today as the “MacLeamy’s Curve”, a revisited version drawn by Patrick MacLeamy in 2004, CEO of HOK at that time. This curve is now very often used and showcased by the promoters of BIM and Integrated Project Delivery (IPD), when the related key concepts are being introduced.

Davis proposed an alternative to the MacLeamy’s curve by suggesting that the increase of cost of design changes could be delayed much later in the project until the start of the tender phase (Davis, 2013, 208). Technically, this could be achieved mainly by making more effort in structuring and modularizing the computational design workflow into different clusters (Davis, 2013, 126-149), a general concept that consultancy companies such as Front and Design-to-Production naturally adopted and pushed even further within their respective practice by segregating their design processes into different models (2.3).

The decision from both Front and Design-to-Production to finally segregate the design process into different parts and models actually goes against Davis first suggestions to keep geometrical linkages and flexibility until late stages, using clustering and partitioning strategies. Ultimately, even those are unsustainable when the design is too complex and too many trades are involved in the project. By segregating the process, the user is finally free to operate classification changes and introduce features (such as names and attributes) to objects at any time during the generation of the building’s data.

Inevitable late changes

The following argument emits the hypothesis that late changes will in any case happen in the design process. In such case, neither the initial assumption made by Paulson and McLeamy (that is to shift the design effort at early design stages), nor the advice of a necessary high degree of front-loading by Rick Smith – “A designer might say I want to move and twist this wall, but you did not foresee that move and there is no parameter to accommodate the change. It then unravels your [parametric model]. Many times you will have to start all over again.” (Smith, 2007, 71) – or Mark Burry – “The designer has to become skilled in anticipating the unintentional over (or under) constraint as well as the inadvertent fixing of a condition by a decision made much earlier in the parametric design process.” (Burry, 1996, 71) – or the proposition of Daniel Davis in modularizing digital design workflows would be the final and only answers to both prevent costs and enable flexibility at late stages. This hypothesis supporting the idea that late changes are in most cases inevitable can be actually verified and supported by many different AEC practices, such as Front and Design-to-Production who developed their own segregated strategies to allow more flexibility and manual intervention during late stages (see section 2.3).

⁸<http://www.danieldavis.com/macleamy/>

As well as attempting to shift all efforts at early stages in the purpose of preventing late changes, it should also be suggested that late changes cannot be entirely avoided, and the need for segregated processes and manual interventions should therefore be taken for granted, and embraced. Thus, specific tools, workflows and strategies should be developed to better manage late changes and enable a more seamless workflow towards the end of the project. Figure 2.9 illustrates such argument by highlighting that the focus should not only be placed at early stages but at late stages as well.

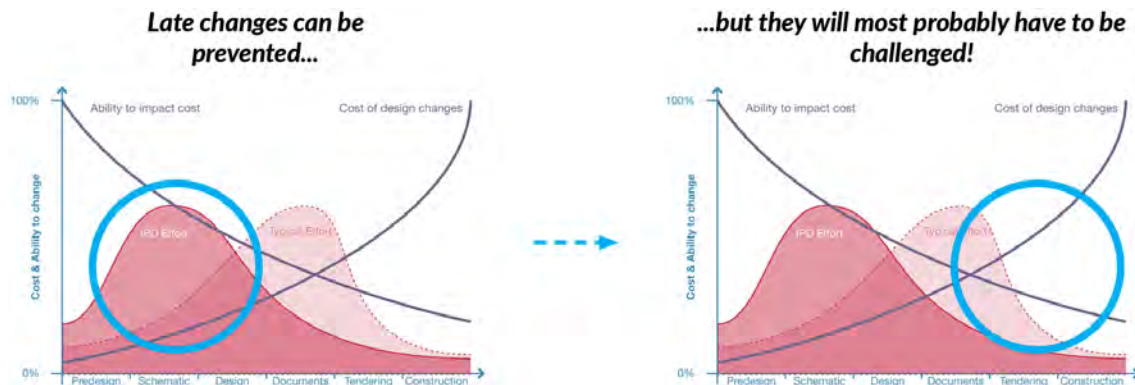


Figure 2.9: The original figure drawn by (Davis, 2013) uses the Paulson and MacLeamy's curve to emphasize the need for shifting the focus on developing strategies from early stage design to late (design) stages. This curve represents the existing correlation between the increase of cost of design changes with the decrease of design flexibility during the design process. To tackle such issue, the common argument is to shift the effort at early stage in the design process (left). However, even if the design effort should still happen at the beginning of the project, late changes and their subsequent issues will most probably be tackled in most cases. Tools and workflows should therefore be developed in order to manage data in a more meaningful way at late stages as well (right). Thus, the focus and design effort should be placed at both ends.

2.2.4 Towards hybrid workflows: integrated and segregated strategies

The above does not suggest that DAGs need to be completely banned from the design process, but instead should be used locally and partially, the whole workflow requiring to be fragmented (or decentralized) so manual interventions become possible. Rick Smith also drew a similar conclusion in *Technical Notes from experiences and studies in using Parametric and BIM architectural software*, where he argues that parametric modelling can be used either at the very beginning of the design process during massing stages, or locally at late stage to solve specific fabrication problems: “We believe parametrically controlled digital modelling has its place in early conceptual design or in the massing stage of a project and also at the back end for manufacturing and fabrication, when warranted or efficient.” (Smith, 2007).

In the essay entitled “Lost in Parameter Space?”, Fabian Scheurer recognized the benefits but also the limitations of integrative computational design workflows, some of which being the impossibility of operating any sort of manual intervention, crucial to communicate data externally:

“Still, designers should be aware that the resulting geometry at every stage is volatile and immediately dependent on the input. While this eliminates the risk of update anomalies it also suppresses the possibility of deliberate redundancies or manual intervention.” (Scheurer and Stehling, 2011)

The need for a hybridization between automation and manual intervention has also been reported by Alexander Peña de Leon in one of his concluding remarks of his thesis “Separation of Concerns: strategies for complex parametric design modelling”, in which he differentiates the “Direct Modelling” approach (relying on the software-related tacit knowledge of the user) and explicit automation strategies:

“[...] a paradox between the need for more automated ways of modelling with the need for more “Direct Modelling” ways of interacting with models. They are contradictory

because automation aims at reducing “Direct Modelling” interaction. [...] we need better ways to integrate automation in design modelling while simultaneously we need more tacit interaction through “Direct Modelling” with our models in order to achieve a greater flexibility in design modelling.” (Pena De Leon, 2014, 245)

Furthermore, similarly to Smith’s observations (quoted above) Peña de Leon specified that automated workflows would better fit very early design strategies and very late stages that relate to fabrication concerns: *“For procedural and logical design problems, for example issues with the “pre-geometric” and “post-geometric” aspects of design, designers can use the algorithmic capabilities of the external automated workflow.”*, whereas *“[...] for spatial problems designers can use the direct modelling 3D interfaces provided by most 3D modelling software. [...] Designers are therefore enabled to use their tacit knowledge in solving complex spatial assembly puzzles using the 3D viewports of standard off-the-shelf parametric software.”* The dichotomy between generative methods for design explorations (allowing for interface flexibility and feedback) and the zero-tolerance of mechanical processes of fabrication has also been highlighted by Tobias Nolte in *Crowdsourcing Embedded Intelligence*:

“In the past, generative methods have been largely used for design explorations, often assuming that implicit geometry or a very few rules of thumb suffice for validating design feasibility. But in fact the mechanical processes of fabrication often have decisive impacts on the design geometry itself, particularly at the detail level. Validation on this level often requires dozens of parameters to be analysed for each assembly piece, even for apparently simple systems. Systems intended for design exploration alone are not sufficiently robust, reliable or simply extensible to give the detailed feedback required in these situations.” (Nolte and Witt, 2014, 86)

As a reaction to all these realizations, remarks and conclusions, computational design specialists and consultancy practices have been developing their own internal methods to better control their generated data (and metadata) through all stages, in order to enhance computational design workflows and also to enable manual intervention when required. Such practices will be illustrated from two different perspectives: the scaling and managing of complex data models and the enhancement of interoperability.

2.3 Scaling Complexity: Separation of Concerns

Following the findings from above, and in order to scale up in complexity, it seems quite essential to separate the concerns:

“It is what I sometimes have called “the separation of concerns” [...]. This is what I mean by “focussing one’s attention upon some aspect”: it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect’s point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.” (Dijkstra, 1982)

This approach resonates with Christopher Alexander’s statement, arguing that a problem of a high degree of complexity should be broken down into smaller parts to solve: “[...] the complexity of the problem will defeat us unless we find a simple way of writing it down, which lets us break it into smaller problems.” (Alexander, 1973, 3). The Separation of Concerns concept has been revisited and applied by Alexander Peña de León within the design process (Pena De Leon, 2014), to segregate, fragment the latter and allow the user to intervene manually through all stages. To support this argument, different workflow strategies and computational design processes conceptualized or developed by Front, Design-to-Production, BuroHappold and Oliver Tessman (Tessmann, 2008) will be described in the next paragraphs.

2.3.1 Staged Models and Activity Networks

2.3.1.1 Front's Building Information Generation

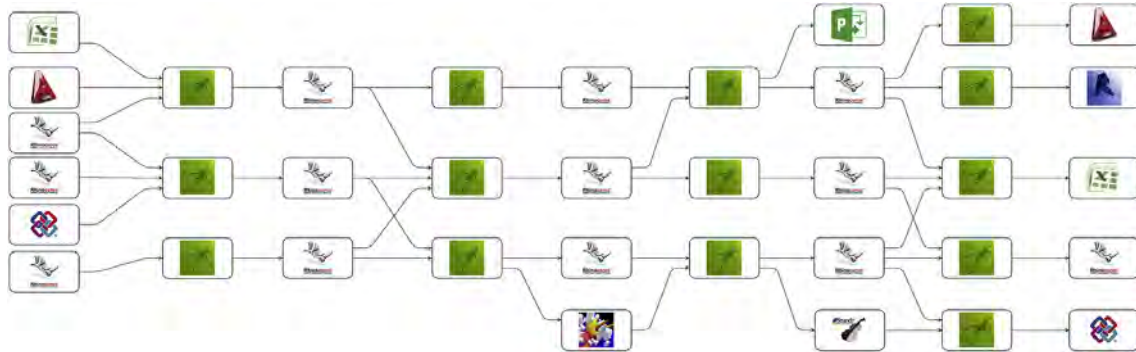


Figure 2.10: Project ecosystem of staged models and generating logic (image courtesy of Front).

During the realization of the City of Dreams Casino Hotel in Macau conceived by Zaha Hadid Architects, the consultancy practice Front developed a modelling strategy called *Building Information Generation* (Van Der Heijden et al., 2015) enabling parallel generation of information and attributes necessary for further fabrication. In order to manage attributes and assign user data to the processed geometrical objects, Front developed in-house a custom Rhino3D Plug-in *Elefront*. The whole modelling process consisted of a strategic alternation between the generation of objects in Grasshopper and their subsequent storage and classification within *staged* Rhino3D models, where the geometry is “frozen” (or “baked”), thus devoid of any geometrical linkage. From these static, fixed geometrical objects were generated further information through a next iteration of Grasshopper3D sessions (in which parametric linkage was kept). Such process repeated itself until the last level of detail was modeled and ready for manufacture, fabrication and assembly (Figure 2.10): Ramon van der Heijden, former Computational Design Specialist at Front Inc and Digital Design and Services Leader and Associate Director at Arup calls this process “staging” :

“The process of creating models that use information from combined results of previously generated models is called staging. At each stage the previous input and generating logic is preserved and available for other stages in the project ecosystem. Each stage creates a model that meets a specific requirement.” (Van Der Heijden et al., 2015, 418-419)

Similarly to the “Separation of Concerns” mentioned above, the author evokes a “discretization of logics”, where many different small files are individually processed, instead of embedding intelligence within one single very large model. Each small file can be manually triggered and re-computed when a change occurs up-front, allowing therefore data propagation through the whole digital chain:

“This collaborative environment fosters the discretization of logics and models that serves to decrease the size and complexity of any particular model or logic, and allows for less complex logics that can be more easily managed by a person, for smaller files that can be processed much faster than a single large file, and for outcomes of each stage that can be used as inputs for different processes and analysis. A change anywhere in the process can be propagated by re-running the subsequent stages.” (Van Der Heijden et al., 2015, 419)

The whole computational design process was therefore fragmented into several parts, allowing manual interventions, proof checking and potential corrections from the expert user, before generating a new set of data based on the previous one, and thus through all scales until the highest modelling resolution.

2.3.1.2 Collaborative dependencies

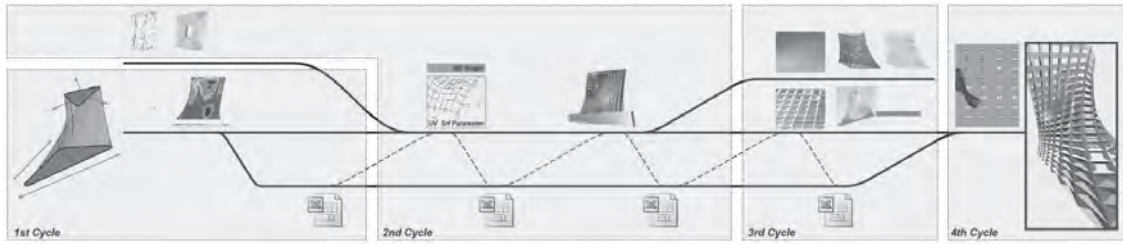


Figure 2.11: Diagram showing the development cycles and collaborative dependencies during the design of the augmentedFRAME architectural project, a competition entry by Oliver Tessmann in collaboration with Mirco Becker, Asko Fromm and Gregor Zimmermann. (Tessmann, 2008, 125)

The approach taken by Front Inc. to rationalize the design process into segregated parts is not new and has also been explored previously by other practices. For example, the project “augmentedFRAME” undertaken by Oliver Tessmann, Mirco Becker, Asko Fromm and Gregor Zimmermann, clustered the whole design processes into different “development cycles” (Figure 2.11). Each cycle focused on one particular task in the design process, involving a specific set of tools and software platforms. When the task is performed, the data flows towards a next cycle, and so on, thus until completion of the design project:

“The design process was comprised of a series of feedback cycles: Finite Element Analysis (FEA) improving the shape and visualizes the flow of forces, geometric analysis, structural space frame analysis and analysis of vistas. Each of these stages allowed going back to any previous stage without losing information. The final design proposal embodied the synchronization of these chosen parameters represented in a complete dataset.” (Tessmann, 2008, 125-126)

2.3.1.3 Bauke de Vries' Activity Network

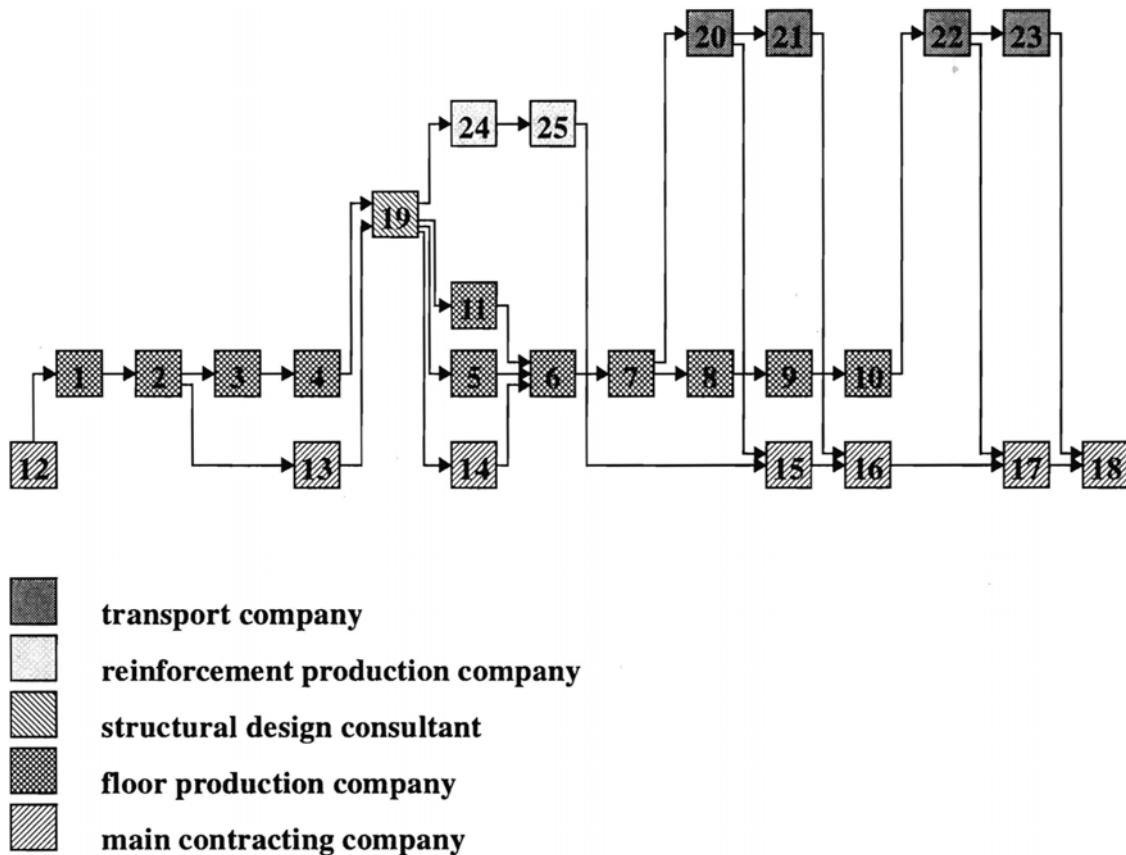


Figure 2.12: The activity network modelled as a PERT (Program Evaluation and Review Technique) network

The staged modelling strategy developed at Front Inc. and the development cycles and collaborative dependencies of the “*augmentedFRAME*” project (Tessmann, 2008, 125) have been formerly conceptualized in 1995 by Bauke de Vries in “*Message Development in the Building Process*” through an “activity network” (Figure 2.12). Bauke de Vries defines the activity network as follows:

*“An activity network shows the information flow between activities. The activity network is manually deduced from the list of messages that is generated from the first order agreement. All activities are set to take equal time to execute and each activity will be started by the participant that initiated the project. **The most important property of the network is that an activity can only start executing if all input channels contain the required information.** The squares in figure 2.12 symbolize the activities. [...] The arrows symbolize the channels through which the information flows from one activity to another.” (de Vries, 1995)*

The activity network therefore acts as simulation model of a building project which can be executed, simulating the information exchange process that takes place between the different involved stakeholders. The activity network highlights which processes are taking place concurrently or sequentially, and informs about the activities that are interdependent and the ones that needs to be re-executed if already accepted information is changed.

2.3.2 Design-To-Production's Digital Craftsmanship

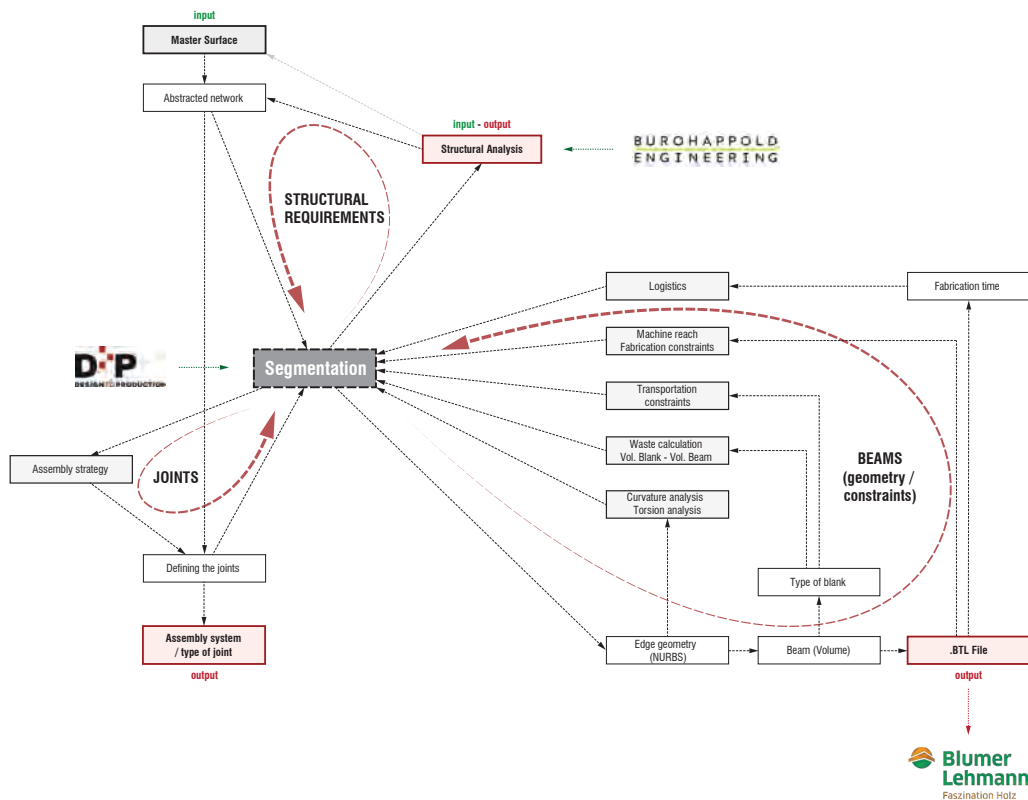
Design-to-Production has developed very strong expertise on the segmentation of free-form timber structures (e.g. the Centre Pompidou-Metz and the Haesley Nine Bridges Country Club by Shigeru Ban Architects) and their complex digital workflow from conception to fabrication (Scheurer et al., 2013). Generally, the company finds itself right in the middle of the industrial chain, in which they must take into account many different factors and constraints coming from the other trades involved in the same project. Those diverge from structural requirements prescribed by the engineer to fabrication constraints (machine reach, transportation, waste, etc.) impacting the contractor's production workflow (Figure 2.13a). This negotiation results in the segmentation of the complex building by Design-to-Production.

Working very closely with timber manufacturers and architects specialized in the realization of complex wooden construction for over a decade, Design-to-Production has developed a very strong expertise in modeling the characteristics of glue-laminated timber elements through all scales, from the single laminate to the assembly sequence between multiple building components. The company also developed very specific workflows and pipelines, not only for their own modelling tasks but also to clearly communicate data to the other trades, each one of them requiring specific file formats for further processing and fabrication.

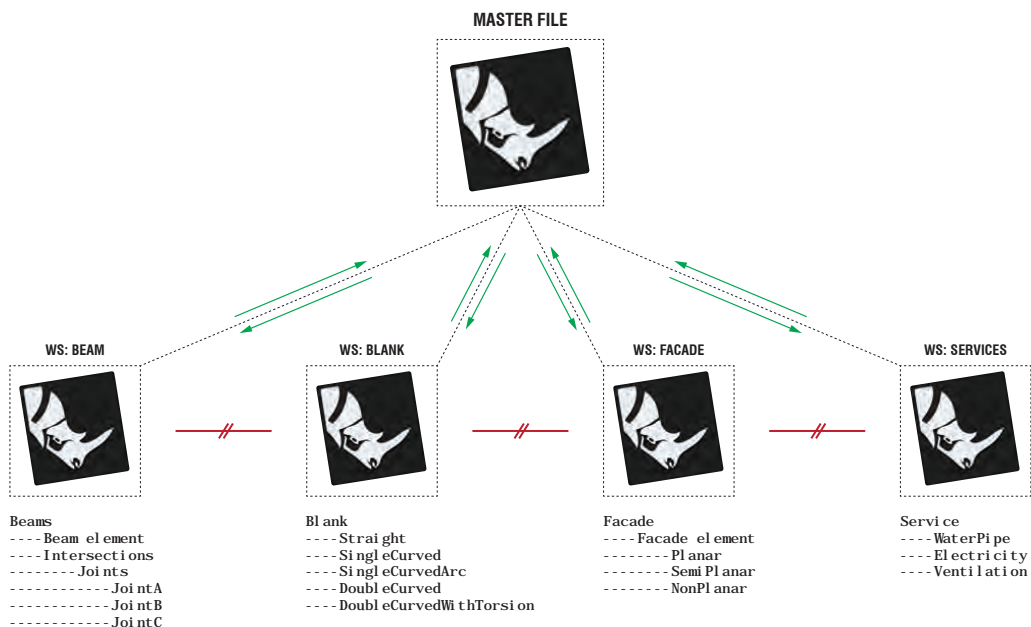
Design-to-Production also makes an effort to generalize most of the scripts and pipelines they are writing for their projects by storing and classifying them within their custom library. It enables the capability to recall available functions within new projects that deal with similar building components requiring the same sort of modeling processes. Thus, the company is able to deepen and reuse specific knowledge related to the discretization and data generation for free-form timber structures. Through multiple real-world examples, the different problems and challenges that emerge both during conception and production will be explicated and specific solutions will be proposed both on a theoretical and practical level.

During the post-tender phase of those projects, Design-to-Production generally employs modelling methods enabling the storage of information in parallel to the generated geometry. The company organizes its objects through the Rhino3D's LayerTable, enabling both its storage and classification. Similar to a directory tree structure, a LayerTable is organized into different levels (or depths) that communicate from a root layer to all its successor sub-layers. Generally, both the root-layer (containing the label of a component) and the leaf-layers (containing its geometries) are defined through the protocol generating the information. The intermediate layer levels are identified by the component's name and mainly serve to structure the information in order to make it more human-readable, defined by the expert user him/herself. Therefore, the layer tree is independent from the "semantics" of the component and just a UI-feature. Depending on the nature of the component (being a beam, an opening, a facade element, etc.), it will be stored within a specific submodel (or worksession) that refers to a larger master file from which data communication with other submodels becomes possible (Figure 2.13b). Fabian Scheurer nicely summed up the need of segregating the design process into different models:

"On reasonably large architectural projects, there will always be a multitude of models serving a multitude of different needs." (Scheurer, 2012)



(a) Design-to-Production is very often situated in the middle of the design process and needs to take into account different constraints coming from the different trades involved in the architectural project.



(b) The digital design workflow developed by Design-to-Production separates the whole design process between different types (or families) of elements which are stored within separate sub-models that can be linked to the master file. Those different sub-models are not able to communicate between themselves and need to refer to the master file if the designer/user wants to check possible geometrical inconsistencies.

Figure 2.13: Design-to-Production's design digital chain and current workflows.

2.3.3 Early stage modelling at BuroHappold Engineering

An initial user requirement workshop has gathered both the author and different parties involved within the AEC Delta Mobility project (3D Repo, BuroHappold, Arup, [Speckle Works](#) and Rhomberg Sersa Rail Group). During the workshop, 10 different user stories have been collected from different actors – planner, estimator, design coordinator, project manager, consultant, engineer, end-user, field operative, admin and software developer – involved in the design process of a building project. Whereas a planner wants to associate planning activities to design objects to be able to plan activities in planning software (e.g. Tilos, Primavera P6, Bentley Synchro, Microsoft Project, etc.), an estimator would need to create a model view on the design scope in order to bring it into an estimation software and create a cost estimate. While a design coordinator collects and federates the latest BIM models to assign design tasks/issues back to the individual consultants, a structural engineer (or user) needs to have different view of the same data so in order to concentrate only what is important at any given time. Indeed, at early stages, BuroHappold Engineering does work on a centralized and unified model but already separates the concerns across multiple models (Figure 2.14) that focus on different areas (such as wind loads, footfall, quantities, energy, etc.). Those models inform design decisions which feeds back to the main architectural and structural models from which are derived the other multiple submodels. Overall, the workshop highlighted the fact that the different actors need different software, models, views from the same project to perform their respective tasks. Furthermore, each actor navigates within its own unique ecosystem of tools.

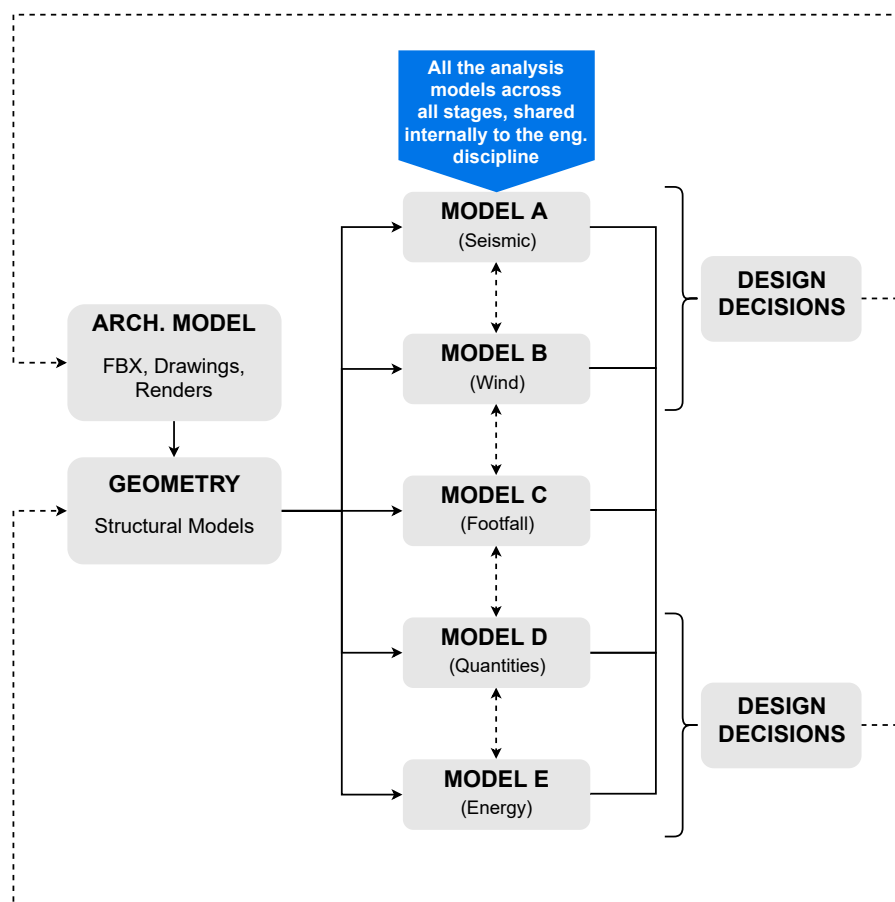


Figure 2.14: This diagram illustrates the cyclical, multi-model structural analysis workflow typically used within BuroHappold Engineering.

2.3.4 From Separation of Concerns to interoperability related challenges

It has been highlighted above that the current modelling processes developed at Front, Design-to-Production and BuroHappold carefully curate and fragment the design process into separate parts in order to allow manual interventions, classifying the generated objects and passing important information between the different (sub)models. Even though these strategies allow an efficient management of the data produced through all stages in the design process, they remain within a very controlled environment defined within each company. As it has been pointed out by Mette Ramsgaard Thomsen, Professor at the Royal Danish Academy of Fine Arts and head of the Centre of Information Technology and Architecture (CITA), the digital design strategies developed in each practice are hardly shared externally:

“The practice of architects building their own information tools, encoding their models and engaging directly with model interfacing is therefore an embedded part of existing practice. However, this effort is project-led, practice-specific and rarely shared.”
([Ramsgaard Thomsen, 2016](#))

Indeed, when it comes to externalize and share the data, communication and interoperability challenges arise, as information needs to be fragmented into multiple redundant models that carry specific data for each particular need (Figure 2.13b). Thus, the “Separation of Concerns” is actually transferred from within a specific practice to a larger network of different trades and stakeholders which also developed specific in-house design workflow strategies. Figure 2.15 gathers different conceptual diagrams from different companies and individuals that illustrate the evolutionary aspect of the design process. The latter keeps fragmenting itself in a hierarchical manner into sub-areas of concerns (Figure 2.15a) aggregating themselves throughout the whole digital chain (Figure 2.15b), resulting into a decentralized network of actors that need to communicate through different channels and software platforms (Figure 2.15c). Figure 2.16 illustrates a decentralized workflow in which multiple clusters of different trades coexist and collaborate during the design process. Each one of these clusters is focusing on particular parts or components of the building. Here, ten different clusters have been defined, from design and system intent to construction documents and support.

The traditional hierarchical model of the networked trades has been criticized by Dennis Shelden, Associate professor and Director of the Digital Building Laboratory at Georgia Institute of Technology and former Chief Technology Officer at Gehry Technologies, who advocates for redefining AEC practices, taking full advantage of the digital tools available to the construction sector:

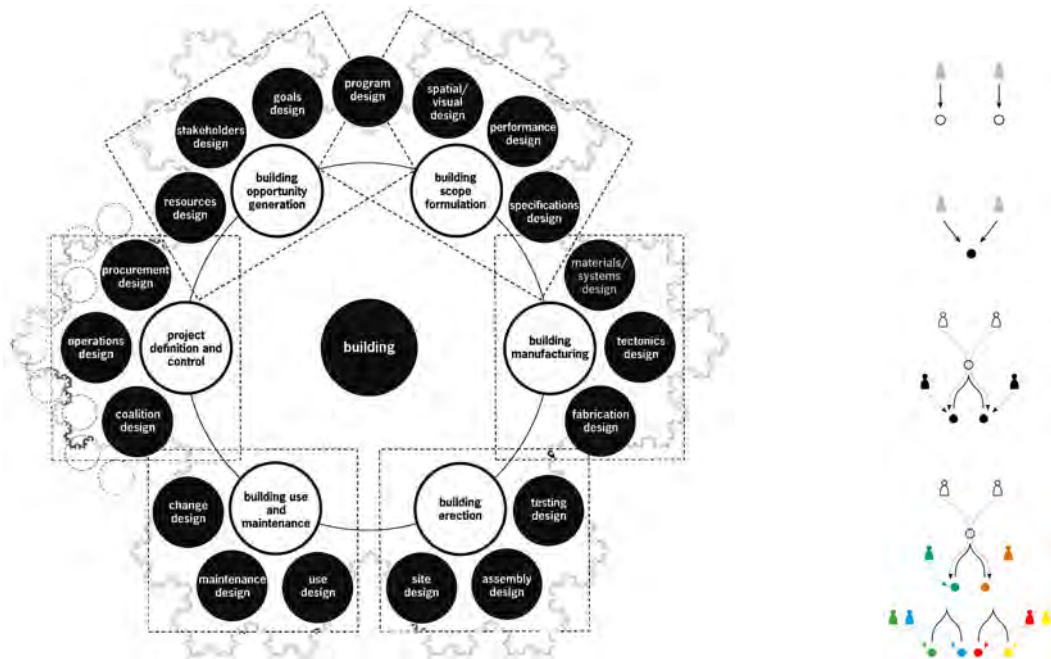
“While the heavy manufacturing industries have optimized around [...] hierarchical command-and-control structures, others have developed around more decentralized, webbed business organizations [...]. The business of building has yet to do either. [...] building teams are constructed as networked organizations, but are contractually restricted from functioning in this manner. The process of building has remained hierarchically structured in control, and linearly structured in time. [...] Digital tools offer a catalyst for pursuing alternatives to traditional project controls, by reducing the risk of innovation below the level of current practice. We are now seeing new ways in which designers and builders provide new value by forming new project organizations that branch across the tree.” ([Shelden, 2006](#))

Mette Ramsgaard Thomsen also criticized traditional hierarchical systems whose rigidity fails to take into account the dynamic relationships and feedback processes that can occur between the scales of design:

“Building practice is traditionally organized hierarchically as a sequence of subsystems each with their own scale of engagement and team of specialized professions. This division persists within modern design practice and is mirrored by legislative according of responsibility within the design chain. However, this understanding of design as a progression through the scales limits the potential for design innovation as it excludes our ability to understand how the small scales – material or detail – can affect large-scales – environment or structure. To support a real implementation of feedback in the design

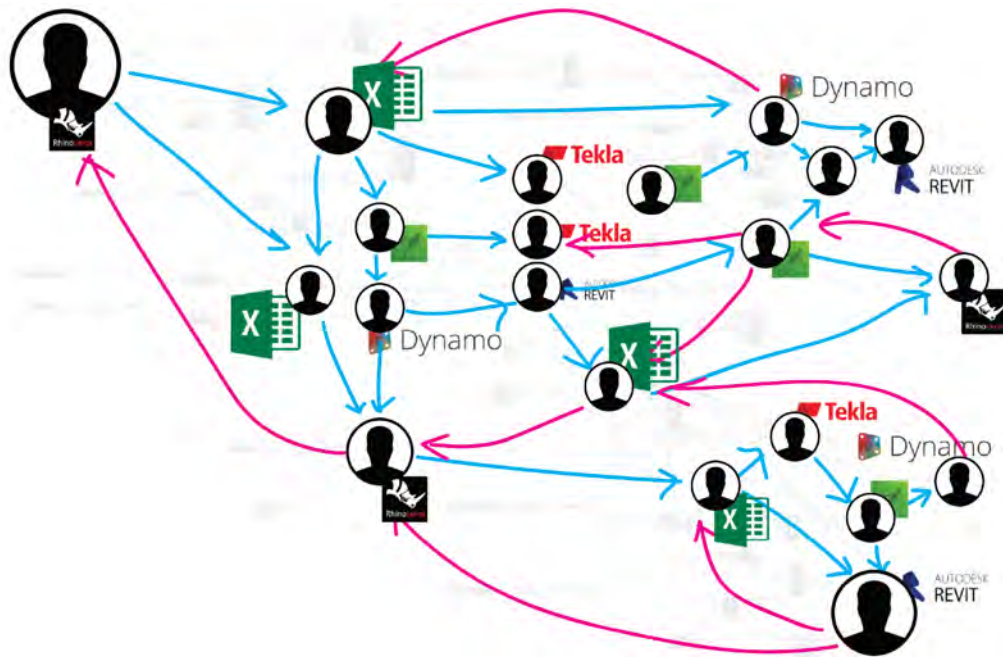
chain we need to develop mechanisms for multi-phased and multi-scalar feedback in which cyclical interdependencies can be investigated and assessed.” (Ramsgaard Thomsen, 2016)

These notions of feedback and communication are tackled within the following section which is describing more in depth interoperability issues within the [AEC](#) realm by taking into account the challenges enumerated above related to the separation of concerns during the modelling processes of large-scale and complex architectural projects.



(a) A Multi-Scalar, fractal representation of the design process across the different domains of expertise: "The different dimensions of the building cycle, wherein specific design activity must take place for goals to be set, strategies laid out and information organized. The system has fractal qualities in that it lends itself to being further subdivided into smaller areas of concern." (Tombesi, 2012)

(b) Progressive accumulation of the different stakeholders within the design process, expanding the digital chain (Kunz et al., 2014).



(c) Presented during a keynote given by Dimitrie Stefanescu at Smart Geometry 2018 in Toronto, this diagram represents the different communication pipelines existing between multiple stakeholders: the overall workflow is not directed and acyclic as a DAG, but non-linear and involving many feedback processes between the different involved trades and software platforms (Stefanescu, 2018).

Figure 2.15: Progressive separation of concerns between the different trades during the design process.

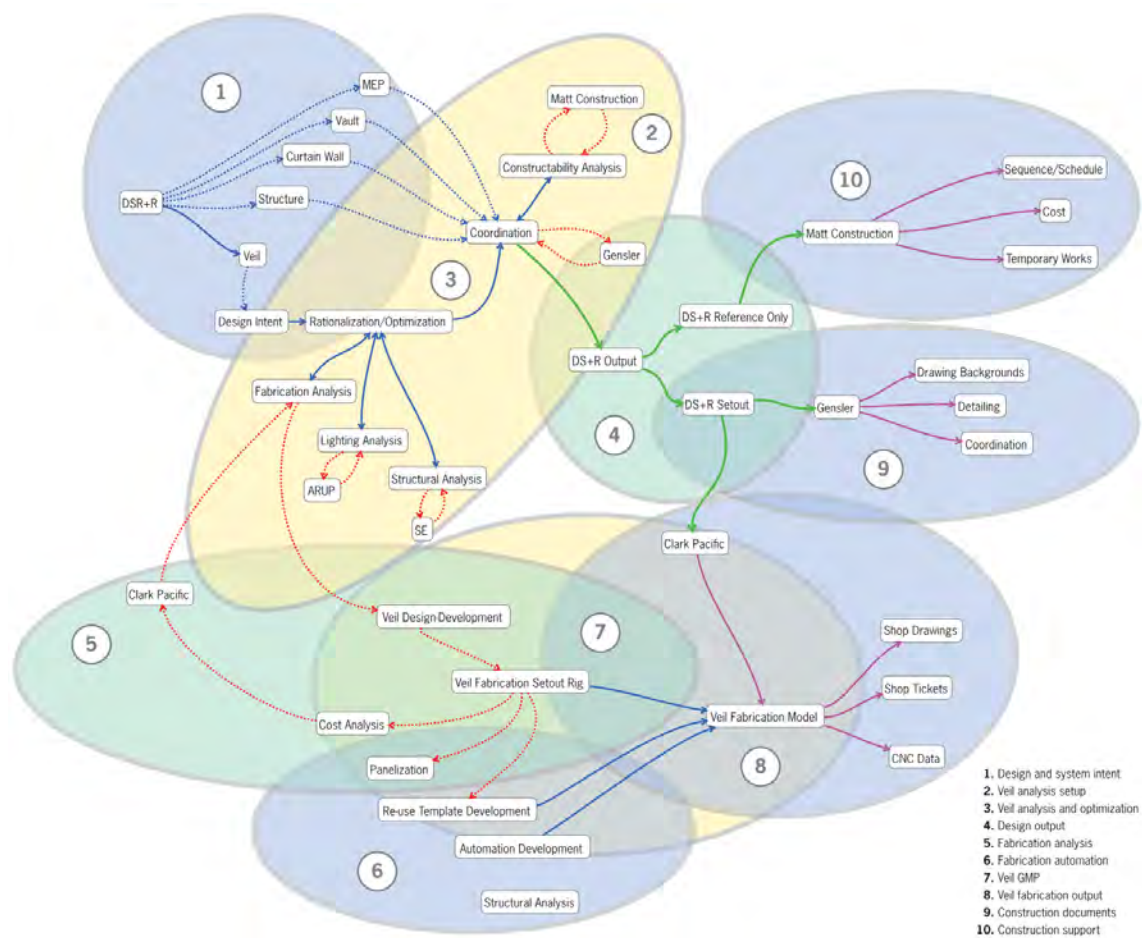


Figure 2.16: During this buy-in process, a workflow diagram of a concurrent design, engineering and fabrication process facilitated the discussion among the team members. For instance, during the design development of the veil around lighting and structural issues, information is being sent to the precast subcontractor to develop their fabrication processes and perform real-time cost analysis. The workflow was broken down into ten tasks as represented by the shaded ovals. The blue lines indicate design information; the red lines indicate fabrication and pre-construction information; and the green lines indicate the delivery of construction documents. (Marble and Kotronis, 2012, 167)

2.4 Enhancing Interoperability

Interoperability, data exchange and transparency is an interdisciplinary topic that do not only concerns the AEC industry but many other domains. The car industry is for example dealing with mass-produced objects of a much smaller scale than the architectural one, and is also facing very similar challenges to the ones encountered within the AEC realm, which relate both to parametric modelling and interoperability concerns. (Hirz et al., 2013).

This section is looking at the state of the art of applied interoperability strategies within AEC related practices. The first subsection is elaborating on the main existing standard, BIM and highlights its inherent problems and technological barriers. Facing those limitations, the second subsection is looking at alternative solutions, customized workflows, software and tools that are currently used and developed within different practices. Based on those case studies, the third subsection is highlighting the general trend for database modelling and the development of neutral format exchange protocols. Finally, the last sections briefly summarizes the above and develops on the social and legal implications of such meta-software strategies.

2.4.1 Building Information Modelling

When discussing the topic of “interoperability” in [AEC](#), the first term or concept mentioned is “[BIM](#)”, standing for: “Building Information Modelling”. [BIM](#) is nowadays the only existing and recognized standard in [AEC](#) that aims to solve the complexity of the design process on the level of interoperability. Nevertheless, 20 years of evolution have resulted in a rigid standard that mostly caters to technical stakeholders and ignores - and stifles - an agile design process. Indeed, since its creation, it is fair to say that [BIM](#) has been facing a lot of critics and polemics, especially coming from architects and designers who have been complaining about it:

“Architects frequently blame [BIM](#) for its bureaucratic bias, and the technology was indeed developed primarily for managerial, not design, purposes.” ([Carpo, 2017](#), 141)

Technical and philosophical definitions (big [BIM](#) vs. little [BIM](#))

[BIM](#) is being generally conceptualized both on a philosophical and technological level. From the philosophical point of view, [BIM](#) represents the idea of a unified, integrated model that comprises the entire complexity of an architectural design project. For the technicians, [BIM](#) relies on the Industry Foundation Classes (IFC) standard format that contains today more than 1200 different object classes (<http://www.buildingsmart-tech.org/ifc/IFC4/final/html/>, link accessed on the 16th of March 2018), a domain specific object-oriented programming (OOP) language that one has to embrace in order to meet the current required specific standards. More than a language, IFC generated objects must keep the parametric relationships that might exist between others. From both the philosophical and technical perspective, the main idea behind [BIM](#) is that the different trades working on different software platforms are able to communicate through the same model, using the same standard (IFCs). A similar distinction has been made by Finith Jernigan who differentiated “[BIM](#)” (uppercase), standing for the theoretical notion of an integrated design and exchange of project data (which we can compare to the above philosophical conception), and “bim” (lowercase), referring to the practical tools, applications and underlying technology on which relies “[BIM](#)”, such as IFCs ([Jernigan, 2008](#)).

Technical barriers and misconceptions of [BIM](#) and IFC classes

It has already been mentioned, both in a critique of Integrated Project Databases ([Amor and Faraj, 2001](#)) and in a paper describing problems related to [BIM](#) ([Aranda-mena et al., 2006](#)), that there are some misconceptions in the implementation/adoption of the Industry Foundation Classes. The latter is facing different barriers, both technical and theoretical:

- **The Object-Oriented Model as the complete solution?**

The fact that IFCs rely on an Object-Oriented Programming library might be very problematic, as it has been observed that OOP difficultly scales up for large and complex construction models:

“Object-oriented modelling and programming is not well suited to very large systems. This should be of serious concern to the construction industry’s modellers as construction models are arguably some of the largest and most complex models which have ever been developed. The main problem here is that object-based systems are well suited to micro-level specification, but have few constructs which enable a macro-level specification to be managed.” ([Amor and Faraj, 2001](#), 60)

Furthermore, those Object-Oriented systems “are not easy to validate” because it is difficult to prove that they actually are “consistent and non-redundant” ([Amor and Faraj, 2001](#), 60). It might be therefore relevant to look for other solutions that attempt implement IFCs in different ways than the OOP approach.

- **The single model/repository fallacy: a confusion between the “model instance” and the “Shared Data Model”**

Most of the people new to the field of [BIM](#) believe that the overall concept/definition of the latter revolves around the idea of having a single 3D model shared by all team members. This might be true in some cases, but it is important to remind that IFCs represent first of all a “Shared Data

Model” (or a shared data structure as a standard) enabling a seamless communication between different software platforms:

“It is thought that non-technical users in the industry mixed up the “Shared Data Model” (meaning IFC as a description of data structure) with an instance (data from a specific project) which was also called “model” in the BIM-terminology.” (Van Berlo et al., 2012, 812)

The subsequent concept of a possible unified 3d model is often criticized and rejected by a large part of the BIM community:

“This idea, which is intoxicating in its naivety, seems to be raised by everyone new to this field. [...] History has consistently shown this to be impossible for complex domains and it is frustrating to have to reiterate the problems with, and arguments against, a single data model with every new generation of modellers in this area.” (Amor and Faraj, 2001, 60)

Van Berlo even distinguished two distinct groups of people within the BIM community itself: one that strongly believes “in working with a central data repository based on a single homogeneous software environment” and another that “only believes in a shared data repository, but finds this has to be based on an open data model like IFC.” (Van Berlo et al., 2012, 812). The distinction between those two groups resonates with the different networked models drawn by Paul Baran in 1964: while the first group would believe in the Centralized network model (Figure 2.17, left), the second one would refer to either the Decentralized or Distributed network models (Figure 2.17, center and right), where connections between nodes persist through the exchange of IFC objects.

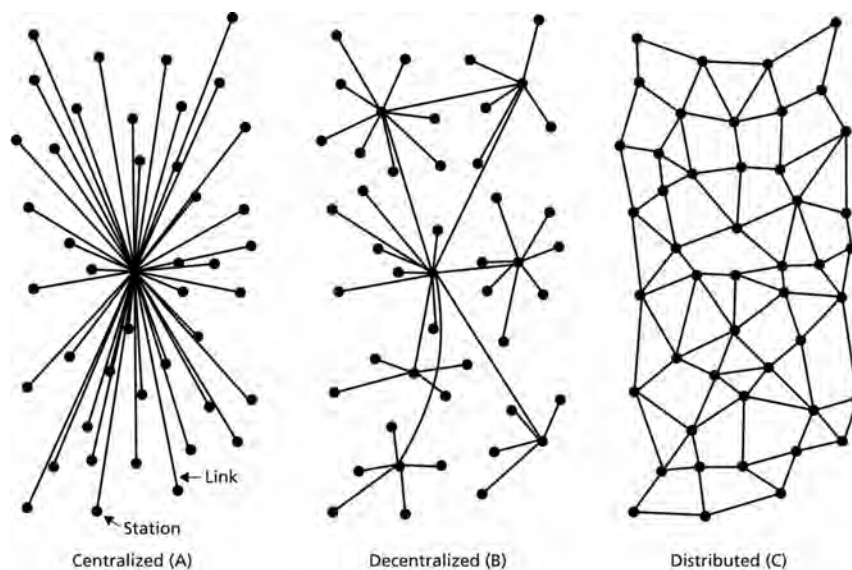


Figure 2.17: Centralized, Decentralized and Distributed Networks. (Baran, 1964)

Two main arguments against the single homogeneous model, have been raised by Amor. The first one, technical, observes that the number of objects required to be modelled within the same environment would be too high: “Current views are that two to three hundred objects in a data model are not easily manageable (the CIS/2 model (CIMsteel 1995) is seen as a likely upper-limit on the size of model which can be handled). A single data model would have tens of thousands of objects to describe, with complex inter-relationships between one another.” (Amor and Faraj, 2001, 60-61). The second argument, more conceptual, highlights the difficulty to conciliate different views coming from different trades: “For example, the world view of an architect (and hence the model they would require) is very different from that of a structural engineer, or a

quantity surveyor. Even if a single model could be created for each domain (and this is not thought to be feasible) it would not be possible to merge all views into a single coherent whole.” (Amor and Faraj, 2001, 61). Today, such issue is often tackled with clash detection algorithms and methods, for instance.

Another argument against the centralized model in favor of decentralized networks to resist security threats has been raised by Paul Baran:

“In Baran’s model, a centralized network – which is ultimately what a tree is – is vulnerable to attack because destroying its center makes it impossible for any of the nodes to communicate with each other. If you take out the top node of a tree, the rest of the tree falls apart. A decentralized structure has a hierarchy of centers, each linked to its own galaxy of nodes, but destroying one of these networked nodes would cause its relatives to be lost. The least vulnerable and most secure structure is the distributed network, with multiple links to each node, all interconnected with one another. The distributed network offers a higher level of redundancy, so if one node were knocked out of service, communication could flow through the links and still be able to communicate effectively.” (Steenson, 2017, 53)

In *Computing the Environment: Digital Design Tools for Simulation and Visualisation of Sustainable Architecture* (Peters and Peters, 2018), Brady Peters also emitted a critic towards the single homogeneous model:

“Traditionally, CAD and BIM systems have been monolithic applications, but increasingly there is pressure to connect. The promise of a single piece of software to carry out all design tasks and be used from conception through to operation is still not realised, and may not be the best strategy anyway. [...] now many pieces of software with many different commands and sub-programs are used, and these form complex design processes.”



**When the architect and the structural engineer give their models
to the bim coordinator for a collision check**

Figure 2.18: A recent meme mocking the current role of BIM Manager has surfaced on internet. It depicts the BIM Manager as the figure of Jesus Christ, who is supposed to integrate and solve all the data coming from the different trades (illustrated here by two of the Three Kings) involved in an architectural project (<https://www.facebook.com/You-are-not-a-BIM-manager-1872064509773725/>).

- **Models only represent reality**

Finally, the third and last main misconception is to believe that it is possible to completely model reality. Such concept is definitely unrealistic, simply because “the effort that would be required to model the full range of attributes of any single object utilised in construction is beyond the effort available for simple objects (e.g., a nail), and representationally impossible for more complex objects (e.g., a door or chair)” (Amor and Faraj, 2001, 61). Instead of trying to model everything at the highest level of resolution, strategies need to be developed in order to manage the minimum amount of data that would be required for the realization of the project, and not just to represent it.

- **Slow and delayed development of the IFC standard to meet specific industry needs**

Fabian Scheurer also raised the issue of the slow development of IFC classes that both failed to meet state of the art requirement in AEC and to coordinate with the releases of its related BIM software:

“Although there has been an ongoing attempt since 1995 with the “Industry Foundation Classes” (IFC), an open standard for so-called “Building Information Modeling” (BIM), there have been long delays between releases of new versions. This timeframe between releases poses an interesting dilemma: while new versions of software packages with new functionality appear on the market every year, a four-year innovation cycle for the underlying data format seems rather unhurried. Can the IFC standard keep up with the development?” (Scheurer, 2012, 111)

Furthermore, Scheurer also pointed out that the IFC classes failed in integrating NURBS geometry (at the time of his writing), important for designers and the mathematical drawing of complex architectural projects but neglected by the IFC’s developers, more concerned about managerial issues:

“But standards do not only make life easier, they also make life simpler. Standards reduce the infinite possibilities of the real world to the least common denominator of all involved parties. In the case of the IFC, for example, a building cannot have any curved free-form shapes. They simply cannot be described within an IFC model, because the definition of such shapes is not (yet) part of the IFC standard. A precise mathematical theory of those shapes (called NURBS) has been available since the 1950’s, and many of today’s CAD packages provide the necessary functionality to model them. But obviously, when IFC2x3 was defined and approved only a few years ago, the precise modeling of curved shapes was regarded as low priority compared to other features.” (Scheurer, 2012, 111)

Additionally, Scheurer explains the lack of interest from the software developers in integrating NURBS geometry within the IFC standard by the fact that free-form architectures only represent a very small market share:

“On the other hand, putting the curved buildings that made it to the pages of the architecture magazines in relation to the total volume of buildings erected during the last four years, the decision seems more reasonable. Why inflate a nice and lean standard with a feature that is needed for less than one percent of all buildings in the world? [...] Since most of the IAI members are software vendors and draw their decisions on the basis of economic considerations, they consequently decided not to waste scarce resources to address less than one percent of the building industry.” (Scheurer, 2012, 112)

Finally, Scheurer goes even further by speculating that even if NURBS geometries are one day added to the IFC standard, future needs from the industry will still have to be continuously considered within this same standard, causing further implementation delay and consequently pushing the designer in developing its own custom methods:

“No matter how many new functionalities are added, a standard tool will always only work for standard designs. And creative designers will always try to escape or overcome the standard.” (Scheurer, 2012, 112)

“Pseudo BIM”

Because of the double understanding of **BIM** (the existing dichotomy between its theoretical aspect and its practical implementation), and its corresponding technical challenges and misconceptions, a lot of confusion emerged within the **AEC** community while trying to agree on its clear and common definition. Therefore, the answer to the question “What is **BIM**?” might differ from one expert to another. For example, the strategies developed at Design-to-Production and Front (see section 2.3) could be in certain cases presented as **BIM** (from a theoretical/philosophical point of view), even though the geometries produced by these same consultants haven’t been processed through the IFC standards and protocols. Such practice is defined by Dominik Holzer, Senior Lecturer in Digital Architecture at the University of Melbourne, as “Bad **BIM**” or “Pseudo **BIM**”: *“There exists a spectrum of **BIM** ‘pseudoness’ eventuating in practice. In its worst form, pseudo **BIM** is used to pretend **BIM** was applied whereas in reality a traditional **CAD** workflow was used to deliver a project. The reasons for such deception may be to impress clients (who may not know the difference), or to conform to client/regulatory requirements.” (Holzer, 2016, 23).*

Because Design-to-Production and Front are specialized in rationalizing geometrically complex projects, using IFC standards presents itself as an extra data format constraint to solve, especially because IFC has been originally thought and developed for producing traditional, standard architectural projects. Fitting (or even forcing) free-form objects within the IFC language means that one has to take the time to understand it, embrace its complexity and comply to it. Such effort might not be worth it when one has to rationalize many different pieces before a short deadline. This particular issue resonates with the above critique of setting up an exclusive **DAG**-based workflows for a unique building: *“[...] the one-off design of most buildings makes it a very different procedural problem from the mass production of a jet or car, where the time and effort required to create a comprehensive digital model can be justified through its repeated use.” (Scheurer, 2012, 130).* The IFC framework therefore stifles the expressiveness at early design stages, breaking the design process into two separate parts: early stage design versus subsequent and later phases. Once embraced, just like with **DAG** models, operating changes becomes a time consuming effort.

BIM vs. parametric design software

The differentiation between “true” **BIM** practices and the methodologies developed at Design-To-Production and Front leads to distinguish **BIM** and Parametric Design. In *“Bridging Building Information Modeling and Parametric Design”*, Stefan Boeykens, **BIM** specialist/consultant for D-Studio (a **BIM** consultancy practice in Belgium) and guest professor at KU Leuven, lays out the differences between the two. Just like Carpo, the author notes that **BIM** is not fully adopted by architects and researchers, who prefer to rely on more generic modeling and drawing techniques, either *“out of lack of experience or based on a different point-of-view on the design process, where **BIM** is often disregarded”*. Indeed, **BIM** limits “Direct Modelling” (Pena De Leon, 2014) by *“using internal algorithms and embedded knowledge about the construction domain”* (Boeykens, 2012, 453). Ramon van der Heijden, former Computational Design Specialist at Front Inc and Digital Design and Services Leader and Associate Director at Arup made a similar observation, witnessing the lack of design freedom in **BIM** software:

*“Conventional **BIM** platforms such as Revit and ArchiCAD, apply a similar object construction logic to their processes. A wall for instance, is defined as a reference line, a wall type and a height. However, the available attributes of a wall are predefined and cannot easily be expanded or modified. Similarly, the relationships this wall can have with other building elements is limited to what is allowed for a “Wall Object” by the software. When a wall needs to be changed into an overhead door, for example, hardly any of the attributes or relationships can be preserved.” (Van Der Heijden et al., 2015)*

Boeykens also distinguishes **BIM** and Parametric Design models by the degree of dependencies that exists between the objects:

“While BIM relies strongly on parametric functionality, it is mostly used on an object-level. The BIM model thus behaves more like an assembly of rather independent objects. In parametric design, the whole project becomes a single assembly, with full control over both the overall form and the smaller details.” (Boeykens, 2012)

The author highlights here the fact that parametric models usually tend to integrate all geometrical constraints through all scales, resulting in a massive global model which becomes inflexible at late stages in the design process when it comes to add information and metadata to the generated geometrical objects. In other words, where BIM models are extensively structured, containing clear semantic information, parametric models are instead exclusively oriented towards geometry and modelling processes, lacking therefore basic semantics and metadata. Dominik Holzer concludes that designers should find the right balance between relying on BIM for its managerial purposes and keeping enough control on the design process by developing custom tools and workflows:

“As designers, we have to scrutinize our design methodology to distinguish where BIM can assist us to do our work more efficiently, but also where it might limit our creativity in the conceptual design process. Only by doing so, we can develop our own specific way of designing with the help of computational processes which are currently not supported by BIM.” (Holzer, 2007)

For that matter, the next paragraph is looking alternative approaches towards BIM, aiming at enhancing both its modelling flexibility and its interoperability features.

Enhancing and simplifying BIM practices

Facing the limitations described above that come with the exhaustive lists of IFC object classes, recent research has been focusing simplifying BIM practices through Linked Data methodologies and more particularly the “Building Ontology Topology” (BOT), a simple data-centric approach seeking to simplify BIM practices by working within a limited set of core building classes that still allow for further domain specific extensions when required (Rasmussen et al., 2017b). This approach has been tested through the development of a web-based application allowing the user to operate precise queries across a multitude of IFC elements and their properties (Rasmussen et al., 2017a). Parallel research and efforts are being made in order to make BIM smarter, more flexible and more open:

- **FlexibleBIM**

The term “FlexibleBIM” has been coined by the developers of VisualARQ, which is a plug-in software for Rhino3D and Grasshopper3D: <https://www.visualarq.com/>. The latter offers both modelling freedom through the Rhino3D environment, interoperability features based on IFC import/export, and the ability to add custom parameters to geometrical objects. Those three features enable a more flexible design workflow in which the user can highly curate and customize its own data, expanding therefore previously rigid processes within more common BIM protocols.

- **OpenBIM**

“OpenBIM” has been coined by buildingSMART – a “worldwide authority driving the transformation of the built asset economy through creation & adoption of open, international standards” (<https://www.buildingsmart.org/about/>) – to propose more open and transparent workflows relying on the existing IFC standards.

Despite the efforts described above, BIM hasn’t solved all modeling and interoperability issues that the AEC world is currently facing. Some other strategies look at bridging together different software which do not necessarily meet IFC standards. As Mette Ramsgaard Thomsen stated, the AEC industry is fully aware of the limitations inherent to Building Information Modeling and therefore its own in-house custom tools and workflows for interfacing better between models and software platforms:

“Practice is aware of these embedded limitations of current modeling practice and the industry development has therefore been paralleled with a series of profession-led research and development efforts creating bespoke modeling methods allowing complex design solutions and creating links to fabrication.” (Ramsgaard Thomsen, 2016)

Nathan Miller also pointed out that these custom workflow present themselves as substitute to the existing software platform that do not provide all the tools to answer particular industry demands:

“These workflows generally begin as highly specific solutions to immediate problems that cannot be solved with functionality given in out-of-the-box software packages.” (Miller, 2010)

Furthermore, a recent report released by the Association of Project Management (APM) and authored by Andrew Davies – Professor of the Management of Projects and Director of Research at the Bartlett School of Construction and Project Management – highlights the need of developing custom specific solutions to tackle contemporary large-scale and complex projects:

“Many infrastructure projects in the UK now recognise the need for solutions that are designed to deal with the specific challenges involved in planning and executing large, complex projects. Innovation is important in addressing the varying degrees of uncertainty that can be found within different parts of a large, complex project. Over the past decade, many of the UK’s largest and most complex infrastructure projects have abandoned traditional delivery models.” (Davies, 2019)

This observation correlates with a recent analysis from the McKinsey Global Institute, ranking construction as second least digitized industry (Figure 2.19).

The next section looking in more details at current custom-made solutions through the lens of multiple leading practices for improving interoperability and further scaling up complexity within the AEC sector.

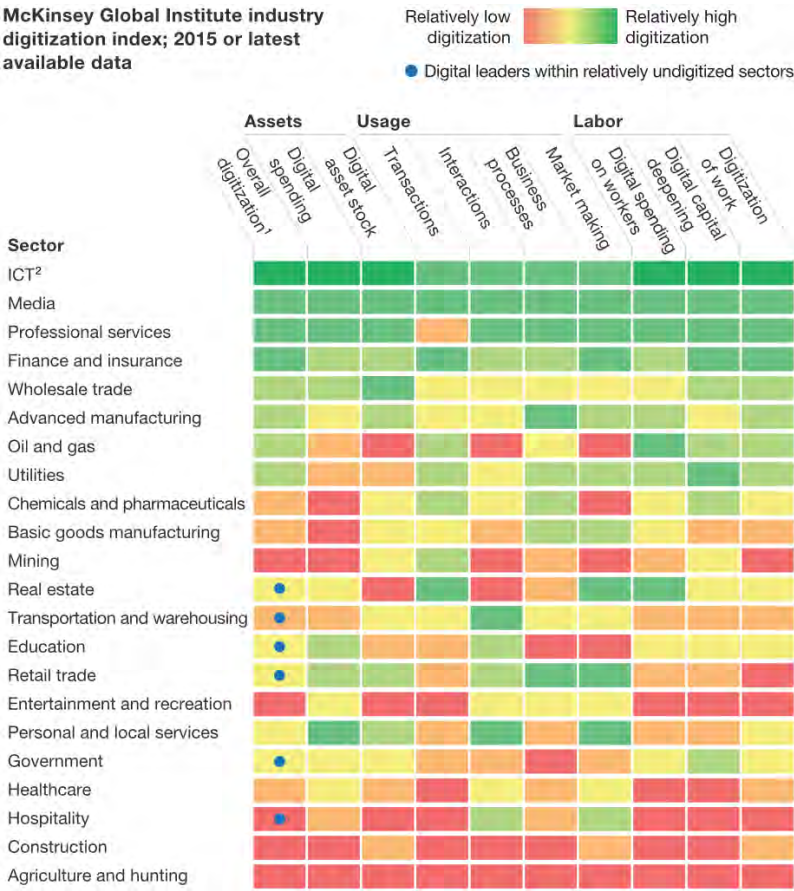


Figure 2.19: “The construction industry is among the least digitized.” according to the McKinsey Global Institute

2.4.2 Custom-made solutions for improving interoperability from within AEC practices



Figure 2.20: The BIM Ecosystem, as illustrated by Paul Wintour (<https://parametricmonkey.com/2016/06/20/bim-ecosystem/>)

Because IFC/BIM-related standards are not perfect and not adopted by all software platforms and the whole AEC community, many in-house solutions have been developed, enhancing interoperability between different environments. The present subsection will describe several existing in-house strategies, tools and methodologies developed by different companies, architects, engineers, consultancy practices and organizations.

The CAD software's centric approach: open API vs. closed API

One of the main approach to tackle interoperability outside the realm of BIM/IFC is to define an overall inter-software framework, within which a common platform is used as a central shared data model, its API serving to translate geometry from/to the other software platforms. Such approach can work internally within an office, or externally if the different involved trades agreed upon the common software to use as the main interoperability tool. To illustrate this method, two existing workflows are described here, the first one placing Rhinoceros 3D at the center of the design process, and the second one using CATIA's 3DEXperience Platform as the main modelling and interoperable environment.

- **Open API: Rhinoceros 3D**

Because of its affordable price, openness (see section 2.1.3), clear API and the many different available import/export formats⁹, Rhinoceros 3D is often chosen by architectural firms to be used as the main and centralized interoperable software platform. This is the case of the custom workflows developed and used by SHoP architects (Figure 2.21a). This practice considers that *"an all-compassing software solution is generally no more effective than using pliers to drive a nail."* Instead, they *"use many different types of software throughout the design, fabrication, and construction phases."* (Nobel et al., 2012).

- **Closed API: CATIA's 3DEXperience Platform**

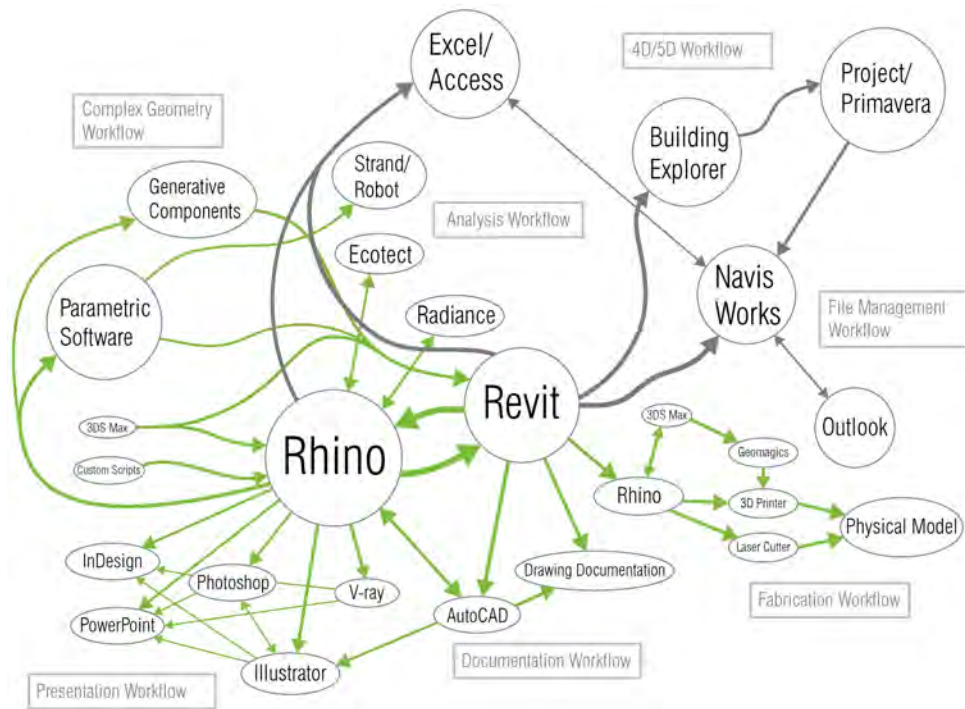
Recently, CATIA has implemented its own web-based interoperability environment - the 3DEXperience Platform - allowing better data exchange and queries across models. Unlike

⁹<https://www.rhino3d.com/formats/>

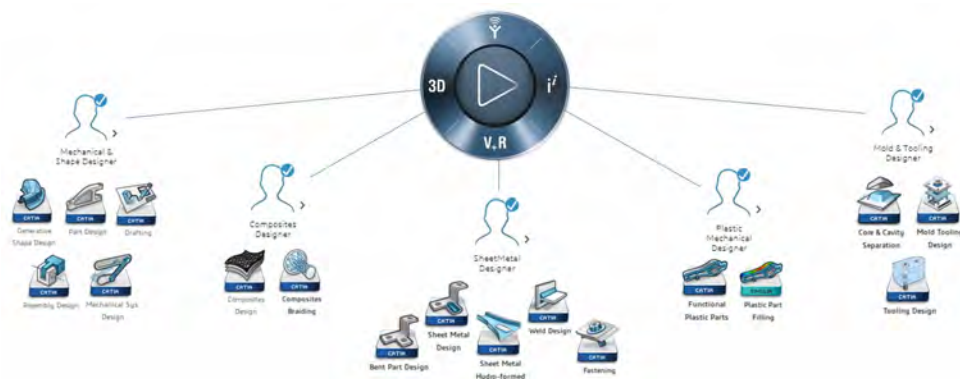
Rhino3D, CATIA's 3DEXperience Platform does not offer a clear and open [API](#) and therefore limits the possibility of creating custom workflows. Instead, it promotes itself as an all-compassing software environment (an approach criticized above by SHoP architects) and asks its users to integrate all processes within it (Figure 2.21b). Such approach has also been criticized by David Stasiuk: *"The thing that I am the most skeptical about Dassault's solution is the fact that the 3DEXperience platform is an impenetrable ecosystem of its own tools, which are notorious for just liking to work with themselves."* ([Miller et al., 2019b](#))

Several architectural and consultancy practices (such as Zahner, Shop Architects and Morphosis) started to adopt this software in order to manage high-resolution data sets from design to fabrication. Even though such platform increased drastically the workflows for its adopted users, one of its remaining limits is its poor capability to operate and communicate seamlessly across external software platforms. The user can hardly create custom interoperability channels to easily share information from the 3DEXperience Platform to other software packages. Instead, the software developers expect its users to perform all operations within CATIA's related products, and thus for commercial purposes.

The next paragraphs in this section will describe custom workflows and tools develop by different leading [AEC](#) practices. Most of them make use of the Open [API](#) approach which enable a custom and curated development of interoperability channels between different software packages.



(a) Software workflow diagram, SHoP Architects (Nobel et al., 2012).



(b) Centralized CATIA's 3DEXperience Platform: Roles and Applications.

Figure 2.21: The CAD Software's Centric Approach: Open API (top) vs. Closed API (bottom).

Proving Ground's custom workflows

Proving Ground is a small-sized and leading digital design consultancy practice that delivers custom interoperability workflows and tools for diverse AEC-related clients, enabling “digital transformation with creative data-driven solutions to the building industry”. Nathan Miller, CEO of Proving Ground, believes that [...] *inventing customized workflows are a necessary part of the design process and have a profound impact on the delivery of the architecture.* (Miller, 2010). The company always tries to abstract and generalize these specific workflows, from which are further developed open-source plugins that could serve more generic purposes and be reused for future collaborations. For example, one of the latest research pursued by Proving Ground focused on mesh-based workflows for handling complex geometry in BIM, aiming at reconciling generative and computational design processes with BIM-related software. Starting from the initial observation that friction still exists between the two – “[...] BIM’s inadequacy in supporting these data structures produces both friction and waste during production, including painful remodeling processes and incomplete or low-fidelity documentation for complex geometries. [...] a time-consuming process that directly contradicts BIM’s fundamental utility for project delivery: to simultaneously manage design geometry across multiple, interdependent, multi-media and on-demand representations.” (Miller and Stasiuk, 2017, 406) – Proving Ground developed a custom workflow, based both on the APIs from Rhinoceros 3D and Revit, in order to process complex mesh-based geometries and translate them into clean IFC objects within Revit families (Figure 2.22), enabling the end-user to “automate mesh importing by directly accessing geometry contained within a Rhino file. This allows users to both control mesh edge visibility and then parameterize them as Revit families.” (Miller and Stasiuk, 2017, 411).

In the past few years, Proving Ground has been developing many different tools that answer different problems, ranging from computational design modelling to interoperability issues. Some are free and open-source, such as: LunchBox (geometry processing and machine learning), Conductor (agile task management), Conduit (data visualization), Rhynamo (interoperability bridge between Rhino and Dynamo, as illustrated in the case study described above), Slingshot (database modelling), and Freightier (packaging and shipping grasshopper tools). Some others are more related to enterprise customization, such as: Conveyor (Rhino to Revit workflow), Minecart (BIM data mining) and Semantic (early stage data)¹⁰.

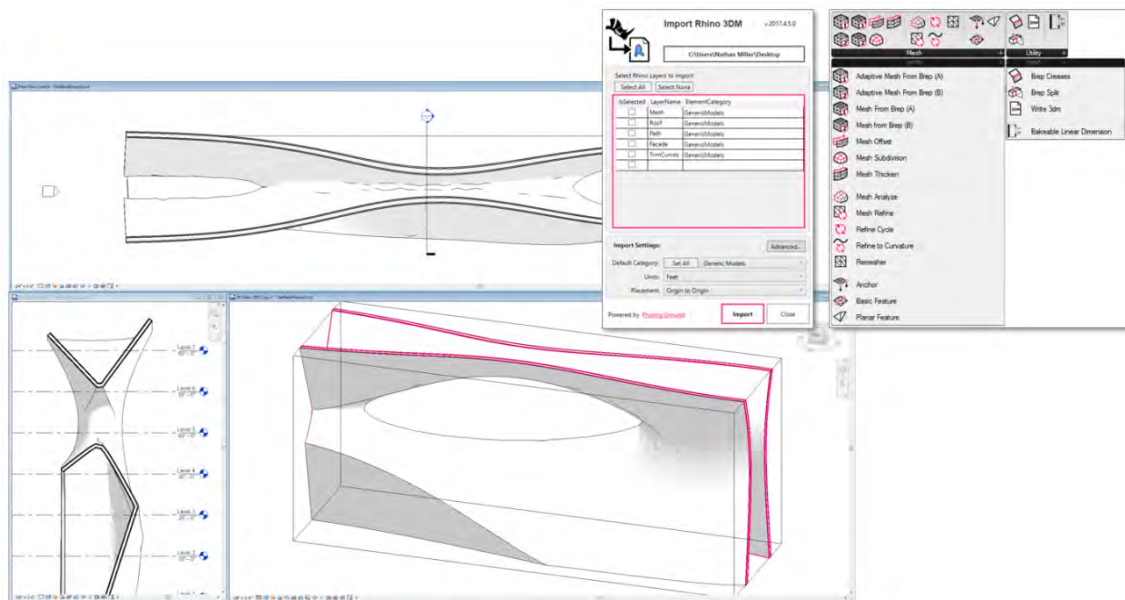


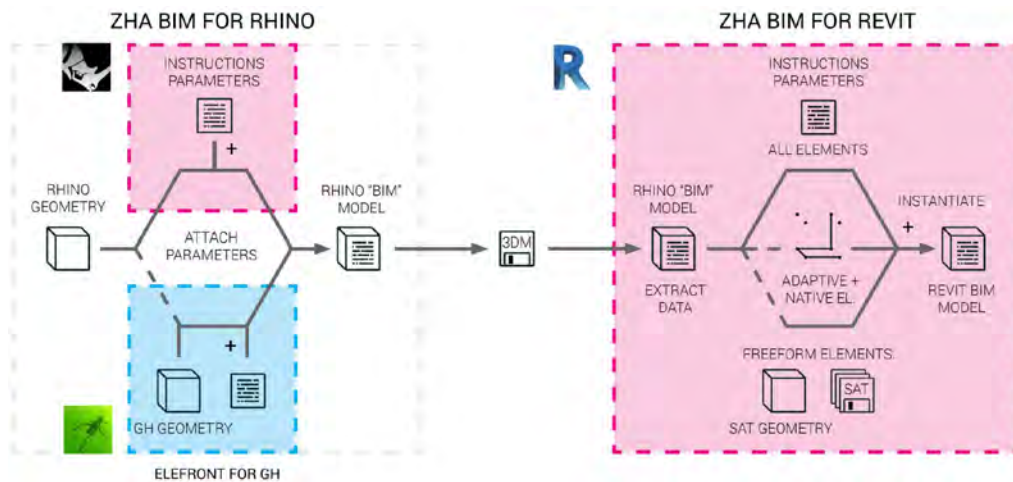
Figure 2.22: The research led by Proving Ground showcases custom Revit and Grasshopper tools used to refine meshes and streamline the import of meshes into the BIM environment. Mesh edge control, graphics options, and data assignment are exposed through custom interfaces.

¹⁰<https://provingground.io/tools/>

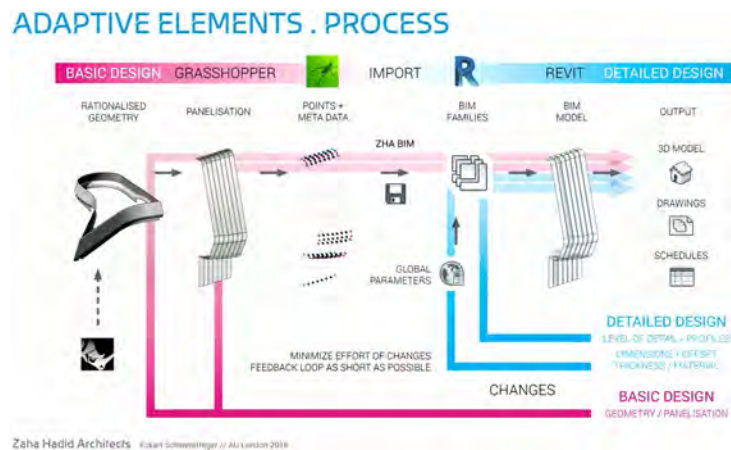
ZHA BIM custom workflows

Zaha Hadid Architects is renowned for delivering very complex and large-scale architectural projects reaching sometimes the urban scale that come with many challenges when translating the overall design intent to precise data for fabrication. Eckart Schwerdtfeger, BIM Lead at Zaha Hadid Architects has developed custom workflows based on the [Speckle](#) framework to enable more direct and fluent cross application information exchange, from the early design stages to the late stages through BIM ([Schwerdtfeger, 2018](#)). Such software development would prevent duplicate work and enable the acceleration of design iterations and the implementation of changes by saving the time spent on repetitive tasks. The main challenge undertaken by Eckart was to connect the two main software platforms used at Zaha Hadid Architects: Rhino3D for the design tasks and Revit for the BIM model generation and documentation part. To achieve this, a consistent workflow has been developed enabling full interoperability between the two software environments along with the management of data-rich geometrical objects (Figure 2.23a). First, instructions parameters are attached to the Rhino3D's designed objects, either through the native software's interface or through the Elefront's plug-in for Grasshopper3D. This enable the generation of a Rhino3D pseudo-BIM model (see 2.4.1), ready to be analyzed and extracted within Revit, in which the geometrical objects are further regenerated and their respective information reinterpreted to produce the final BIM documentation. In general, the information transferred between Rhino3D and Revit is in this case kept to the very minimum allowing the geometrical reconstruction within the second platform. Points and Polylines are preferred to NURBS geometry, both for their simple, light format and their ability to be serialized during exchange protocols (Figure 2.23b). Based on the existing [Speckle's](#) Application Programming Interface, custom Dynamo's (the Revit's Visual Programming environment) nodes have also been developed to give an efficient UI to transfer geometrical data between the two platforms (Figure 2.23c).

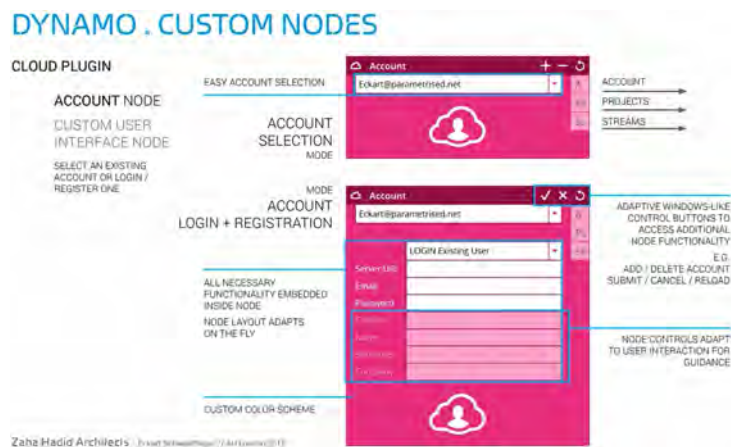
2.4. ENHANCING INTEROPERABILITY



(a) Custom workflow developed by Eckart Schwerdtfeger enabling interoperability between Rhino3D and Revit, translating data-rich geometrical objects (designed in the former platform) to IFC elements ready for documentation (generated in the latter).



(b) The workflow described above in figure 2.23a has been used on different projects realized by Zaha Hadid Architects. It is common practice to transfer the minimum of data required enabling its reconstruction within the receiver software platform.

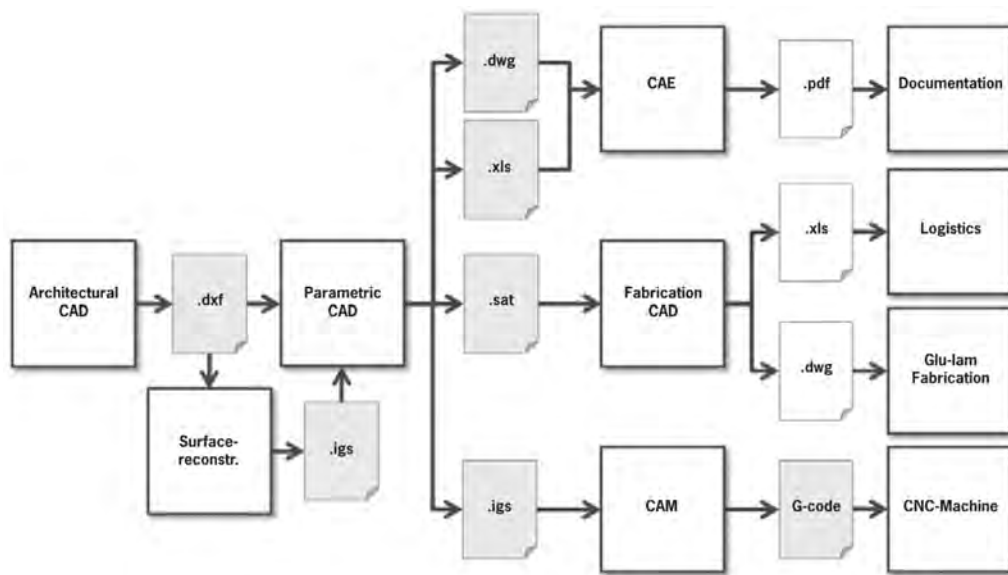


(c) Based on the [Speckle](#) Application Programming Interface, custom Dynamo nodes have also been developed by Eckart Schwerdtfeger enabling the designer to exchange information between software platforms in a very intuitive manner.

Figure 2.23: Different custom tools and workflows developed by Eckart Schwerdtfeger at Zaha Hadid Architects.

Design-to-Production's custom workflows

It has been described earlier (2.3.2) that Design-to-Production needs to separate the concerns and curate specific 3D models for each external trade (Figure 2.24a). Design-to-Production is increasingly concerned with management and workflow issues for prefabrication projects, and is developing digital tools to improve those workflows. Amongst them, a work-in-progress web-platform would allow the office to better manage the produced objects for further delivery. Naming, tags, status, location and all the assembly related properties of each element would be gathered within an on-line table that is for the moment a traditional Excel table developed in-house (Figure 2.24b) from which any client of Design-to-Production would be able to retrieve, query the components needed for fabrication and assembly, thus progressively while the project is being erected.



(a) "Diagram showing the flow of information between the various software programs used for planning and fabricating the timber roof structure of the Centre Pompidou-Metz by Shigeru Ban and Jean de Gastines." (Scheurer,

2012, 110)

Export Status											
Exportiert zur Freigabe: 67.4%											
Exportiert zur Fertigung: 67.0%											
g	Name	Traeger	Achse	Lage	Lfd.Nummer auf Achse	Baubabschnitt	Devi-Zone	Gebauedeseite	Montagenummer	Site-Nummer	Exportdatum
10	011.B01	Man	011	B	01	om4	322.803		2	om4N.002	29.01.2017
11	011.B02	Man	011	B	02	om4	322.803		1	om4N.001	01.02.2017
12	013.B01	Man	013	B	01	om4	322.801		3	om4N.003	29.01.2017
13	013.B02	Man	013	B	02	om4	322.803		5	om4N.005	29.01.2017
14	013.B03	Man	013	B	03	om3	322.803		9	om3N.009	01.02.2017
15	013.D02	Man	013	D	02	om2			0	om3N.????	17.02.2017
16	014.A01	Man	014	A	01	om4	322.803		10	om4S.010	02.02.2017
17	014.C01	Man	014	C	01	om4	322.801		11	om4N.011	27.01.2017
18	014.C02	Man	014	C	02	om4	322.801		22	om4S.022	27.01.2017
19	014.C03	Man	014	C	03	om4	322.803		11	om4S.011	27.01.2017
20	015.B01	Man	015	B	01	om4	322.801		4	om4N.004	01.02.2017
21	015.B02	Man	015	B	02	om4	322.803		6	om4N.006	01.02.2017
22	015.B03	Man	015	B	03	om3	322.803		20	om3N.020	21.01.2017
23	015.B04	Man	015	B	04	om3	322.803		8	om3N.008	01.02.2017
24	015.D01	Man	015	D	01	om4			0	om4S.????	22.02.2017
25	015.D02	Man	015	D	02	om3			0	om3N.????	17.02.2017
26	016.A01	Man	016	A	01	om4	322.803		10	om4N.010	07.02.2017
27	016.A02	Man	016	A	02	om4	322.803		8	om4S.008	28.01.2017
28	016.C01	Man	016	C	01	om4	322.801		10	om4N.010	27.01.2017
29	016.C02	Man	016	C	02	om4	322.801		21	om4N.021	27.01.2017
30	016.C03	Man	016	C	03	om4	322.803		21	om4S.021	27.01.2017
31	016.C04	Man	016	C	04	om4	322.803		10	om4S.010	27.01.2017
32	017.B01	Man	017	B	01	om4	322.801		15	om4S.015	01.02.2017
33	017.B02	Man	017	B	02	om4	322.801		7	om4N.007	01.02.2017
34	017.B03	Man	017	B	03	om3	322.803		19	om3N.019	29.01.2017
35	017.B04	Man	017	B	04	om3	322.803		7	om3N.007	01.02.2017
36	017.D01	Man	017	D	01	om4			0	om4S.????	22.02.2017
37	017.D02	Man	017	D	02	om3			0	om3N.????	17.02.2017
38	018.A01	Man	018	A	01	om4	322.801		9	om4N.009	02.02.2017
39	018.A02	Man	018	A	02	om4	322.803		8	om4S.008	28.01.2017
40	018.C01	Man	018	C	01	om4	322.801		9	om4N.009	27.01.2017
41	018.C02	Man	018	C	02	om4	322.801		20	om4N.020	27.01.2017

(b) Design-to-Production's Excel sheet for managing the produced elements of Shigeru Ban's Swatch building, across the different trades involved in this particular project.

Figure 2.24: Design-to-Production's curated workflow, database model and management tools.

KieranTimberlake's database management interface

KieranTimberlake, an American architecture firm founded by Stephen Kieran and James Timberlake, focuses its practice around sustainable design, energy modelling, and in-depth research and development. Within the practice of KieranTimberlake, the design team systematically needs to capture and track data from various external stakeholders. Matthew Krissel, Partner at KieranTimberlake, has highlighted the difficulty for the architecture firm's designers to gather key information that is often disaggregated amongst the different involved parties:

"Work often begins with a spreadsheet created by the client or the design team, and the data typically remains separate from the building model. The designer has to move back and forth between disaggregated data and software platforms; an inefficient workflow that risks data loss or misinterpretation. As the project evolves, the spreadsheet(s) are often managed by select individuals on the team, making the information even more remote to designers. Yet the data culled in the programming phase from user groups and the design team provides some of the most useful qualitative and quantitative information." (Deutsch, 2017, 97)

Facing this challenge, KieranTimberlake developed an in-house database application (Figure 2.25) enabling better data management and exchange between different software environments:

"[...] Because no third-party software could meet our aspirations for an efficient and generative building programming workflow, we designed a relational database that combines qualitative with quantitative information and is linked to the BIM model. The resulting database gains intelligence over time, allowing the design team to build, organize, and query in more ways than they could with a spreadsheet. The data flows back and forth from the database to the model, ensuring that the design process and the people modeling the geometry are persistently working with live data. [...] It is important to find ways to fluidly transfer information across software platforms to eliminate the redundancy and information loss that occur when operating within a multi-tool design environment like BIM. With this workflow, we set out to establish a dynamic, relational framework where we could ensure that project knowledge is pervasive, evolving, and effectively informing the design. We have since taken this workflow beyond the programming phase to support iterative energy modeling workflows with consultants." (Deutsch, 2017, 97-98).

This particular practice-based example demonstrates the great potential of developing custom applications to answer specific needs, instead of relying merely on the existing and available software platforms.

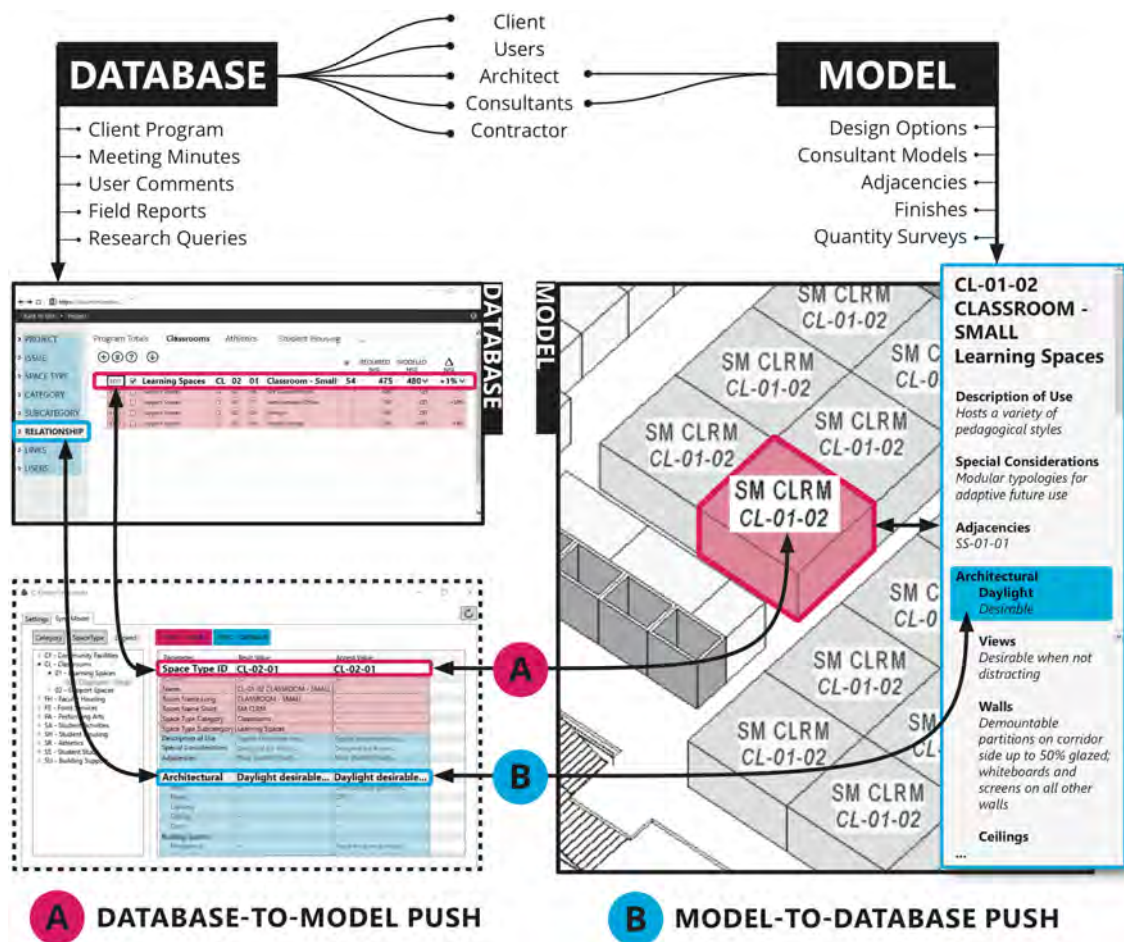


Figure 2.25: KieranTimberlake, *BIM Workflow for Integrated and Informative Programming*, 2016. The architects developed a relational database and linked it to the BIM model, creating a generative building programming workflow that gains intelligence over time. (Deutsch, 2017, 96)

BuroHappold's BHoM open-source platform

As a very large engineering company that occupies 23 offices around the world and brings together more than 1800 employees, BuroHappold Engineering is facing both internal and external interoperability challenges. Because the company operates within different sectors of the building industry, BuroHappold employs engineers specialized in multiple fields; structural engineers, MEP (Mechanical, Electrical, and Plumbing) engineers, façade consultants and architects are very often working altogether on the same architectural projects. Since each specialized engineering field uses specific software for unique purposes, extra work-load is generated to communicate and translate data to other specialists who operate within another set of software platforms. In order to tackle such issue, the Computational Collective of BuroHappold Engineering has undertaken for the past few years a research and development agenda which led to the co-creation of the “Buildings and Habitats object Model” (BHoM) platform, enabling seamless data exchange across the different used software packages. BHoM has been designed as a hybrid model for code architecture - integrating a number of concepts from across existing languages/platforms. Specifically, BHoM has a data structure and data manipulation strategy which is directly compatible with both visual flow-based programming and text-based imperative code. Practically, BHoM offers a neutral schema to define design objects that can be converted to and from various software platforms. This neutral environment proprietary software functionality can be extended by adding and calling generic methods through system reflection.

More than being a simple interoperability tool, BHoM also genericizes the redundant functions terminologies that needs to be redefined across the various software environments by defining a generic language allowing to both describe and call those functions through method injection, in the same manner from any platform. Thus, datasets can seamlessly be sent, analyzed and received back and forth from external structural analysis software packages. The BHoM interoperability data model has been recently released ¹¹ and open-sourced on github ¹². The general BHoM framework consists of an Entity-Component-System (Figure 2.27a) composed of relationships (describing a single, neutral computation graph), behaviours (methods injection), representations (adapter and translations injection), and properties (The BHoM properties). Those different dimensions compose also most of the BHoM objects, described through properties, behaviours and representations (Figure 2.27b).

¹¹<https://bhbm.xyz/>

¹²<https://github.com/BHoM>

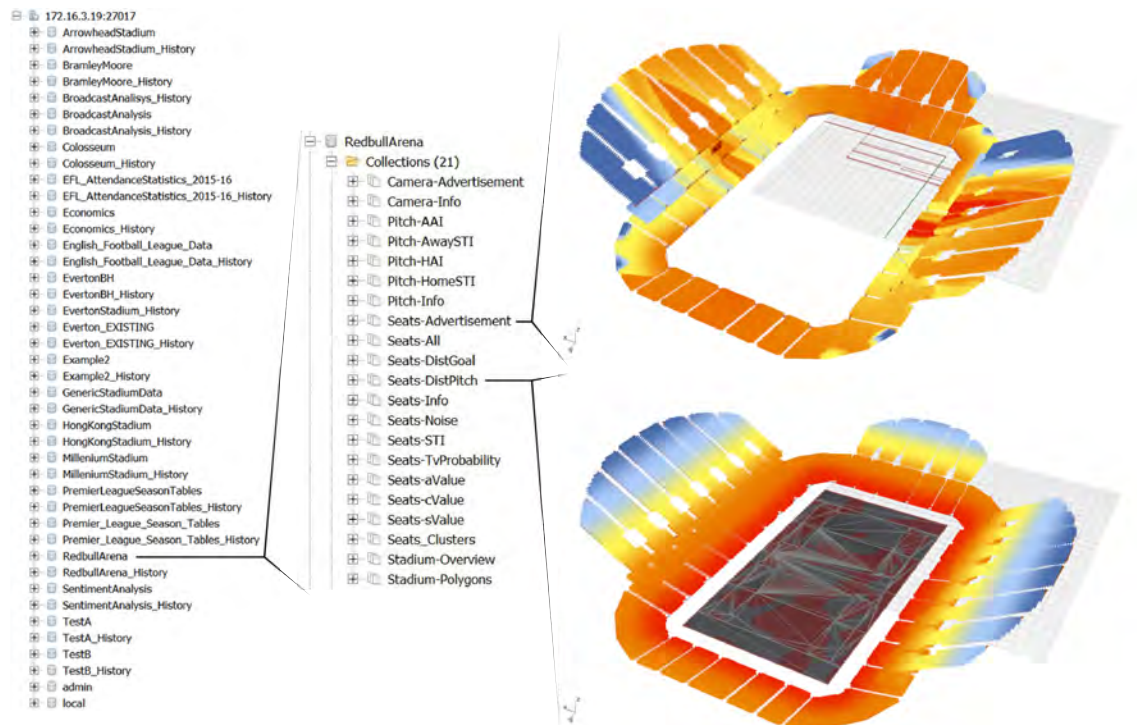
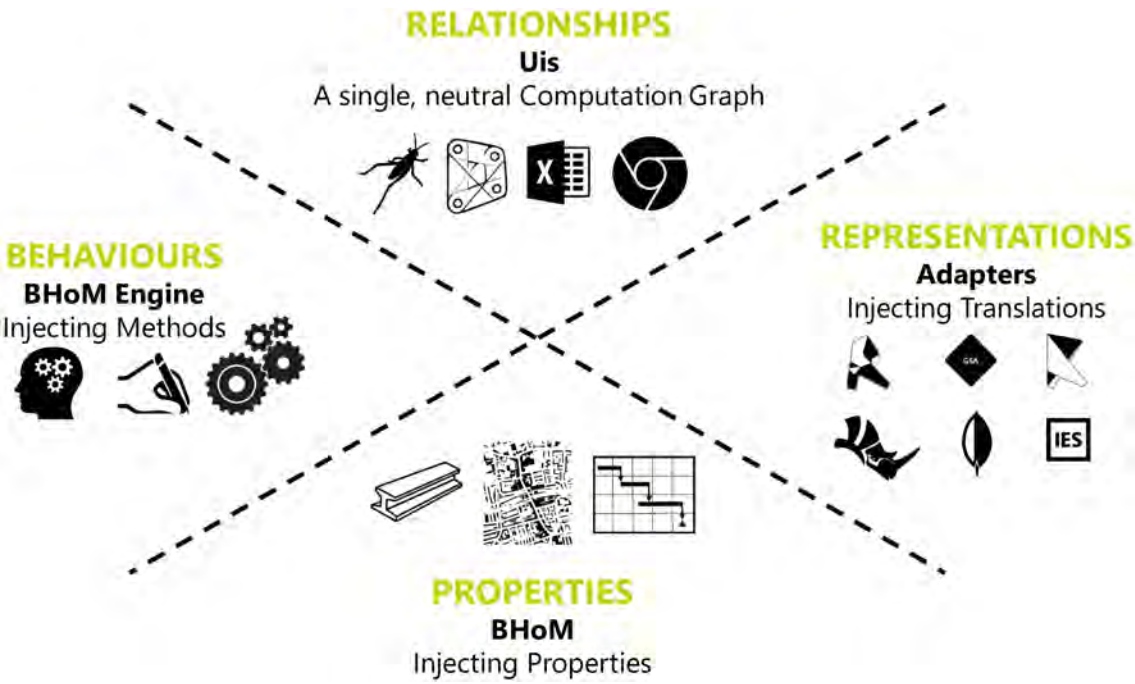
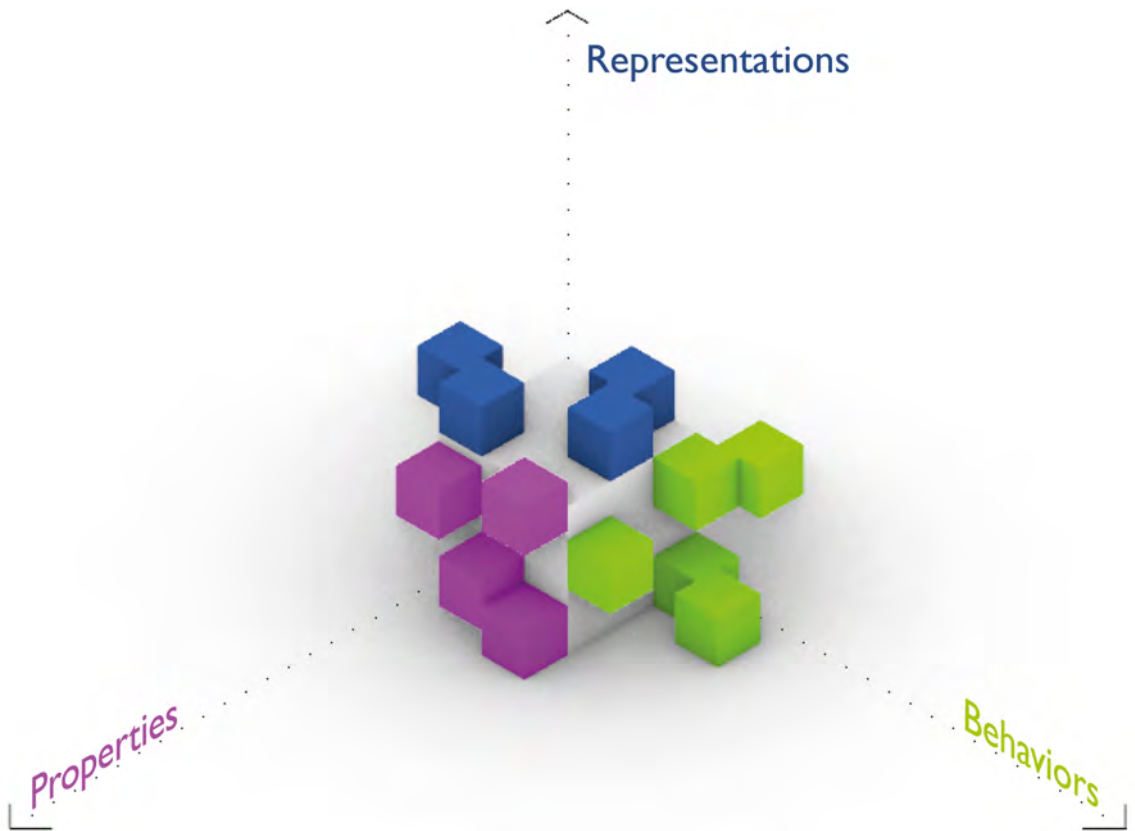


Figure 2.26: Hierarchical datasets for running various analysis of a stadium.



(a) The BuroHappold's Entity-Component-System.



(b) The three dimensions composing a BHoM object.

Figure 2.27: BuroHappold's Buildings and Habitats object Model (BHoM).

Treegram: communicating tree structures and granularity

Treegram | M4D is proposing to structure data from architects and engineers in order to enable more transparency and intuition across software. The modeling processes developed within the company are based on the common concept of *granularity* (Kunz et al., 2014), allowing to keep in memory the operations and different actions that link the different model components. Objects can evolve according to the project phases, integrate more or less details, and adapt to the different production logics. To support this concept of granularity, the company relies mainly on Directory Tree Structures. It is understood that the main difficulty remains in the calibration of the latter: the placing of the nodes, the type and the number of divisions. One needs to perfectly understand the inputs and outputs required by each trade and anticipate the flow of information through the different actors and processes, the loops and reiterations. The rules should remain very simple, able to unfold, as well as generative. For economical reasons and organisational purposes, the tree-like data structure (Figure 2.28a) of the LUMA Fondation project has been conceived as a global entity which relies on a single generic rule (Figure 2.28b). Then comes the question of sharing such structured information. For this purpose, “Treegram | M4D” has developed its own in-house custom tools allowing interoperability between the different software platforms used both internally and externally by the other involved trades.

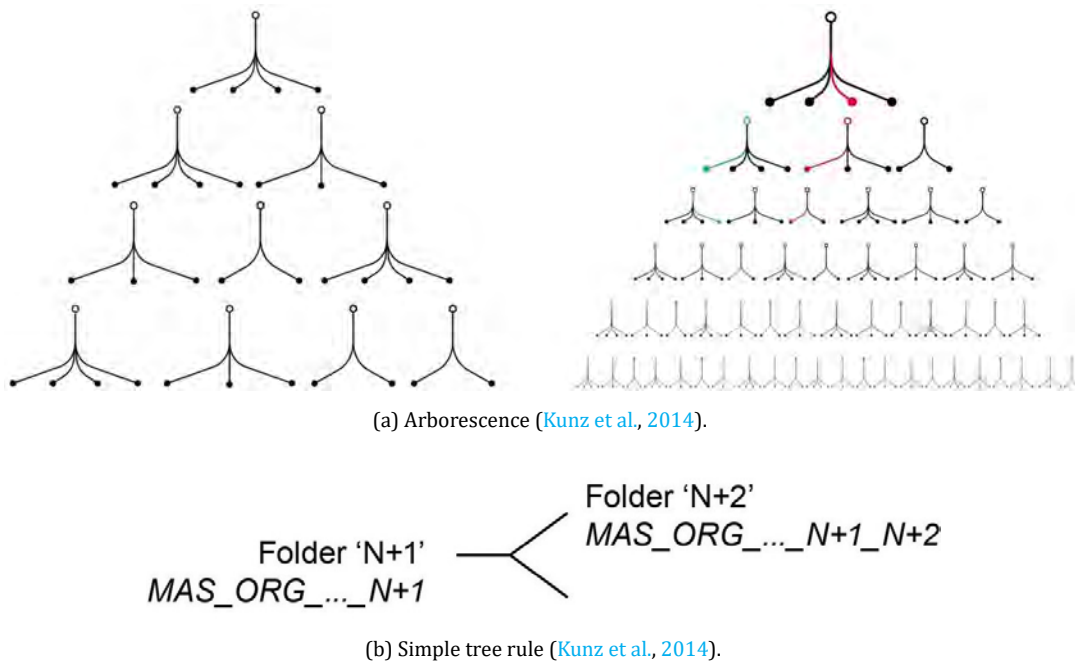


Figure 2.28: Project's arborescence relying on a simple tree rule, propagating across folders and sub-folders.

Woods Bagot's data-driven workflows and client-driven applications

The architecture firm Woods Bagot has been developing internal methods and customized workflows for improving communication and interoperability across trades. During the conception and fabrication process of the South Australian Health and Medical Research Institute (SAHMRI), the company used color-coded strategies, object-naming schemas and other analytical tools in order to keep track of the project's data throughout the whole process (Burger, 2014).

During a lecture delivered on the 14th of September 2017 at the Construction Innovation 2017 Forum held at the Melbourne Convention and Exhibition Centre (MCEC), Shane Burger, Principal and Global Leader at Woods Bagot, stated that the AEC industry “is migrating from a file-based collaboration system to a one that is data-driven, as database across multiple platforms.” Software platforms are not seen here as limiting the design possibilities but rather as an array of tools from which the architect can pick and choose, serving the project's needs to be designed and delivered. Moreover, Burger added that such paradigm allows the company to “keep each platform as pure as

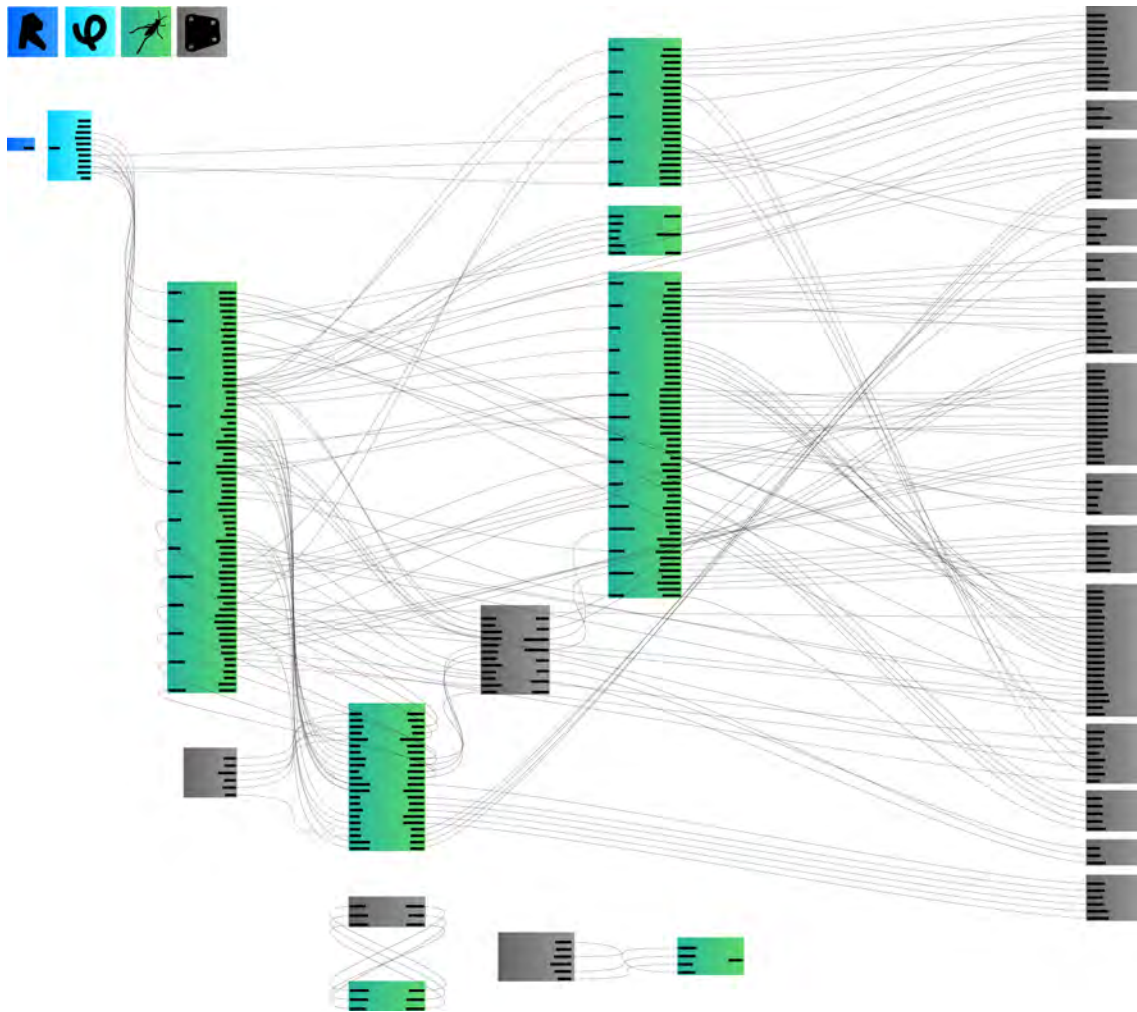


Figure 2.29: The *Metagraph* developed by Woods Bagot represents data key relationships between different scripts from different software platforms, using the Grasshopper canvas with its parameters and wire connections.

possible for what it's suited for, [...] allowing the designers to make them bend to their will in the service of these design and delivery methodologies." Based on this approach, Woods Bagot has developed the concept of *Metagraph* (Figure 2.29), a data visualization tool based on Flux data keys which are used to represent all relationships between the scripts of a same project. The data exchange between different software platforms (Rhino, Grasshopper, Revit and Dynamo) are represented through a decentralized global network of nodes. While being useful for understanding and debugging models at systemic level, it is still being developed to allow for local-level programmatic control. In one of his recent lectures, Brian Ringley, former associate and global specialist at the design technology department of Woods Bagot, speculated that the next step of the *Metagraph* would be to *"set another layer of parametric intelligence where different scripts can adapt and update based on changes of other scripts."* (Ringley, 2017). This last approach is very interesting as it shares quite similar concepts (such as Discrete Event System Specification, Multi-Model and distributed Data Management) with the various Multi-Scalar Modeling strategies for Building Design that I will develop more in depth in the next chapters. Woods Bagot is also making some effort to share their parametric models through the form of re-deployable applications and simple UI to let the client to manipulate the design in real time.

Thornton Tomasetti's CORE studio

Thornton Tomasetti's CORE studio focuses on Research & Development and is constantly developing a set of multiple applications (Figure 2.30) which enable better analysis, data management and communication during both early and late design stages: Konstru enable interoperability between multiple software platforms, Asterisk allows to operate rapid structural analysis at early stages through machine-learned data sets or precedents cases, Thread and Design Explorer are particularly useful for interacting with intricate data sets and navigating across different design options, and FIM (Forensic Information Modelling) Portal enables rapid analysis of BIM models by filtering 3D objects based on their User-Data. Those different applications are continuously under development and get tested both during practice and through the yearly AEC hackathon event held at the office.

Initiated at Thornton Tomasetti's CORE Studio in the development framework of TTX, Konstru is now an independent start-up that presents itself as *"a secure cloud-based platform for engineers, architects, and building contractors to track, share, collaborate, and update BIM data across popular modeling and analysis software tools"*. Unlike Flux Data Inc. (see section 2.4.3) that relied on a multitude of different applications which can communicate between each other, Konstru presents itself as more centralized, integrative platform within which a BIM model can be accessed and modified from a multitude of different external software packages.

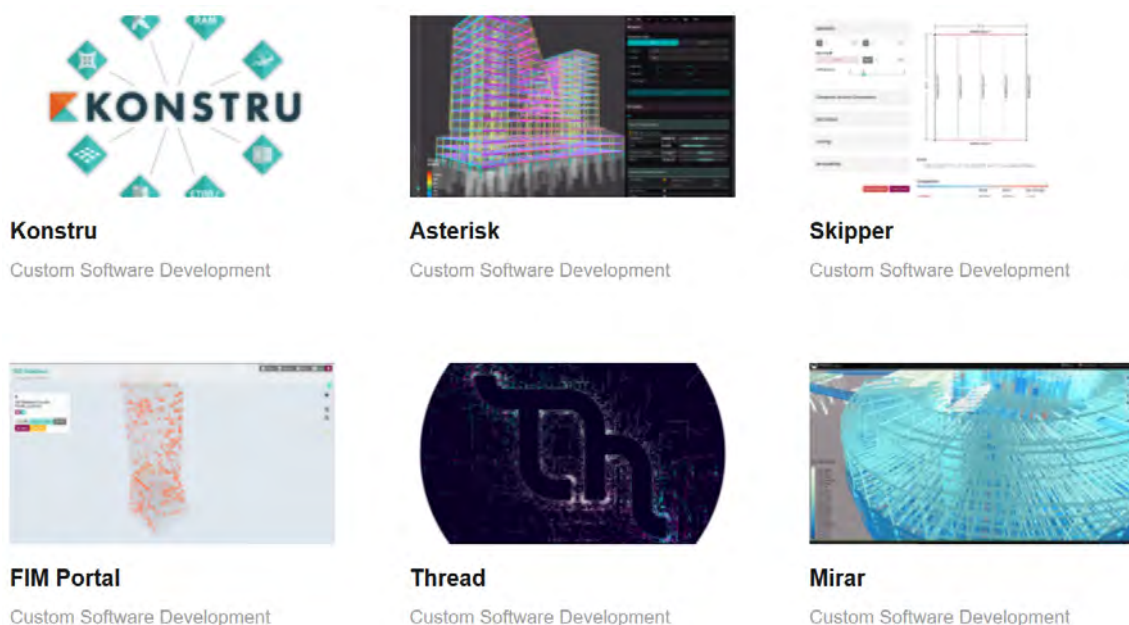


Figure 2.30: Thornton Tomasetti's CORE studio is developing custom applications to improve workflows both at early and late design stages.

Conclusion

In all the industry cases described above, the companies are developing their own tools and workflows, as the currently available software platforms for AEC do not answer their specific needs. Most of the substitute solutions commonly share the following goals and/or features:

- **Improving interoperability through custom development enabling the transfer of data-rich geometrical objects**
- **Keeping geometrical objects as lightweight as possible for serialization and further reconstruction.**
- **Using extensible neutral formats to facilitate both object customization (scaling and nesting of data-rich information) and interoperability.**

These particular answers to the common diagnosed issues are currently brought by a growing number of [AEC](#) practices which are facing unknown challenging design problems and therefore look at technological alternatives to solve them. However, there is no common agreement for setting up an underlying data-exchange infrastructure that would enable better linkage between the different practices, which constantly need to build custom application and workflows for unrevealed design problems. Mette Ramsgaard Thomsen captured and foresaw this very particular need and consequent modelling paradigm shift:

“Rather than building common standards and libraries for known practices, we need to develop the fundamental infrastructures for yet unknown practices. This position fundamentally challenges some of the cornerstones of present modelling paradigms.”
([Ramsgaard Thomsen, 2016](#))

The next section looks at the different existing solutions and initiatives provided by emerging companies or through open-source projects.

2.4.3 Existing meta-software and neutral formats: from a model-driven to a database-driven approach

Because the current models and design processes are isolated and fragmented, data gets lost on the way, translated from a model to another without being traced or saved anywhere. This generates unnecessary duplicate data and residual models that are constantly being “washed” in order to be understood within the next used software environment. Facing such issue, a growing number of practices (engineers, architects and consultants) are starting to change their perspective about the current data/model relationship. If yesterday we believed that the model comes first and its metadata last, we realize today that data should be placed at the core of the design process, from which can be derived many different kinds of models. This paradigm shift would allow the whole [AEC](#) community to query and retrieve just what is needed for a specific task, without creating unnecessary duplicate information. Such paradigm shift has been nicely captured by Brian Ringley:

“At first glance, we are facing the technical challenge of migrating interoperable data sets through a myriad of platforms that all interpret elements differently. What may prove to be a smarter approach is a stable data set through which various models pass through and manipulate.” ([Ringley, 2017](#))

The different cases described in the previous sections also converged towards very similar conclusions: the need for developing meta-software, using neutral data formats in order to exchange geometrical data between different software platforms. During a talk at USC (University of Southern California), Ramon Van der Heijden, Computational Design Specialist at Front during the conception and construction phases of the City of Dreams Casino Hotel in Macau (conceived by Zaha Hadid Architects) mentioned the need for shifting the design process towards a database-centric approach:

“At the beginning of the project, when we started to divide the whole design process, we wanted to start with a database, but it was quite hard to do so while meeting the deadlines. Therefore we decided to store the data in the 3D model itself. Currently, we are rethinking this strategy. At that time, we just had to get the job done, and it was working. I would really like to use a platform or a communication language, to exchange not just data but also geometry on a neutral level. At the beginning, this could work between different software platforms and afterwards maybe within its own platform that could be accessed by anyone.” ([Van Der Heijden, 2017](#))

Today, different solutions exist and attempt to meet the expectations described above. New tools, like [Speckle](#), are mostly focused of web-based data communication. Free tools, such as Salamander (developed by Ramboll), are creating connections between design and FEA structural software such as GSA and ETABS. Some tools, such as Flux, unfortunately aborted, while some others are either in development or fully mature. The next sections will describe each of those existing interoperability technologies and highlight both their advantages and disadvantages.

Flux

Flux Data Inc. paved the way to transfer transfer seamlessly building data across different software platforms. It became popular within the [AEC](#) community and was largely used by many architects, engineers, and consultants before it ceased software development on the 31st of March 2018. This event was unfortunate for a substantial part of the industry, especially for those who have built up their digital workflows upon this platform. Flux had four main management tools that were accessible from the site: “Community” enabled the users to help each other on an exchange platform, “Data Explorer” helped in assessing the project’s workload, “Flow” allowed to visualize the different data flows within the same project, and “Projects” enabled the different stakeholders to keep track and see the current status of the projects in progress.

Depending on the client’s need, the appropriate applications could be downloaded and installed from the main website, enabling better communication between the specific software environments used by the office. If the company used a software platform that was not supported by the Flux applications, it was still possible to undertake third party software development through the available Software Development Kit (SDK). Flux software was being used by “more than 6,200 companies in 151

countries and relied upon by Computational Designers, Engineers and sophisticated BIM professionals at Frank Gehry Partners, BIG, SHOP, Arup, BuroHappold Engineering, Thornton Tomasetti and more". The companies that fully integrated Flux products within their workflows had to look for placeholder solutions such as Konstru (an integrative BIM platform) or Speckle (an emerging open-source data communication platform for AEC).

Speckle: an extensible design and AEC data communication protocol and platform

Contrary to Flux Data Inc. and Konstru which developed their business models through the offer of online services in exchange for a monthly fee, Speckle (Stefanescu et al., 2018) differentiates itself by proposing "an open-source (MIT License) initiative for developing an extensible Design and AEC data communication protocol and platform." (<https://speckle.systems/>) By remaining totally open, Speckle allows anyone to contribute to the platform, from building specific applications for a company's need to the creation of additional generic object types, extending therefore the communication possibilities between different software environments. Philosophically, Speckle believes in the emergence of many different practices which will ultimately lead to a crystallization of the platform itself.

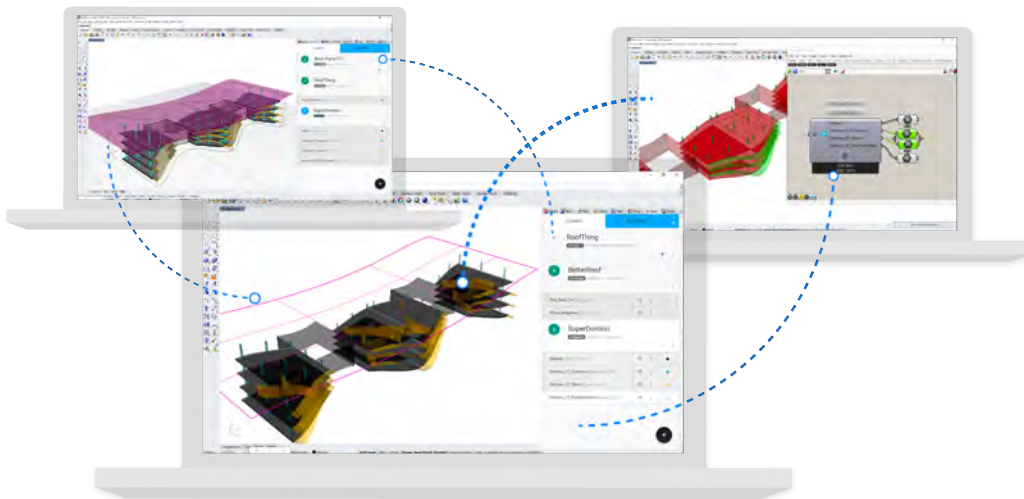


Figure 2.31: Speckle allows to exchange seamlessly geometrical data through a server between different software platforms, such as Rhino/Grasshopper and Dynamo.

Today, Speckle is continuously developed and is currently improving its management interface (Figure 2.32) from which users can share and manage different streams of objects across the different supported plug-ins. Description, tags, comments, team members and streams can be constantly modified according to the project's needs. To maintain privacy and security, streams can be set either as public or private and each team member can access its own account with a log in ID and password. From its own platform, the user has access to the streams and projects that were shared with him/her.

Speckle is developed and maintained by Dimitrie Stefanescu and other contributors.

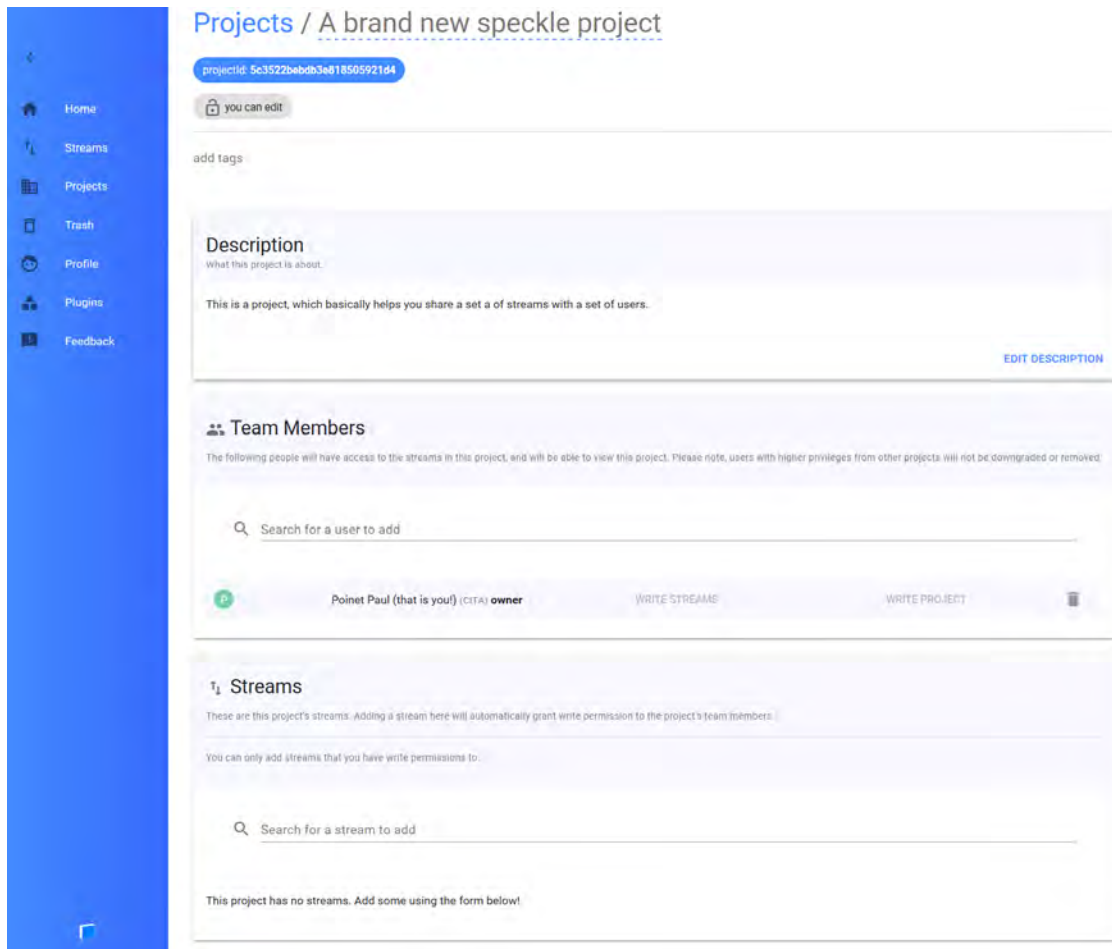


Figure 2.32: The current project management interface in [Speckle](#).

Neutral and open formats: XML and JSON

Generally, the different solutions and tools mentioned above and available today aim at minimizing the technological framework helping in transporting information. The concern of exchanging and communicating with neutral format is also considered and tackled within the automotive industry that needs to constantly pass information between design and engineering software packages:

“Neutral data exchange formats enable the transfer of product data between different software applications. In the case of product design and simulation-related processes, neutral data formats provide product geometry data and limited additional product information for the exchange between different [CAD](#) software packages or between [CAD](#) and CAE applications. Neutral data formats can be divided in geometrically accurate systems, which transform CAD-native product information into sets of mathematical descriptions of the model geometries.” (Hirz et al., 2013)

Neutral formats would further enable the nesting and scaling of data-rich information. Just like *Treegram’s* concept of granularity, the automotive industry is not only interested in sharing solely geometry but also the nested relationships that exist between objects:

“Besides a purely geometrical representation, enhanced neutral data formats are able to include additional facts, such as product structuring information or the configuration of components, modules and sub-assemblies.” (Hirz et al., 2013)

Despite the failure of Flux in maintaining a sustainable business model, the different software frameworks described above show great potential for establishing a future infrastructure allowing

data exchange between the complex network of trades and models during the conception of an architectural project. Such infrastructure would enable the further integration of design interfaces and deployment of Multi-Scalar Modelling, simulation strategies through all phases of large-scale and complex architectural projects. After tackling the topics of communication and transparency on a more conceptual level, the following section is introducing existing industrial design interfaces (such as “Skeleton Models”, “Abstract Networks” and “Adapter Models”) whose inspiring concepts and methods could be further used for setting up working Multi-Scalar Models.

2.5 Towards Transparency, Concurrent Design and Multi-Scalar Modelling

The present section is introducing the topics of Concurrent Design and Multi-Scalar Modelling that could inscribe themselves within the neutral data-exchange infrastructures described in the previous section. Before elaborating on those design strategies, the enabling of communication and the concern of transparency must be addressed, as it is a necessary requirement to further develop and integrate Multi-Scalar Modelling processes.

2.5.1 Enabling communication: connections rather than computation

In “*New Rules for the New Economy: 10 Radical Strategies for a Connected World*” (Kelly, 1998), Kevin Kelly, founding executive director of the Wired, argued that software itself might be overrated and the emphasis on communication would be a more important factor for future innovation: “*The grand irony of our times is that the era of computers is over. All the major consequences of stand-alone computers have already taken place. Computers have speeded up our lives a bit, and that’s it. In contrast, all the most promising technologies making their debut now are chiefly due to communication between computers - that is, to **connections rather than to computations**. And since communication is the basis of culture, fiddling at this level is indeed momentous.*” (<https://www.wired.com/1997/09/newrules/>) Before software, it is people themselves that are the most important asset to solve communication problems between complex systems. This applies both internally (within the same company) and externally (between multiple trades):

- **Internal Communication**

To ease communication, collaboration and interoperability in-house, and to enable a continuum of participation from entry level computational skills to advanced development, BuroHappold has created the Computational Collective, aiming at gathering computational knowledge from all employees and developing a series of Computational Competencies (Figure 2.34). The employee’s computational skills is categorized here into 4 categories, from novice (appreciation of Visual Programming) to expert (expert knowledge of text-based programming). The first category is expected to have basic notions of parametric modeling. The second category must know how to handle visual scripting properly (Grasshopper and/or Dynamo). The third category must be fluent with at least one programming language and be comfortable with scripting within the software’s scripting interface. The fourth and last category is expected to be able to use Visual Studio, be able to develop its own plug-in and contribute to the company’s software development strategy (Buildings and Habitats object Model - BHoM). The skill level of a person plays therefore an important role in defining their ability to work and communicate with other employees that have (at least) the same skill set. In order to get a better overview of the different skill sets and the location of each individual from the Computational Collective, an online UI (Figure 2.33) was developed by BuroHappold in order to filter and find very quickly who knows what and in which location of the world. On the left side, highlighted in blue, the different software used in the office are hierarchically classified from the main category it belongs to, to the particular piece of software itself. On the right side, highlighted in green, the specificities of each individual are mapped as well in a hierarchical manner, from their main area of expertise (e.g. MEP or structural engineer) to their respective location (e.g. London, Bath or Hong Kong) and their respective names. Finally,

Level	Description	Knowledge	Experience	Ability
1	Visual Programming – Entry	Appreciation for the potential for computational engineering and design, and process automation and optimisation using visual programming tools such as Grasshopper and Dynamo. Have an understanding of parametric design in the context of engineering.	Have experience of creating very simple visual programming scripts to generate or manipulate calculation or geometry models. Able to open and understand more complex scripts authored by others.	Be confident in the application of visual programming on projects and tasks to generate or manipulate calculation or geometry models using tools such as Grasshopper and Dynamo. Be willing and able to teach others the lessons learned.
2	Visual Programming – Experienced	Appreciation for the application of advanced visual programming to solve problems beyond a single discipline, user or model. Understand the importance of sharing and collaborating on scripts.	Have experience of creating complex visual programming scripts that perform multiple functions. Understand data structures in the context of Grasshopper data trees, or Dynamo laced lists.	Be able to create visual programming scripts and definitions for use by others, by writing robust solutions that are shared between small groups and a small range of expected behaviour is predicted and accommodated. Be willing and able to teach others the lessons learned.
3	Programming – Entry	Appreciate the benefits of moving beyond visual programming toward scriptable components and definitions in order to provide functionality and process efficiencies beyond the commercially/currently available definitions included in packages such as Grasshopper and Dynamo.	Have experience in writing macros and short scripts to extend the functionality of software such as Excel, Grasshopper and Dynamo.	Be able to write extensive scripted processes and macros within Grasshopper, Excel or Dynamo to extend their functionality, and link more than one script or function together as part of a larger application or process. Be willing and able to teach others the lessons learned.
4	Programming – Experienced	Appreciate the limitations of single-user programming and scripts and the potential benefits of migrating single functions or processes to a centrally managed repository of code. Understand how Visual Studio can be used to author and manage larger more complex scripts. Understand C# programming and object oriented programming, and associated benefits over functional programming.	Have experience in writing, compiling and debugging simple programming scripts and C# class libraries in Visual Studio. Understand dependencies, assembly referencing and the importance of documentation and version control. Have access to and understand the functionality of GitHub as the central code repository.	Be able to contribute meaningfully to the central code repository and the BHoM and associated application Toolkits and plugins. Be fluent in C# and object oriented programming languages, and understand the importance of other programming languages in the context of the BHoM and other code libraries. Understand GitHub and associated applications well enough to become an administrator on a repository. Be willing and able to teach others the lessons learned.

Figure 2.34: Computational Competencies matrix as defined by BuroHappold Engineering as part of Learning and Development for technical staff.

by such systems. It may well be that wisdom (which is essential for the pursuit of ideals or ultimately valued ends) is the characteristic that differentiates man from machines.” (Ackoff, 1989)

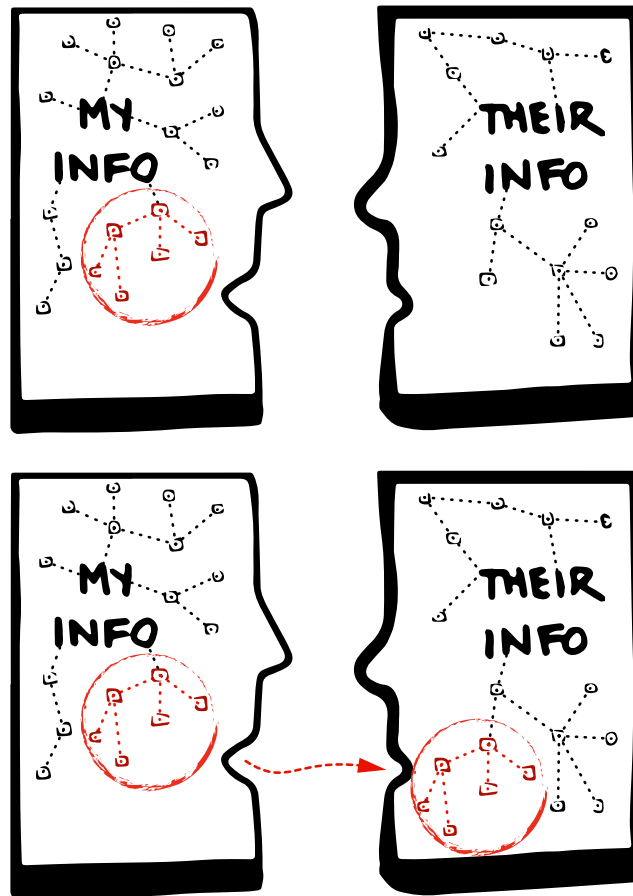
This conclusion follows the establishment of the Data-Information-Knowledge-Wisdom (DIKW) pyramid, a hierarchical concept that makes the distinction between raw data and its gradual transfer into the “wisdom” state that enables full understanding from the designer/user (Figure 2.36a), and therefore the possibility to reuse the received data meaningfully later on within the same (or another) process.

During the conception and realization of a large-scale and complex architectural project, a big amount data is usually left aside within different models without never being fully exploited, either because the sender did not make any effort in articulating implicit information (Figure 2.36b) that lies within explicit data structures or because the receiver does not trust the generated data and prefers instead to reinterpret, reconstruct it, at the cost of creating duplicate information and “residual models”.

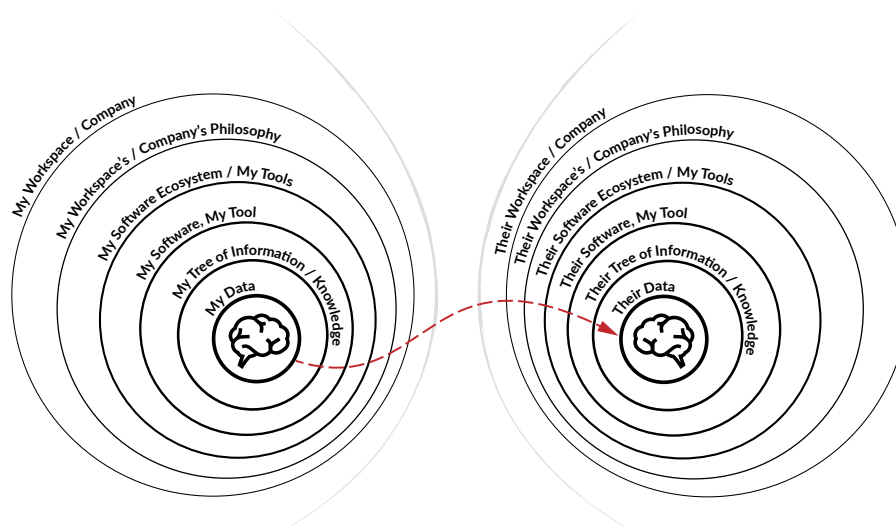
2.5.2 Skepticism towards Transparency

In a scenario where full interoperability is enabled and where the many different trades can seamlessly communicate complex data sets that are part of larger tree-structures through different software platforms, then comes the question of transparency: who owns the data and who controls it (Deutsch, 2015). This particular concern has been faced and discussed by Lea Sattler and Thomas Maigne (Treegram | M4D), façade consultants for the LUMA Fondation designed by Frank Gehry:

“At that time, one of our discussions with Gehry Partners about the BIM systems development for the Luma Fondation was striking. One of their requests was to be able to access the entirety of the digital models produced by all the companies involved in the project. When we told them that we managed to set up an open model without any filters for validation (meaning that they could stream and access a gigantic and collaborative Work-In-Progress model), things got complicated... The architects stepped back: they wanted to be able to open a clean model, not a “draft”. In other terms, they wanted to re-establish a sort of feudalism: the information should be delivered to them without having to access themselves the information. They added themselves a bureaucratic “gate” and thus prevented to enable the seamless information flow that they actually needed : a contradictory desire to control everything without however getting their hands dirty.” (Sattler, 2018)

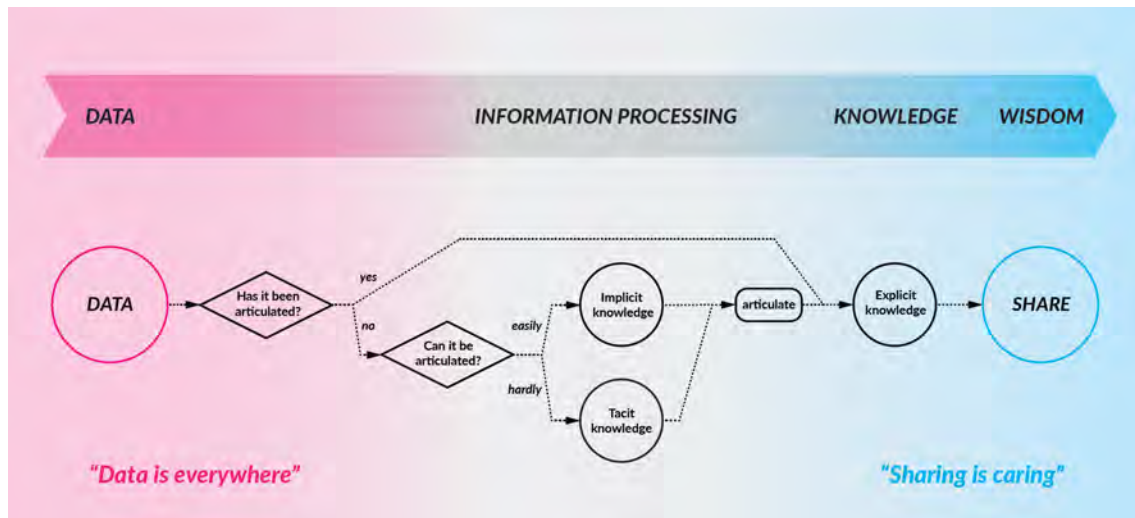


(a) Reinterpretation of the diagram "My info, their info" (<http://abbytheia.com/2016/06/15/information-is-made-of-people/>). Ideally, the sender user would be able to communicate their personal structured data (tree of knowledge) and integrate it seamlessly within the receiver user's own information directory.

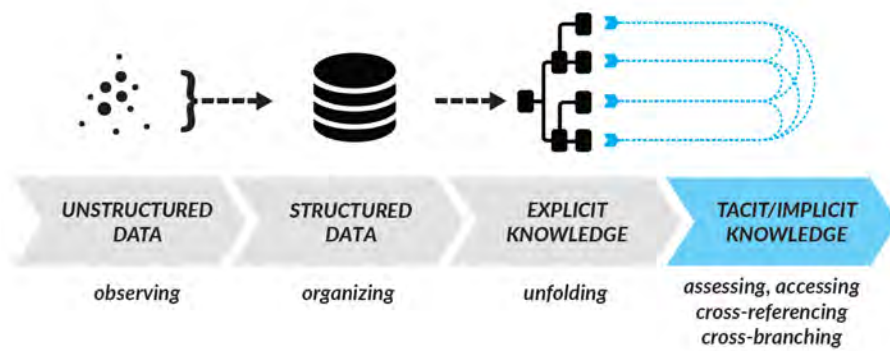


(b) Communicating through different layers of understanding

Figure 2.35: Communicating information between people and trades.



(a) This diagram presents the author's personal interpretation of a DIKW pipeline, where data (or tacit/implicit knowledge) is articulated into through information processing in order to be rendered explicit and therefore sharable.



(b) Tacit knowledge lying inside explicit knowledge.

Figure 2.36: Gradual transfer of knowledge, from implicit to explicit.

This scenario shows that not everyone would vote in favor of transparency, even if this costs the loss of better interoperability between models. Some companies might prefer to exercise a bureaucratic control over highly curated and contracted data. In that case, the architect does not want all the information that all collaborators have access to, but only what is relevant for him/her, without having to deal with an unhinged model, a live view of what others are doing and modifying in real time. Preferably, the architect would like to “disconnect” from the live model, allowing him/her to work in a more controlled manner at various levels of detail. This resonates with an observation made by Dennis Shelden, observing that the building industry often privileges control over collaboration:

“Building has a history of a hierarchically structured organisational model and supply chain that stems from the ‘limited bandwidth’, or limited capabilities for the transmission and processing of information, of the predigital age, and suggests a preference for ‘command and control’ over collaboration.” (Shelden, 2006)

Such perspective on the design process would obstruct transparency and prevent Concurrent Design and Multi-Scalar Modelling. In order to keep full transparency, full trust should be ensured between the different stakeholders. The lack of trust in data manipulation from external trades has been highlighted by Nathan Miller:

“Ultimately the benefits of data interoperability among the larger team hinge on one factor that is also the most challenging to establish: trust. [...] Liability concerns can keep useful digital assets locked away from other participants. In some instances, it is often the preference of the receiving party to discard and rebuild entire models. The reason: They outright assume the data is incorrect. The result: restarts, rework, and data loss.” (Miller, 2010)

Highlighting the lack of trust, the author goes further by observing that interoperability problems are more of a social nature than a technical one: *“[...] in my experience some of the more challenging problems of interoperability are less about technical limitations present in a user’s software stack: often times larger problems stem from skills gaps, poor project planning, and a lack of trust in data from others.”*

2.5.3 Collaborative, concurrent design and common interfaces: skeleton models, adapter models and abstract networks

If full transparency is enabled by a technological framework allowing all stakeholders to exchange data seamlessly, different concepts such as “Concurrent Design”, “Concurrent Engineering” and by extension “Multi-Scalar Modelling” can finally be enabled. Concurrent Design/Engineering is understood as an evolutionary step from Collaborative Design by bringing advanced data management and concurrent authoring, subdividing the design process into smaller tasks (as the previously mentioned Separation of Concerns) that can be undertaken by different persons or trades in parallel. It is crucial here to define and agree on the “points of contact” and “handshake” protocols from which the information can be sent and received, triggering subsequent computational workflow generating data for other models:

“One important factor in concurrent design is the definition of design interfaces, such as flanges, adapter geometries or the implementation of skeleton models.” (Hirz et al., 2013)

If such issues are not addressed properly, Concurrent Design may not work effectively, since it requires that computer models are exchanged efficiently, something that can be difficult to set up properly in practice. The key to design collaboration is the definition of common interfaces, allowing multiple trades to collaborate on the same product. Again, the automotive industry provides insightful design methodologies to operate with such interfaces: “Skeleton Models” and “Adapter Models”:

- **Skeleton Models**

Skeleton models represent the minimum amount of information (wireframe elements: points, lines, planes, etc.) to generate more intricate geometrical datasets, and can serve as a point of contact to control two (or multiple) different (sub-)models:

“In the present example, the engine centerline and several reference planes are imported into the crankshaft model to serve as indicators for the design process. In this way, the positions of significant components of the crankshaft can be directly controlled by the crankcase model.” (Hirz et al., 2013, 303)

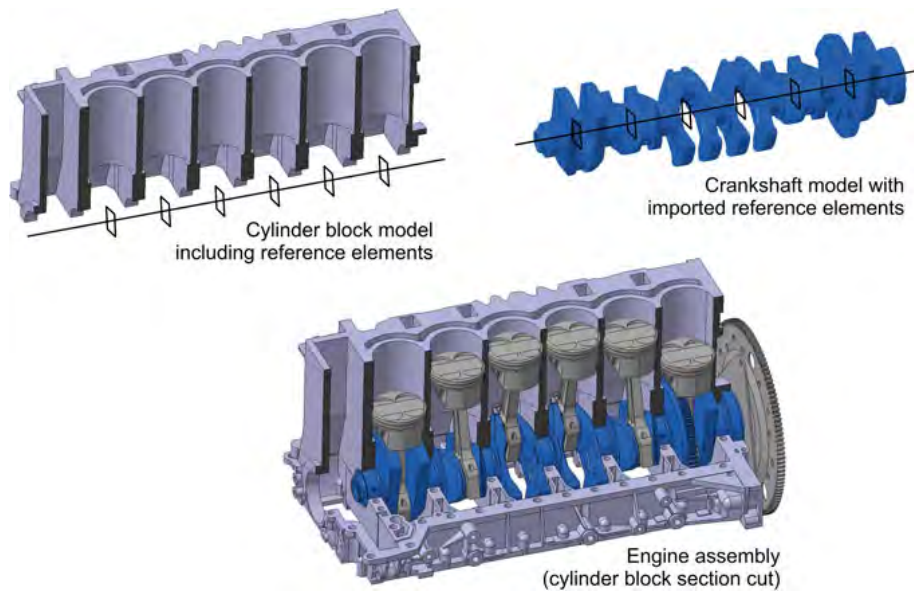
Figure 2.37a illustrates a skeleton model that links a cylinder block model with a crankshaft model, enabling the final engine assembly.

- **Adapter Models**

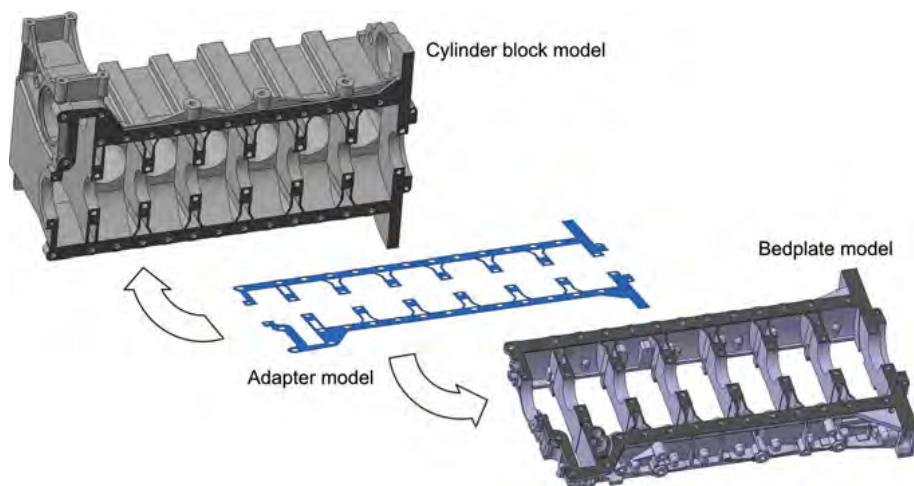
Adapter models are similar to skeleton models but they also embed associative modelling behaviour between the different interfaced components:

“Similar to skeleton models, adapter models are separate components in assembly structures and enable an efficient control of geometrical information in several parts. However, unlike skeleton models, adapter models do not define the positioning of components, but rather have direct access to the geometrical characteristics of the components.” (Hirz et al., 2013, 304)

This methodology is illustrated in the following example (Figure 2.37b), within which “[...] the adapter model includes the geometrical information of the sealing flange for the crankshaft bearing unit. This geometrical information is provided to the two concerned components of the bearing unit, the cylinder model and the bedplate model. In the assembly, the lower flange of the cylinder block is bolted to the upper flange of the bedplate. The adapter model is positioned in between these two components and defines the geometrical extension of the flange. Due to the parametric-associative linkage of both concerned flanges with the adapter model, the flange geometry of the cylinder block and the bedplate fit perfectly to each other. Modifications to the flange geometry are made in the adapter model, which leads to adjustments in the corresponding components.” (Hirz et al., 2013, 304-305).



(a) Skeleton model of an inline 6-cylinder engine assembly.



(b) Exemplary application of an adapter model in engine design.

Figure 2.37: Description of Skeleton and Adapter models. Source: M. Hirz et al., *Integrated Computer-Aided Design in Automotive Development* © Springer-Verlag Berlin Heidelberg 2013, 2013.

Both the Adapter and Skeleton Models show great potential to interface multiple models shared across different trades.

Management interfaces and versioning

Finally, once transparency and the Adapter/Skeleton Models are enabled, custom management interfaces have to be yet configured in order to enable better collaboration between trades and coworkers. Large-scale and complex architectural projects involving an heterogeneous landscape of contractors necessitate a collaboration framework that can handle such level of complexity and allow users to navigate across it in an intuitive manner. Recent research at Autodesk has been investigating appropriate management interfaces through the development of custom applications such as EgoLines, allowing egocentric analysis of dynamic networks. EgoLines takes into account that “many of these networks are dynamic, i.e., the topology of a network and/or the attributes of its nodes and links vary over time, revealing relationship dynamics in real-world systems.” (Zhao et al., 2016, 5003). This is specifically the case of AEC networks of trades that keep evolving during the conception and realization of an architectural project by either entering or exiting it (Figure 2.15). EgoLines employs a “subway map metaphor” to reveal the “evolutionary patterns of a dynamic ego-network. Each actor (the ego and alters) in the network is represented as an actor line from the time step when the actor first joins the network to the time step when he/she last appears, where dashed lines indicate temporary absences. [...] It allows a user to identify the lifespan of an actor in the ego-network and pinpoint the events of entering and leaving the network. Users can also observe the stability of an ego-network by viewing the lengths of the lines. For example, a large number of shorter actor lines indicates that there are frequent turnovers of alters in the ego-network.” (Zhao et al., 2016, 5004). Such subway map would be particularly useful to track down collaboration between trades throughout a whole project.

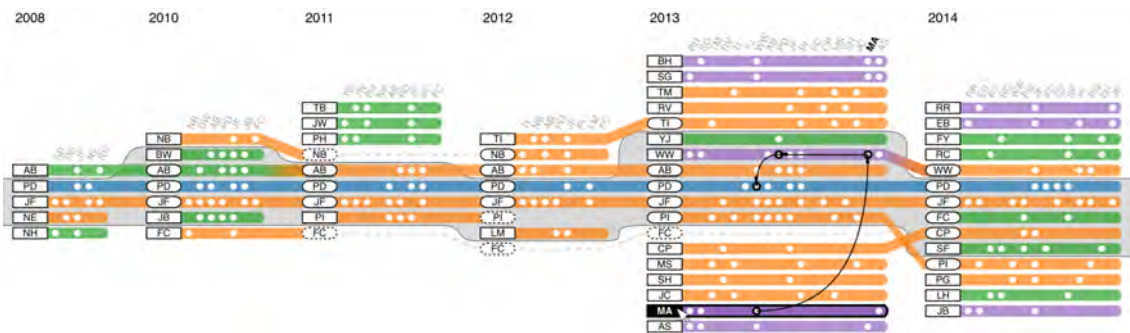


Figure 2.38: A dynamic network of email communications among multiple employees (Zhao et al., 2016).

Starting from the observation that “handoff of partial findings is challenging because externalizations produced by analysts may not adequately communicate their investigative process.” (Zhao et al., 2018), other research undertaken at Autodesk looked more particularly at GUIs for articulating implicit information exchanged during asynchronous collaboration, in which many different actors can contribute through comments, annotation and the sharing of other relevant metadata. Taking as a case study investigative processes, a Knowledge-Transfer Graph (KTGraph) has been developed, facilitating “the handoff of partial findings to subsequent investigators in asynchronous collaborative sensemaking. Specifically, [the] techniques support handoff through explicit user-annotations and implicit playback of the analytic process.” (Zhao et al., 2018). For example, KTGraph offers the possibility to rewind the investigative process through an interactive time-line from which the user can retrieve a particular state of the investigation, along with the related graph, communications and comments. In the realm of AEC, interfaces like KTGraph would be particularly helpful in the context of Multi-Scalar Modelling for commenting information and exchange metadata across networked models and interactively collaborate across the different software platforms and trades.

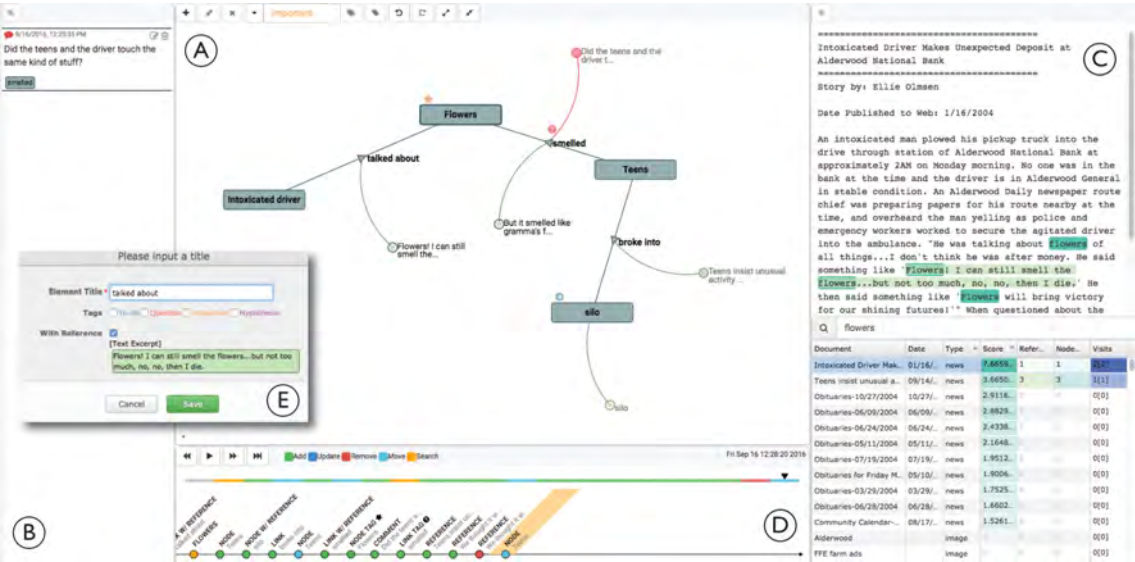


Figure 2.39: Knowledge-Transfer Graph and historical time-line of an investigative process (Zhao et al., 2018).

2.6 Conclusion

This present chapter has described and discussed the history and state of the art of both [GUI](#) (Graphical User Interface) – to better manage complexity – and interoperability – to better share data – outside and within the [AEC](#) (Architecture, Engineering and Construction) realm. Section 2.2 focused more particularly on the current practices of parametric modelling in [AEC](#) and highlighted the related concerns and challenges encountered by this field. Based on these current practices, section 2.3 described existing proposed solutions developed by leading architecture, engineering and consultancy practices to both scale up geometrical complexity and improve interoperability. Generally, the adopted strategies are relying on a Separation of Concerns ([Dijkstra, 1982](#)) ([Pena De Leon, 2014](#)), consisting of segregating the design workflows and activity networks ([de Vries, 1995](#)) into smaller processes, staged models ([Van Der Heijden et al., 2015](#)) or design cycles ([Tessmann, 2008](#)). Based on initial observations and insights, section 2.4 emphasized the need for improving interoperability and section 2.5 pointed out the need for more neutral format exchange protocols, transparency, and the development of custom [UI](#) (User Interface), [UX](#) (User Experience), communication and search interfaces. As it has already been suggested by Nathan Miller:

“By developing and coordinating different communication techniques, the hope is to create opportunities for seamless collaboration regardless of the interfaces or modeling tools different participants would be using in the design process.” ([Miller, 2010](#))

It has also been shown that existing concepts – such as the “*Skeleton Models*” and/or “*Adapter Models*” mentioned in section 2.5.3 – borrowed from other industry-related fields (such as the car industry for modelling complex systems and the computer science field for managing complex data sets) could be great help in solving and improving current digital design workflows of complex and large-scale architectural systems.

If both transparency and common interfaces are clearly defined and fully enabled, simulation could finally enter the design process. Trades would be able to seamlessly communicate through common interfaces, transport geometrical information from a modelling environment to a structural analysis software, run a simulation, and obtain some feedback from it, which could further re-inform the initial defined design. This is exactly where the topic of Multi-Scalar Modelling is introduced and will be further elaborated in the next chapter by investigating current research in academia, before proposing design methodologies to transfer and implement Multi-Scalar Modelling within [AEC](#) practices.

Chapter 3

TOWARDS MULTI-SCALAR MODELLING FOR BUILDING DESIGN: A THEORETICAL FRAMEWORK AND RESEARCH METHODOLOGY

“The multiscale, multi-physics viewpoint opens up unprecedented opportunities for modeling. It opens up the opportunity to put engineering models on a solid footing. It allows us to connect engineering applications with basic science. It offers a more unified view to modeling, by focusing more on the different levels of physical laws and the relations between them, with the specific applications as examples.”

E Weinan

Multi-Scalar Modelling: one paradigm, different definitions and methodologies

This chapter will discuss and introduce the interdisciplinary concept of Multi-Scalar Modelling, which is also a research field concerned about the simultaneous description of a system through different scales and models (E, 2011). The concept behind Multi-Scalar Modelling (MSM) has been largely used in many different scientific domains – e.g. in Computational Fluid Dynamics (CFD) (Wang et al., 2010), in Structural Topology Optimization (Xia and Breitkopf, 2015) or in Phase-Change Materials (PCM) (Faircloth et al., 2018) – that encounter the problem of modelling and transferring information across multiple scales. It is recognized that the conceptual development of Multi-Scalar Modelling comes from the realization that the full behavior of a system cannot be represented within one single model since the level of detail differs from one scale to another (Figure 3.1). Maintaining full detailed resolution through all the levels of the system becomes most of the time inefficient and computationally expensive (Nicholas, 2016). Because MSM is a general concept that does not belong to a one field in particular, it is not referring to any specific tool or predefined methodology. Even though there exist a lot of precedents in the development of MSM tools, those have always been developed and used within specific contexts and research areas (CFD, FEA, etc.). It is therefore up to the researcher, based on its own domain of expertise and the related existing precedents, to redefine its own particular MSM methodologies and the tools that he or she can employ for the specific tasks he wants to achieve.

The present chapter is divided into four sections. The first one will attempt to understand the origins of Multi-Scalar Modelling in the context of architectural design research and draw its state of the art by focusing on different built demonstrators and highlight their inherent employed methodologies. Then, the second section will focus on the current discrepancy that exists between academia and practice – two realms that usually do not focus on the same research questions and problematics – and explain why the current Multi-Scalar Modelling methodologies used in academia need to be revisited and extended in order to take into account the realities of building design (which have been described in Chapter 2). From those findings, the third section will list these revisited methodologies

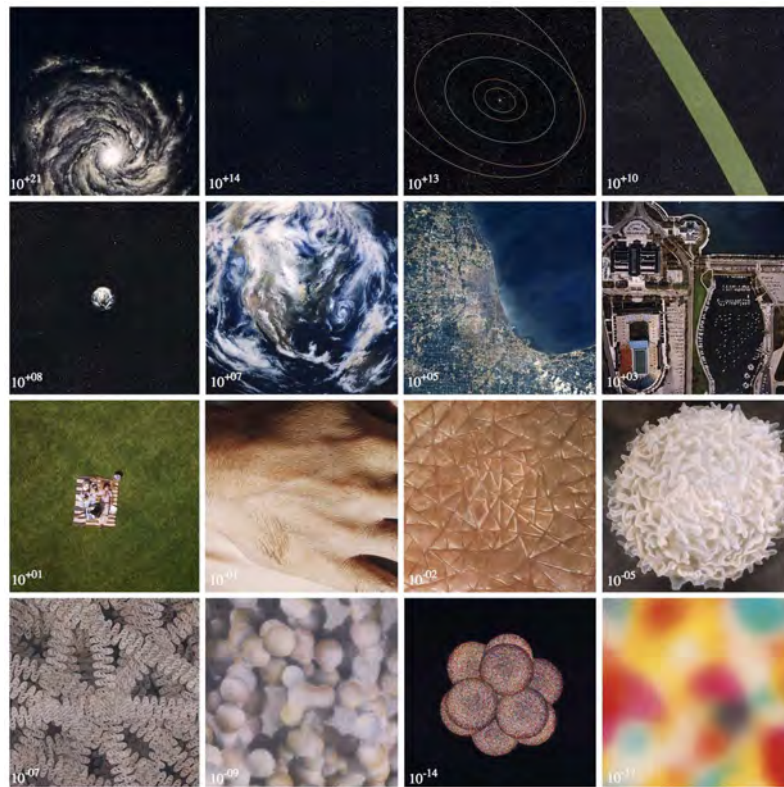


Figure 3.1: Powers Of Ten, Charles and Ray Eames, 1977

and the new ones that need to be considered for industry related concerns. Finally, the last section of this chapter will describe the research methods that have been adopted to carry the present thesis work, which had to be balanced between inputs from academia and industry secondments.

3.1 Multi-Scalar Modelling in Architectural Design Research: Origins and Existing Methodologies

3.1.1 Learning from the cybernetic precedents: the importance of interfacing, usability and front-loading enabling algorithmic and design thinking

A Multi-Scalar Modelling framework is first and foremost the result of a formulation of thought by the main designer(s), impacting drastically the final design results. It has already been suggested that formulating thought is more crucial than the processor speed during computational design process:

"[...] to address the conceptual fluidity so crucial to the early design stages. For this to happen, computation has to become more integral to the design process and become faster, not in terms of processor speed, but in ease of formulating thought." (Kilian, 2013).

"Formulating thought" refers here to the notion of "front-loading", whose importance has been mentioned earlier by Rick Smith (Section 2.2.2), in order to anticipate changes later on in the design process.

When facing a specific geometrical problem, the computational designer might have multiple ideas on how to proceed, and will come across numerous design modelling decisions in order to reach the final output. Each choice will have a small (but nevertheless important) impact on the result, which might almost be as valid as another one regarding specific constraints, e.g. fabrication economics and structural performances. Therefore, the final decision won't be purely objective, instead influenced by

different subjective design decisions along the process. Three important steps are defined here in the context of computational design thinking and algorithmic modelling. All these steps have substantial impacts onto final Multi-Scalar Model, and it is therefore important for the computational designer to evaluate each of them very carefully in order to secure a consistent Multi-Scalar Model that is able to span across multiple scales:

- **The Design Intent (What is modelled?)**

The design intent is defined right at the beginning of the design process. It presents the main guidelines that will impact the final form. The design intent can be drawn by hand, conceived numerically, or simply imagined. Moreover, the designer could also reflect upon the further modelling methodologies and implementation to adopt, and this consideration would already affect the design intent itself by refining its scope.

- **The Modelling Methodology (How is it modelled?)**

The modelling methodology is defined here as the particular process of representing and discretizing the design intent into different objects, parts and meshes that can aggregate together to form a whole. During modelling, the designer has to anticipate the next step which is the implementation strategy. The way in which the different objects are bound together might influence already on the modelling methodology to adopt.

- **The Implementation Methodology (How are the models linked together?)**

The implementation of a design modelling methodology, as the particular data structure – or organisation – of the script/program, influences the way that the computational designer will interact with the design outcome. The final modelling pipeline or workflow defines a specific design space with a certain degree of flexibility let to the user (e.g. the number of parameters and sliders, the degree of interactivity and the User-Interface operability). Therefore, the way in which a design methodology is implemented has an important impact on its flexibility and on its search space at a higher level of resolution.

During those different steps and constant algorithmic thinking, the computational designer strongly interacts with the machine. This growing relationship influences the future usage and flexibility of the final implemented digital design workflow. Such relationship between the human and the machine (or design object) has been already described by several architects and researchers:

- **Kiesler's Vision Machine (1939)**

Best known for his *Endless House* project of the 1950s, Frederick Kiesler also developed the concept of *Vision Machine* during his directorship at the *Laboratory for Design Correlation at Columbia University*. He imagines a sort of electrical transmission between the man and the design object and wonders if we actually "*see in a two-way system*" (Figure 3.3a, top). In other words, Kiesler already reflected on how the design object can influence back onto our own modelling decisions. This resonates strongly with Marshall McLuhan's famous quote: *We shape our tools, and thereafter our tools shape us.* (McLuhan and Fiore, 1967)

- **The Yershóv Diagram (1965)**

A.P. Yershóv described the "director-agent model of interaction" where man and machine can collaborate within a closed feedback loop. The machine is interpreted here as a closed black box with only two portals allowing both information inputs and outputs (Figure 3.3b, bottom). In the context of encoding information within a Multi-Scalar Model, the machine should evolve into a more open system within which the designer can constantly modify, tune and refine data until acquisition of satisfying results.

- **The "Edit-Compile-Run" Cycle**

Today's programmers are now familiar with the *Edit-Compile-Run* sequence of events during coding sessions and interaction with the design interface and outputs (Davis, 2013, 159). Using *Yeti* (an interactive programming plugin for Rhino developed by the author), Davis emphasizes the need for a more interactive design process where changes operated in the code editor can influence directly on the design output. This somehow echoes with A.P.

Yershóv's following remark: *"In other words, if we want the machine to become (paraphrasing the well-known expression) a continuation of the human brain, we must install into the organization of the machine and the system of programming the same flexibility and plasticity which distinguishes the human brain."* (Yershóv, 1965)

It can be observed here that the theories surrounding the man-machine interaction have evolved from the idea of a simple closed input-output transmission to a more open system within which the machine is flexible enough to accommodate any kind of modification wished by the human mind. This aspect is crucial in the setup of a Multi-Scalar Model which needs to be open and flexible enough in order to adapt to late changes. The next paragraph highlights once again the importance of front-loading – already mentioned in section 2.2.2 – from a modelling process perspective, during which the computational designer needs to carefully review each implementation step that tend to progressively constraint the design space, decreasing therefore its flexibility.

The importance of front-loading

After (if not during) the completion of a particular algorithm, the computational designer fully realizes the *design space* and its limits described by the particular set-up of constraints he implemented at the beginning of the programming phase. Even if minor changes are possible, they remain *parametric* and affect very superficially the underlying hierarchy of the overall design system in which the implemented data-structure's (or graph's) typology is kept intact:

"Smooth variation is usually only local in a model with many variables. The geometry itself quickly becomes a complex system as the size of the graph of relations grows, so that while its variability may be predictable for changes in a single variable over a limited range of values, that range varies with changes to other parameter values." (Burry, 2007)

In contrary, major changes can't be made (and if they should be, a new algorithm needs to be re-written). This paradoxical inflexibility (Figure 3.2) can limit potentially the designer's creativity due to either lack of time, or simply lack of desire to rewrite a major part of the script. It has been shown that many experienced computational designers already encountered the problem of dealing with a geometrical exception within a continuous parametric environment and had to face the moment where the most (if not the totality) of the script/program had to be redone from scratch (Davis, 2013). This resonates strongly with Donald Knuth's provocative statement:

"Premature optimization is the root of all evil." (Knuth, 1976).

Front-loading is therefore very important when setting up a Multi-Scalar Model that should give enough freedom to the user to operate important topological changes during the design process, without breaking the model itself. This aspect is very critical for any type of parametric modelling techniques, for both Multi-Scalar Modelling and the other modelling methodologies that preceded it, described in the next section.

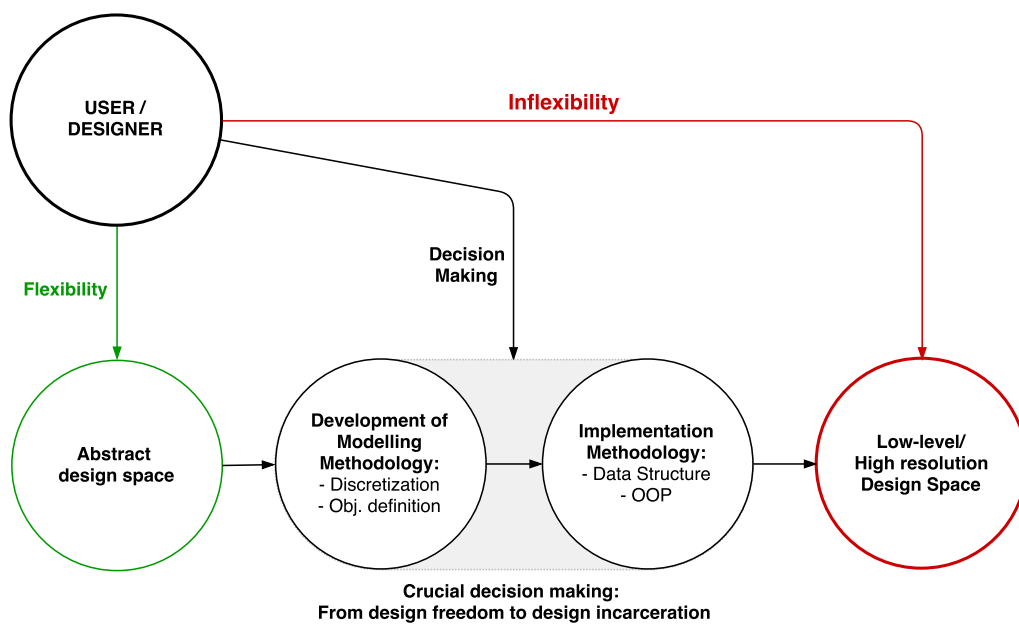
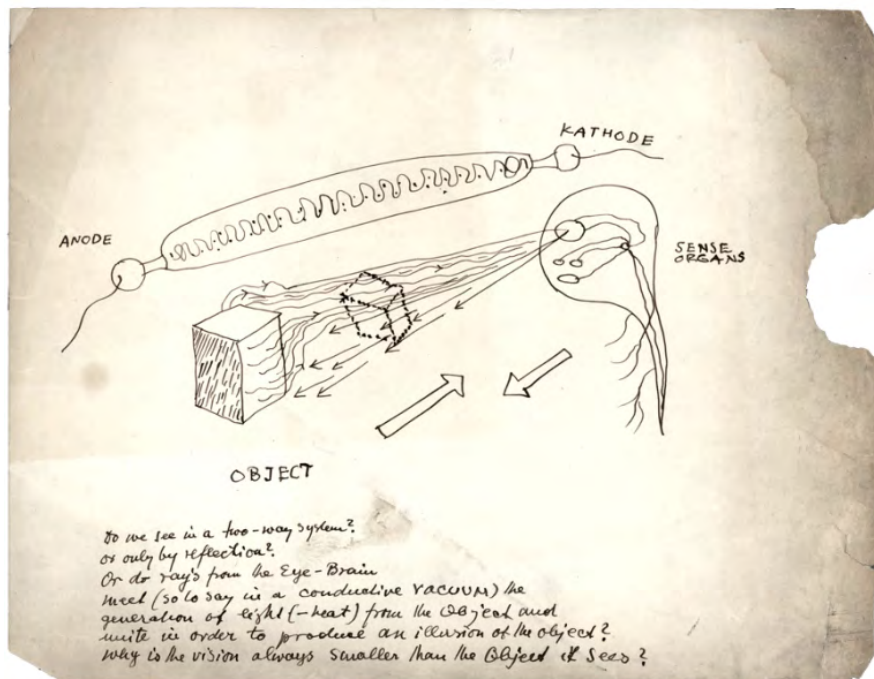
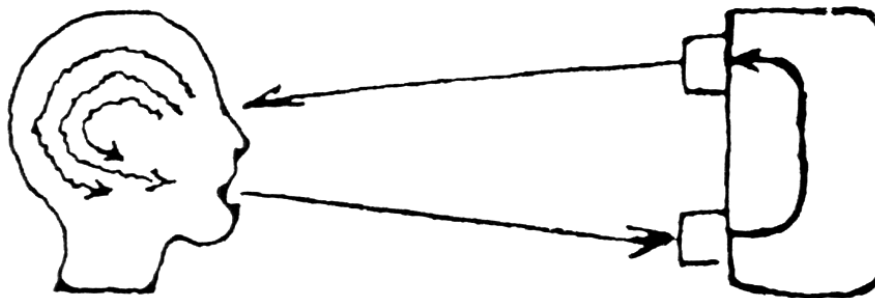


Figure 3.2: (In)flexibilities of the design space.

3.1. MULTI-SCALAR MODELLING IN ARCHITECTURAL DESIGN RESEARCH: ORIGINS AND EXISTING METHODOLOGIES



(a) Frederick Kiesler: Vision Machine studies, 1937–1941. Diagrams. © 2009 Austrian Frederick and Lillian.



(b) The Yershov Diagram, 1963.



(c) Timothy Johnson interacting with Ivan Sutherland's Sketchpad.

Figure 3.3: front-loading and algorithmic thinking: man-machine interaction.

3.1.2 Origins of Multi-Scalar Modelling and preceding modelling concepts

In order to fully understand the meaning of Multi-Scalar Modelling from the architectural spectrum, it is necessary to describe the earlier terminologies that historically preceded this one. Parametric, Computational and Generative Modelling all preceded Multi-Scalar Modelling and compose its backbone at various degrees. Indeed, those different concepts enable its ability to automatize and process algorithmically multiple pieces of geometrical information. David Stasiuk, Director of Applied Research at Proving Ground, wrote recently a white paper that lays out the different existing modelling terminologies used in the design and architectural realm (Stasiuk, 2018). The author differentiates here three types of procedural modeling activities: “*parametric*”, “*computational*” and “*generative*”. He distinguishes them by their or degree of transformative behavior or “*emergence*”. The following paragraph summarizes the author’s definitions and lays out their differences.

- **Parametric Modelling**

Based on Davis final definition of parametric modeling “*a set of equations that express a geometric model as explicit functions of a number of parameters*” (Davis, 2013), Stasiuk redefines it by adding the informational dimension, noticing that “*parametric models have the ability to deliver not just geometries, but systematic collections of information relevant to the delivery of an architectural solution.*” (Stasiuk, 2018, 3). Therefore, he redefines parametric modeling as follows:

“ [...] a set of equations that express information regarding the deployment of an architectural information system, as explicit functions of a number of parameters.”
(Stasiuk, 2018, 3).

- **Computational Modelling**

To introduce computational modelling, Stasiuk first carefully reminds the importance of distinguishing *computation* and *computerization*, referring to Menges and Ahlquist: while the former concept “*increases the amount and specificity of information*”, the latter “*only contains as much information as is initially supplied.*” (Menges and Ahlquist, 2011, 10). Stasiuk arrives then to the following definition:

“computation is the expression of a system of reckoning that informs through operations of transformation, modification, or the ordering of observed physical or symbolic entities.” (Stasiuk, 2018, 4)

In order to illustrate the difference between computational modelling and parametric modelling, Stasiuk takes the case study of a computational design model for designing a series of vaults defined by a network of catenary geometries, highlighting by their transformative and adaptive properties – e.g. the pioneering work of Axel Kilian (Kilian, 2006) and the latest exploratory topology modeling framework developed by Anders Holden Deleuran (Deleuran et al., 2016).

- **Generative Modelling**

Generative Modelling takes a further step by deploying more algorithmic transformations and autonomous behaviors than what computational modelling already does. To exemplify Generative Modelling, Stasiuk takes here as main example a fully generative system developed by Gregory Hornby, Hod Lipson and Jordan B. Pollack (Hornby et al., 2001), highlighting its progressive formation and the evolution of its digital representations through the use of evolutionary algorithms:

“Over hundreds of generations, unanticipated forms are fully generated, and the instruction set for the l-system is continuously redeveloped, intrinsic to the model itself.”
(Stasiuk, 2018, 5)

The complex modelling framework: towards a Multi-Scalar Modelling paradigm for architectural design research and practice

The above enumerated design modelling terminologies described by David Stasiuk hardly tackle the issue of scale or the transfer of information between different levels of resolution across an architectural design project. The four year Complex Modelling research project undertaken at [CITA](#) has been investigating the topic of Multi-Scalar Modelling, aiming at solving this specific problem through different architectural research projects that will be described in the next subsection. This research investigation has been redefining modelling as *“the strategic construction of a network of dedicated sub-models in which simulation models for both design generation and analysis appear as distributed and recurrent events across the design chain.”* ([Ramsgaard Thomsen et al., 2017](#)). Four main points have been raised to describe this specific practice, highlighting (1) the presence of multiple sub-models which *“[...] are composed in an overarching modelling network (workflow)”*, (2) the integration of different kinds of simulation : *“[...] that operate at different levels of fidelity according to locally available information and needs for design accuracy and precision.”*, (3) the tailored validation of each sub-model:

“the dedicated sub-models are locally validated and verified in respect to the particular mode of analysis undertaken.”, and (4) the crafting of different types of handshake techniques for each sub-model transitions: *“[...] information is passed between the models through different kinds of handshakes with different levels of sophistication operating within the models.”* ([Ramsgaard Thomsen et al., 2017](#), 8)

This redefinition of modelling practices also comes with a critique of [BIM](#), where the single model/repository appears here therefore as a fallacy. The concept of Multi-Scalar Modelling attempts to bring answers by inter-linking different and decentralized models, or views, that can start to “talk” to each other, instead of attempting to integrate everything within the same environment:

“The ideal of the unified model: The ambition to integrate all design phases and practices has proven difficult as different practices use different kinds of tools to analyze and represent knowledge. [BIM](#) models are breaking their own modelling frameworks in terms of pure scale becoming bigger, wider and deeper in the sense that they encompass more information from more disciplines (bigger), they include more phases or design (wider) and they expand into new scales of design concern (deeper). This is creating a bottle-neck that is impeding innovation and creativity in architectural design practice. Instead of integrating information into one containing model, we need to build networked models that pass information between discreet part models that are dedicated to particular tasks and can be continuously tuned and changed.” ([Ramsgaard Thomsen, 2016](#), 35)

3.1.3 Multi-Scalar Modelling in architectural design research

As explained in the introduction section, the concept of Multi-Scalar Modelling has been largely referenced and used in different fields such as engineering, mathematics, physics, meteorology and computer science, but very rarely in the realm of architecture. The coupling of computational design and material science have opened-up a new research field within the architectural practice that argues for the emergence of architectural systems at a global scale through the specification of the material at a very refined scale. At [CITA](#), the concept of Multi-Scalar Modelling has been specifically introduced through the realization of different design probes and physical demonstrators. The architectural research projects Composite Territories, The Faraday Pavilion, The Social Weavers ([Nicholas and Tamke, 2013](#)) and Stressed-Skins ([Nicholas et al., 2015](#)) introduce Multi-Scalar Modelling strategies enabling a direct communication between multiple scales, from material specifications at high resolution to the global design environment. In order to communicate between those different frameworks, bridging or “handshaking” techniques ([Winsberg, 2010](#)) have been investigated within architectural design research, allowing the translation of information between different models through coarsening and uncoarsening strategies ([Nicholas, 2016](#)). In the following paragraphs, the various aspects and implication of Multi-Scalar Modelling are discussed more in

depth through the scope of multiple existing architectural design research projects developed at CITA:

- **Stressed Skins / A Bridge Too Far: adaptive levels and resolutions**

Two recent research projects *Stressed Skins* (Nicholas et al., 2016) and *A Bridge Too Far* (Nicholas et al., 2017) developed at CITA focused both on robotic Incremental Sheet Forming (ISF). Each research project employed different meshing methodologies with inherent multi-scalar and multi-resolution properties in order to simulate the appropriate “shifting of plane” that happens during the Incremental Sheet Forming fabrication process. While the former project used a digital design workflow based on triangulated meshing strategies, the latter focused on the implementation of hexagonal meshing through the whole design process. Both methods lead to different qualities and design tectonics. Determining the appropriate levels of resolution is also an important step in the design process which allows the designer to navigate in a flexible manner between the different models in order to verify design criteria and/or structural properties.

Through those described meshing tectonics, the Stressed Skins project manages to transport information from the micro level (material performance and behaviour) to the meso level (plate formation and local structural condition) and the macro level (global structural design). This holistic approach of a design problem by considering in parallel different levels of detail could as well be transposed to an architectural scale in order to communicate information between different scales.

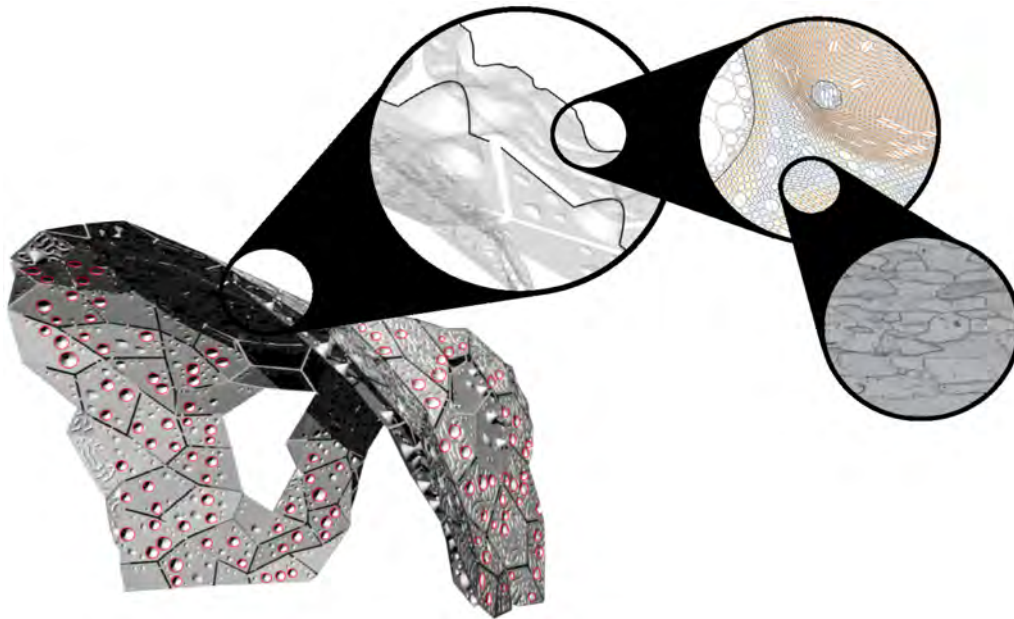


Figure 3.4: The different of resolution investigated and tackled during the modelling process of the Stressed Skins research project.

- **The Rise: simulating material behaviour at different scales**

The Rise is a research-based installation that investigates both Multi-Scalar Modelling and growth simulation using variable sizes of flexible rattan members as physical design probe. The model allows both the simulation of the bending behaviour of each rattan liana, as well as the formation of bifurcating nodes and self-grafting conditions. The design framework integrated feedback also required “to take place on several levels and at different time steps during design development.” (Tamke et al., 2013). The calibrated simulation of the overall bending behaviour through the whole structure allowed the user to interactively run multiple design iterations until fulfillment of design and structural requirements.

Here, The Rise demonstrates the possibility of propagating simulation across multiple scales

throughout the design process, an important aspect to consider in order to interface between different levels of resolution, in the context of Multi-Scalar Modelling for Building Design.

- **Dermoid: a collaborative digital workflow**

The Dermoid is a collaborative research effort between CITA and the Spatial Information Architecture Laboratory (SIAL) in Melbourne (Burry, 2011, 235). The architectural research project consists of an aggregation network of laser-cutted and actively bent plywood components following the topology of a given master surface. The parametric model, as described by Daniel Davis (Davis, 2013, 145) presents relevant Multi-Scalar Modelling properties as the computational designer needs to take into account design parameters at high level (the overall arrangement pattern of the abstract network of elements) as well as material properties at low level (the flexible behaviour of each stripe and the detail generation for each component). Because the passing of information through the different scales presented quite a challenge in the design process and fabrication of the demonstrator, it was an absolute necessity to structure and organize carefully the overall digital workflow (Davis, 2013, 147) by segregating it into different pipelines.

The Dermoid project is an interesting case study to consider when it comes to model and simulate behaviour at different levels of resolutions. The question of organizing and interfacing models arise: setting up a strong digital infrastructure to carry geometrical information in a lossless manner becomes crucial.

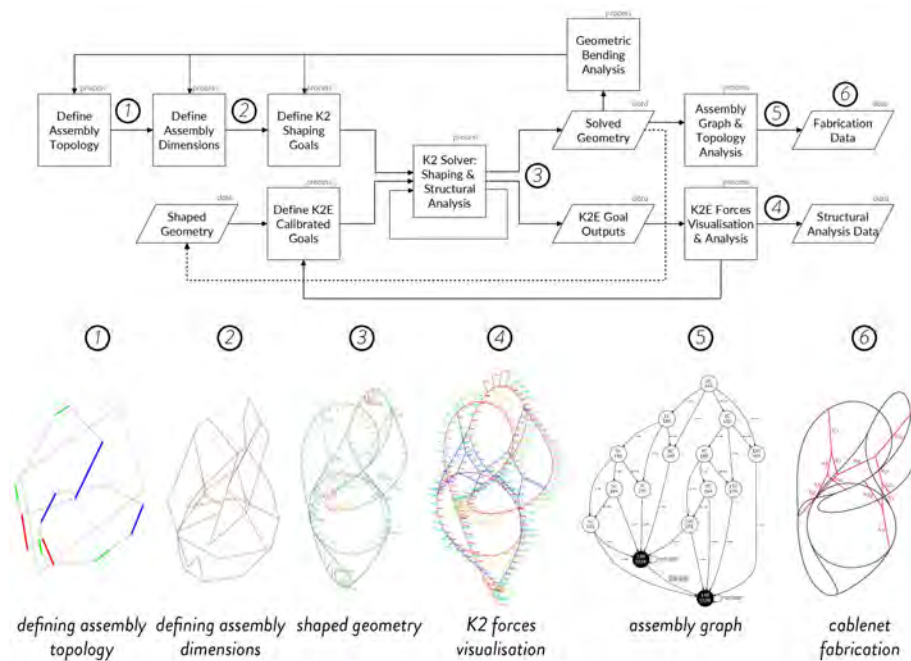
- **Lace Wall / Hybrid Tower: maintaining visual feedback about fabrication information through connected pipelines**

The Lace Wall and Hybrid Tower research projects undertaken at CITA have investigated how hybrid structures that incorporate both tension and compression structural elements can be designed to obtain a highly stiff large-scale structure. The Lace Wall demonstrator is mainly composed from fiberglass beams that are bent, shaped and restrained by an intricate cable network. The computational design modelling pipelines developed both for the Lace Wall and the Hybrid Tower research projects aimed at enabling both interactive user design exploration and the data generation related to fabrication. This particular challenge has been handled by segregating the design process into different sub-models that trigger each other:

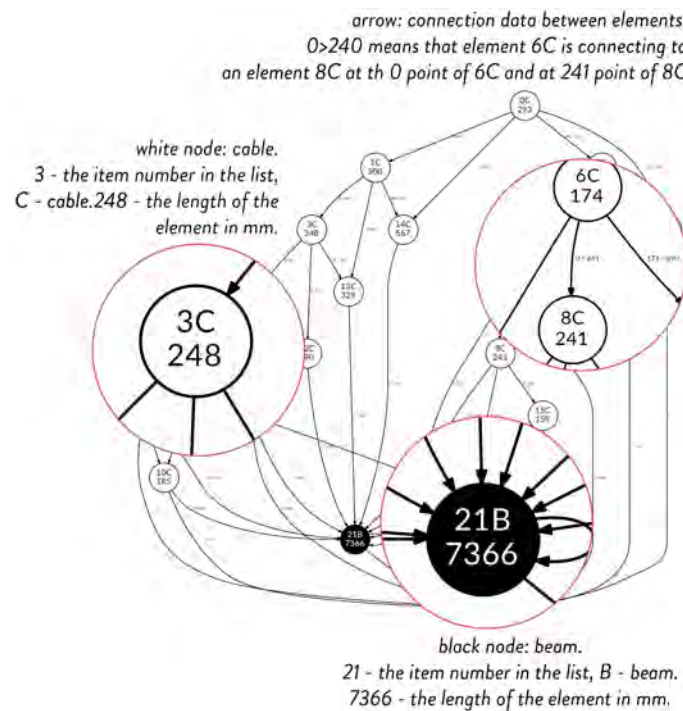
"[...] a computational workflow was developed consisting of a four-step-process: Model Variables contain the definition for the tower design, constant and flexible geometric parameters and material properties, the Generative Model instance produces a sets of geometric possible results based on the parameters set before. The Analytical Model produces performance data of environmental impact, The Design Instance transfers the model data to the required production data. If the design doesn't match the desired requirements, the process is started again." (Ramsgaard Thomsen et al., 2015)

In the case of Lace Wall, 6 different pipelines exchange crucial information for the final cablenet fabrication (Figure 3.5a). First, the assembly topology is defined as the main design driver before adding to it the main assembly dimensions. From this last model, the global geometry can be shaped and the subsequent forces can be visualized through the Kangaroo2 plug-in for Grasshopper3D. Finally, the assembly graph can be extracted for further fabrication purposes (Figure 3.5b). Even though the whole modelling pipeline is contained here within the same environment that is the Grasshopper3D's canvas, it has been segregated into an event-based framework in which different sub-models and simulations are discretely triggered through a continuous exchange of information.

The segregation of the whole design process into different sub-models and interconnected pipelines is an interesting design methodology that could be transposed to Multi-Scalar Modelling for the conception of a larger scale building project, during which multiple models created by different stakeholders need to exchange crucial structural and fabrication data sets.



(a) From user exploration to structural analysis and graph assembly: six different connected computational pipelines enable the final cablenet prototyping and fabrication (Tamke et al., 2017).



(b) Fabrification information model (or assembly graph) incorporating the length index and connectivity of each structural element (Tamke et al., 2017).

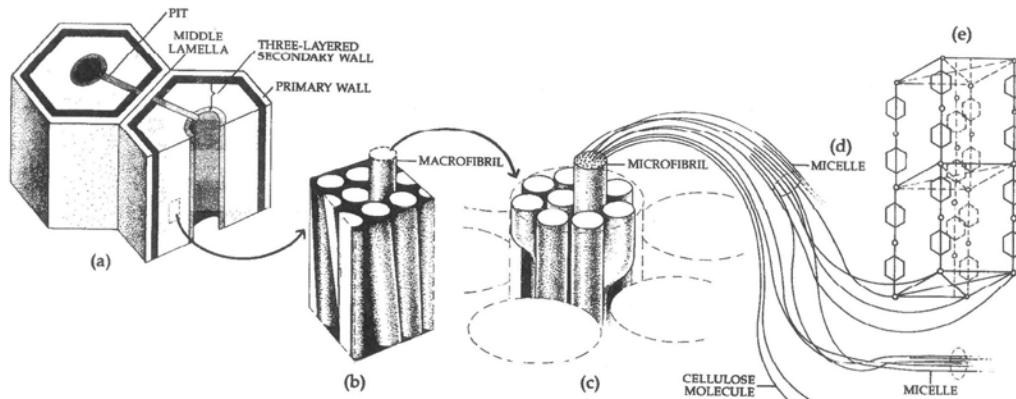
Figure 3.5: Modelling pipelines and overall workflow of the Lace Wall architectural research project.

ICD/ITKE Research Pavilions: an integrative approach

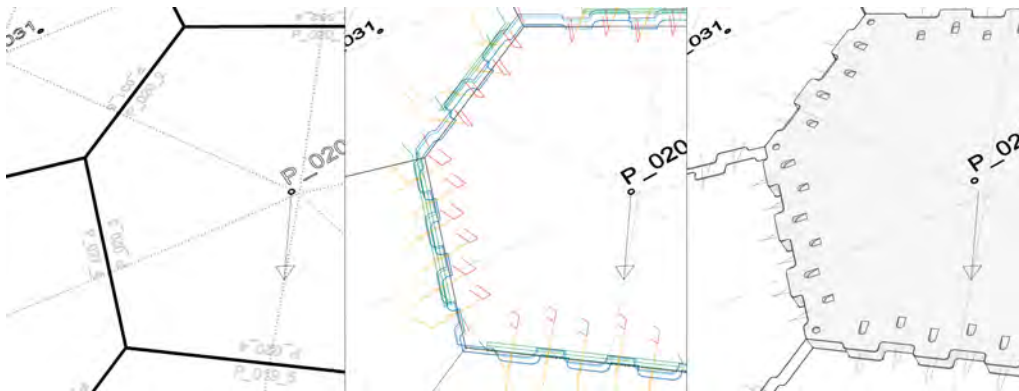
Outside the realm of [CITA](#), other research projects look at interlinking different scales during the design process, although often in a more continuous, integrative manner than the discrete, event-based approach described above. For example, the Landesgartenschau Exhibition Hall, a segmented wooden shell structure developed and fabricated in a collaborative research effort between the Institute for Computational Design and Construction (ICD) and the Institute of Building Structures and Structural Design (ITKE), focused both on the material performance of wood at low level (Figure 3.6a), the fabrication parameters – defined here as the *machinic morphospace* ([Schwinn and Menges, 2015](#), 95) – influencing the differentiated plate morphologies at intermediate level (Figure 3.6b) and the global design requirements at high level. Interfacing between levels and scales was also an important aspect during structural analysis and FE simulation (Figure 3.6c). All those mentioned parameters had to be taken here into consideration during the whole design and optimization process.

The research undertaken at the Institute for Computational Design and Construction generally understands wood as a material that acts across different scales and takes into strong consideration its inherent anisotropic and hygroscopic properties for predicting its mechanical behaviour on a higher level ([Rafsanjani et al., 2012](#)). From the micro-fibril to the formation of the wood-cells (Figure 3.6a) to the industrial processing of the timber, current architectural design research projects are proving that wood can be tailored and self-actuated in order to integrate performance through the whole treatment process until construction, initiating therefore a “*shift toward a large-scale, self-construction methodology*”. ([Wood et al., 2016](#)).

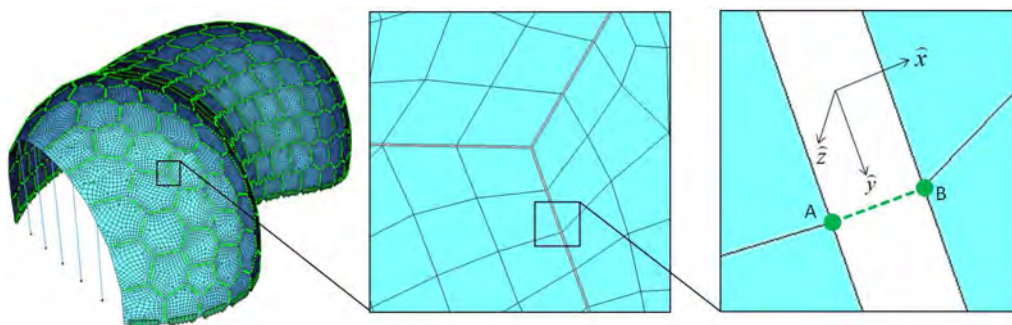
Based on the previously described precedents in academia – the Multi-Scalar Modelling projects from [CITA](#) and the integrative modelling approach used at the Institute for Computational Design (ICD) –, the next section attempts to extract the different methodologies enabling the digital workflows that supported the design and fabrication from each of these projects.



(a) Structure of the cell wall: A: a strand of fibre cells. B: a layer of primary wall and three layers of secondary wall. C: a fragment of middle layer of secondary wall showing macrofibrils (white) cellulose and spaces between the macrofibrils (black). D: a fragment of macrofibril showing microfibrils (white). E: structure of microfibrils. In some parts called micelles there are chain-like molecules of cellulose orderly arranged. F: a fragment of a micelle showing parts of chain-like cellulose molecules. (Esau, 1960)



(b) The tool-paths for the robotic fabrication of a plate are generated from the topology analysis of a surface model (ICD/ITKE/IIGS University of Stuttgart 2014) (Schwinn et al., 2014)



(c) The global FE model of the pavilion. Right: Spring elements act on two nodes, A and B, situated at the two opposite sides of neighboring panels. Axes x, y, and z are the directions of the axial force, the in-plane shear, and the out-of-plane shear, respectively. Axis y is also the axis of bending resistance. (Li and Knippers, 2015)

Figure 3.6: From its cell wall structure to its fabrication processing into differentiated plates that interlock altogether into a large-scale shell structure, wood can be understood here as a Multi-Scalar material where information is transferred between different models, from design to fabrication concerns.

3.1.4 Existing methodologies for Multi-Scalar Modelling in architectural design research

3.1.4.1 Approximating the models: levels, resolutions and uncertainties

Architectural projects are not only context-specific but they embody also unique design objects. Thus, there exist nothing such as a generic Multi-Scalar Model that could solve all problems for all architectural projects. To each project corresponds a custom-made model that is *artificially tuned* in order to respond to multiple (and most often contradictory) objectives, such as material performance, design criteria, structural requirements, site conditions, etc. Therefore, in the context of simulation processes within a Multi-Scalar Modelling framework, each level and resolution needs to be clearly defined and curated for specific purposes (e.g. structural optimization, form-finding, data generation for machine-ready files and deliverables, etc.). It is important here to understand the nuance that differentiates the two terms “Level” and “Resolution”:

- **Level**

A “level” can be described as a degree of hierarchy between two different objects (parent and child) within a object-oriented data structure. The relationship between two levels corresponds to a degree of kinship spanning across two generations of objects.

- **Resolution**

A “resolution” refers to the accuracy (or precision) state of an object. The “*level of resolution*”, that is more commonly known as *Level Of Detail (LOD)* expresses the precision state of an object¹. The representation of the same design object can vary according to its resolution – level of resolution. For example, Multi-Resolution Modelling is often used in computer graphics in order to optimize the amount of data needed to approximate a piece of geometry (Helmut et al., 2007). Similarly, Adaptive Mesh Refinement (AMR) is used in Computational Fluid Dynamics (CFD) to optimize the mesh grid, so that it is denser within areas of greater turbulences than in areas within which less friction happens.

Whereas high level corresponds to a coarse or low resolution state, a high resolution state corresponds to a low level or a non coarse (or refined) representation of the object. The following paragraph highlights the importance of context when it comes to choose at which resolution state the object has to be manipulated.

“Structural model uncertainty”

Eric Winsberg suspected that simulation could be falsified by the important part of approximation during the encoding process:

“Approximate truth is a slippery subject. There is nothing like widespread agreement among philosophers about what a theory of approximate truth should look like, or even, for that matter, if such a theory is even desirable or possible.” (Winsberg, 2010, 130)

The author goes further by questioning the credibility of the tools used in practice which depend on the mathematical models and methods implemented by the developer. These methods target specific practices and might omit parameters that would be essential for other usages:

“The credibility, in other words, of discretization and finite-element methods, solvers, parameterization schemes, multiscale modeling, handshaking algorithms, modeling fictions, and so on cannot be argued for entirely on mathematical or theoretical grounds – they bring their own reliability, established in their own disciplinary traditions.” (Winsberg, 2010, 137).

It is known that climate models are extremely difficult to represent and calibrate, not only because they affect multiple scales simultaneously, but also for the reason that they rely on many approximation and parametrization schemes related to complex and specific context (Figure 3.8).

¹The concept of level of resolution is more commonly known as Level Of Detail (LOD), which adapts the complexity of a 3D object representation accordingly to the viewer’s perspective and the screen’s zoom percentage.

Therefore, in order to replicate natural and realistic phenomena, the scientist needs to go beyond theoretical numerical analysis and take into account a certain degree of uncertainty, which Eric Winsberg calls “*structural model uncertainty*” (Winsberg, 2010). For example, the scientific community recognizes that climate models are subject to an important level of artificial tuning and approximation:

“It is sometimes argued that the parameterization schemes of Regional Climate Models (RMCs) are artificially “tuned” to replicate the climate of a specific area of interest at a given resolution, and that the model projections when applied under different future forcing scenarios are not reliable” (Engelbrecht et al., 2011)

This means that the model cannot be generalized and remains context specific (Figure 3.7), due to important calibrations made by the scientist.

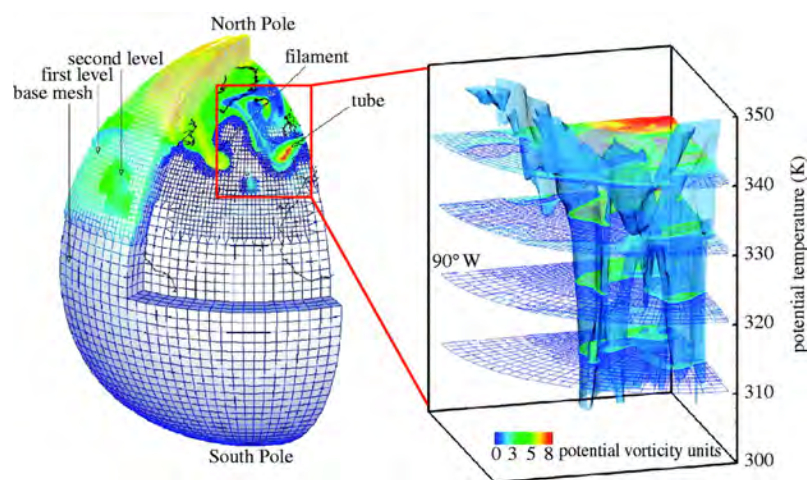
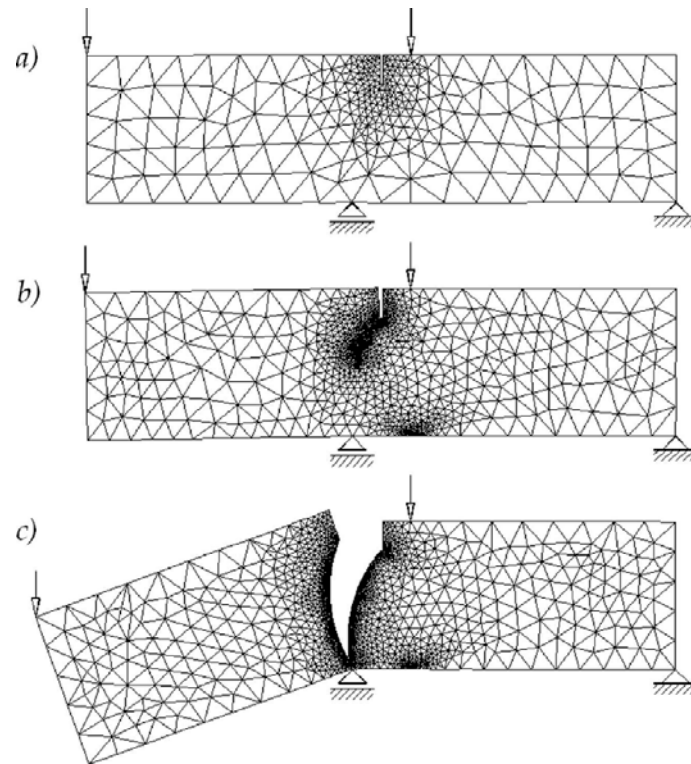
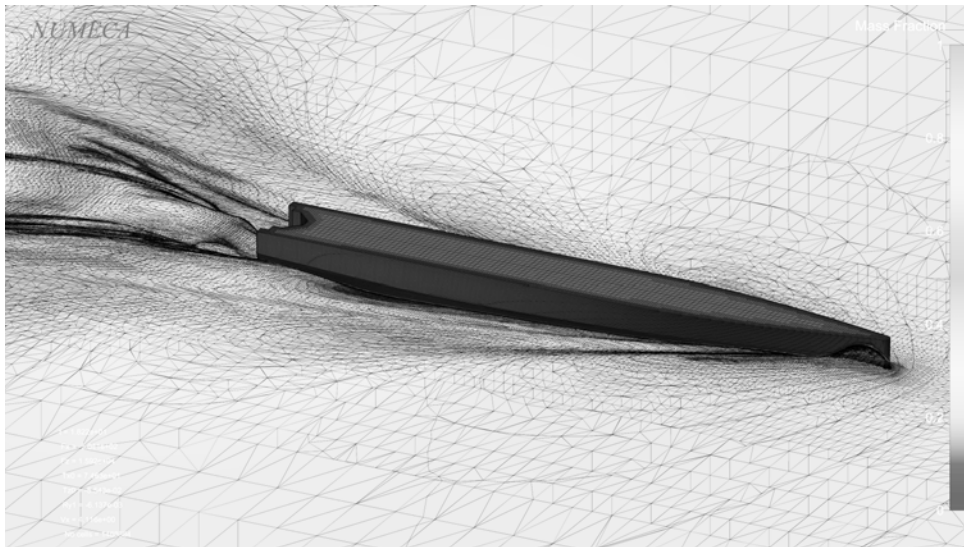


Figure 3.7: Example of an off-line simulation of a tropopause-folding event (June 1996) using structured hierarchical Adaptive Meshing Refinement and Multi-Scalar Modelling on a regular longitude–latitude spherical grid (Nikiforakis, 2003)

After having defined what the concepts of “level” and “resolution” mean in the context of Multi-Scalar Modelling, the strategies and infrastructure enabling the exchange of data between those different scales need to be defined. The following subsection looks at existing practices enabling this exchange, such as the Coarsening/Uncoarsening strategies.



(a) Adaptive Mesh Refinement in the evolutionary problem of crack propagation in a cohesive material. As the image shows, the algorithm uses high resolution grids only at the physical locations and times where they are required. (Secchi and Simoni, 2003)



(b) Visualization of the Adaptive Mesh Refinement algorithm from Numeca's Fine/Marine software using ISIS-CFD RANS solver.

Figure 3.8: Different examples of Adaptive Mesh Refinement (AMR) methods used in the context of crack simulation (Figure 3.8a) and wave behaviour (Figure 3.8b).

3.1.4.2 Coarsening and uncoarsening strategies

After defining what levels and resolutions are, the interfaces between those need to be properly established in order to enable seamless communication between them. As Weinan E stated:

“Alternatively and ultimately, Multi-Scalar Modelling allows the consideration of coupling and interfacing models (or frameworks) at different scales, so we can arrive at an approach that shares the efficiency of the macroscopic models as well as the accuracy of the microscopic models.” (E, 2011)

The Skeleton and Adapter Models defined in the previous chapter are good examples of what such interfaces could look like. However, the methodologies enabling the passing of information between those interfaces still needs to be defined. The present section is introducing the concepts of Coarsening/Uncoarsening through the scope of existing research precedents and case studies from academia. Coarsening and Uncoarsening design strategies (also known as Upscaling and Downscaling techniques), are generally used to:

- **Pass information**

During simulation, information needs to be passed between different scales and resolutions within the same model (see figure 3.9a) by partitioning the graph at high-level (or low resolution) before passing the obtained partition at low-level (or high resolution). Such strategy has been implemented within the research projects *Stressed-Skins* (Figure 3.4) and *The Bridge Too Far* (Figure 3.9b) developed at CITA:

“One example of upstream data propagation is the minimal wall thickness information gained from strains calculation. This process happens at the lowest level of the tree, and to visually inspect the results it is easiest to recursively query each top-level parent to get the lowest value of each of its children.” (Nicholas, 2016, 270)

- **Adapt the representation**

The representation of the model can be adjusted using Adaptive Mesh Refinement (Miller and Stasiuk, 2017). Redundant information can be skipped or deleted without affecting important geometrical features present in the model:

“The subdivision method was extended to support creases (chains of edges which break the curvature continuity) and anchor points (points that stay in place during the process), which are utilised to efficiently and precisely model the deformation. Using this adaptive subdivision strategy, the resolution of a typical mesh used in the first demonstrator can be reduced by up to 30%” (ibid., 268)

- **Maintain flexibility and visual feedback**

During modelling of the Lace Wall project (Figure 3.5), not only the model evolves but also its parallel graphical representation that helps to understand abstractly how the different elements are connected between each other, as well as the overall construction assembly sequences. At the beginning of the design process, the instantaneous visual feedback experienced by the designer allows to reiterate many explorations until obtaining a satisfying design outcome. It is therefore important to keep enough flexibility and remain “unspecific” before the satisfaction of primitive conditions and further implementations of the employed modelling methodologies. In the context of building design, this notion of visual feedback is particularly relevant when it comes to map highly complex geometries of a non-standard structure (Figure 3.16a).

3.1.4.3 Pipelines and workflows

The coarsening and uncoarsening strategies detailed above need digital infrastructures to be deployed. Currently, there are operated through different visual programming software platforms that enable, establish and represent hierarchical links between objects. From Generative Components, DesignScript and Dynamo (Aish, 2013) to Grasshopper3D (plug-in for Rhinoceros3D), visual programming interfaces have become the fundamental infrastructures supporting the setup of “pipelines” and “workflows”:

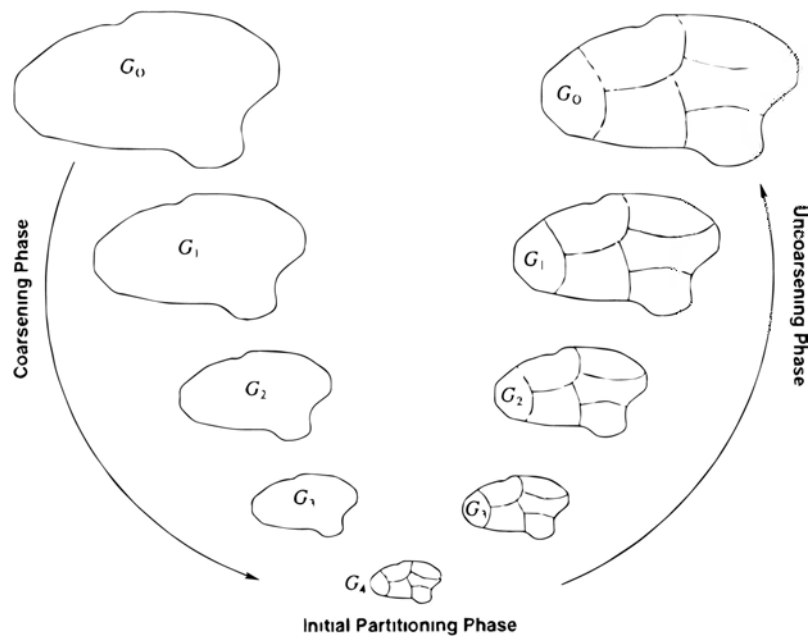
- **Pipeline**

A pipeline is defined as a linear series of processing elements (or algorithms) which quickly filter and transform data, and where the output of one element is the input of the next one. Usually, a pipeline can be represented through the form of a [DAC](#), and its inherent processes are supposed to run synchronously without looping.

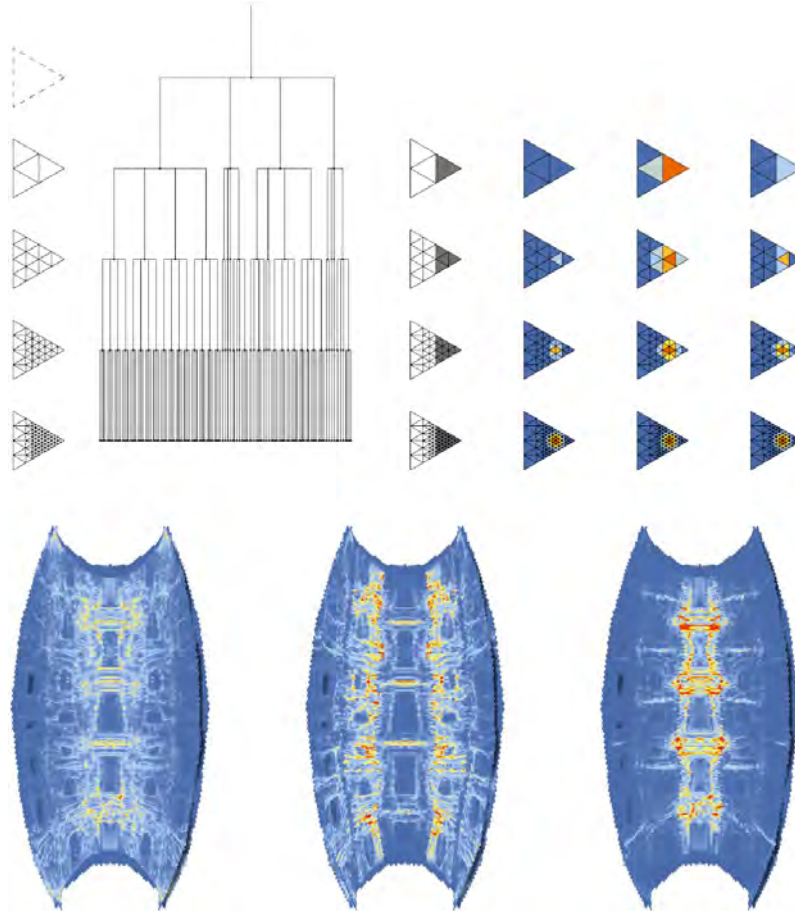
- **Workflow and Connected Pipelines**

Contrary to a pipeline, a workflow is a non-linear set of processes that filter and transform data, involving both machine and human interventions. A workflow often triggers external events (see section 3.1.4.4) and its inherent processes are not supposed to be running synchronously, but are allowed to branch or loop. It can therefore be represented through the form of a Undirected Cyclical Graph, where no first process is clearly defined. Indeed, data may enter the workflow from different sources. Establishing connection between multiple pipelines can form a unique workflow. Anders Holden Deleuran, Ph.D. Fellow at [CITA](#) and Computational Design Specialist at Bjarke Ingels Group (BIG) also refers to “Non-Linear Modelling Pipeline” to describe multiple pipelines that are connected and trigger one another, forming a global computational design workflow ([Deleuran, 2015](#)).

Following the definitions pipelines and workflows in the context of Multi-Scalar Modelling, the next subsection elaborates on Discrete Event Simulation (DES), a conceptual framework focusing on asynchronous workflows and the triggering of events, enabling the passing of information through different scales using upscaling and downscaling techniques.



(a) Diagram representing the coarsening and uncoarsening phases, where a graph is partitioned at high-level before passing the obtained and informed data at low-level.



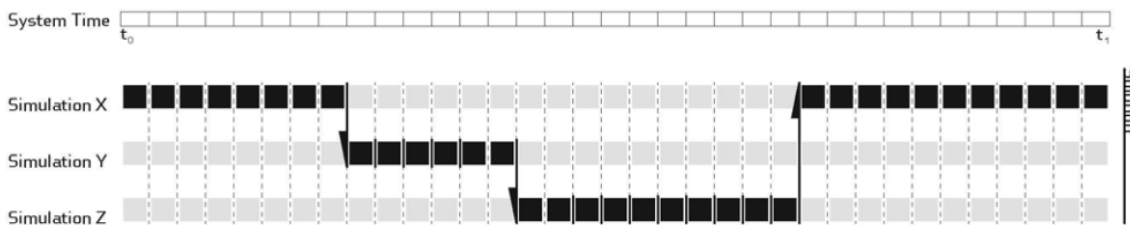
(b) Bi-directional data propagation between low and high resolution during the modelling process of the research project *The Bridge Too Far*. (Nicholas, 2016)

Figure 3.9: Bi-directional workflows, coarsening and uncoarsening strategies.

3.1.4.4 Discrete Event Simulation (DES): linking levels and resolutions through Multi-Scalar Simulation and optimizations

It has been seen earlier that the consideration of keeping full continuity in the design process of a large-scale and complex architectural project is more of an illusion than a realistic scenario. Therefore, one needs to *Separate the Concerns* and segregate the whole pipeline. This is where the Discrete Event Simulation (DES) concept comes into play. A DES can be represented by a step function, and is defined as a non-linear process in which different events can happen in parallel, one event triggering another in an asynchronous manner (Matloff, 2008). This differs a lot from the more traditional so-called Continuous Simulation which could be represented by a continuous function in which optimization tasks are running one after the other. One task needs the previous one to finish in order to start, and all processes follow a strict chronological order. In order to illustrate a DES, Matloff takes the simple example of a warehouse: “Purchase orders come in and are filled, reducing inventory, but inventory is replenished from time to time. [...] The events here—decreases and increases in the inventory—are discrete variables, not continuous ones.” Many other examples could be taken, such as: air traffic at an airport, banking systems, car traffic at a parking lot, queuing time at an elevator, and so on.

Continuous simulation



Discrete Event Simulation

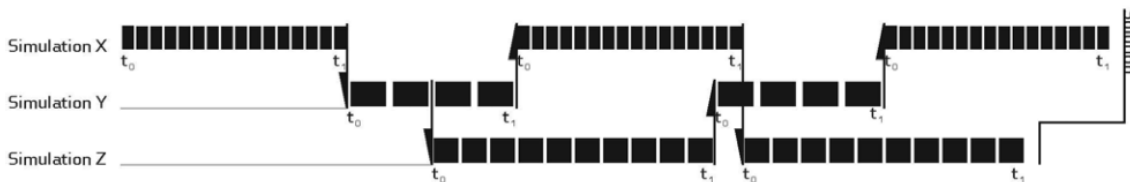


Figure 3.10: The different time frameworks of the continuous simulation (top) and the Discrete Event Simulation (bottom). (Tamke et al., 2014)

The Separation of Concerns (Dijkstra, 1982) and the DES paradigm have been investigated through the Transmissive Assemblies project, a demonstrator and ceiling installation developed and fabricated at CITA made from an assembly of different hexagonal sandwich panels. The prototype asks “*how a modern composite sandwich might be designed to modulate the transmission of light in a controlled manner through strategic material variation.*” (Tamke et al., 2014, 66). Before fabrication, the digital design model is segregated between five different modelling and simulation frameworks (a Boid type simulation, a generative height map, a boundaries extractor, a structural analysis and an optimization via genetic algorithm) which separately focus on achieving particular design or structural goals. These different frameworks communicate and pass information between each other until all parameters are satisfied and validated by the designer:

“The digital design process links together a number of different Generative models, each related to one of the constituent materials within the sandwich assembly. Over a time-based process, these models trigger one another, and transmit information between themselves, until a stable condition is achieved that satisfies a design intention regarding light conditions as well as structural stability.” (Tamke et al., 2014, 66)

Transmissive Assemblies is tackling here the current modeling complexities that involve a multitude of pipelines and workflows that need to “talk” and connect to each other. Today, both

Continuous Simulation and the DES can be used to navigate across different pipeline that handle different modeling processes for different solutions and resolutions (Tamke et al., 2014). However, Tamke argues that DES are more flexible than continuous simulation, because of their more organized data-structure:

"The modularity of the DES formalism suggests a more organized code structure, which makes models more flexible, transferable, and reusable." (Tamke et al., 2014, 65)

Furthermore, DES enables the coupling of multiple sub-models, allowing "interdisciplinary collaboration, scalability of systems and the reduction of management of simulation." (Tamke et al., 2014, 65). Because of its inherent scalability and flexibility properties DES presents interesting aspects that could be used in the context of Multi-Scalar Modelling, within which scales and resolutions also need to inform each other throughout the whole design process.

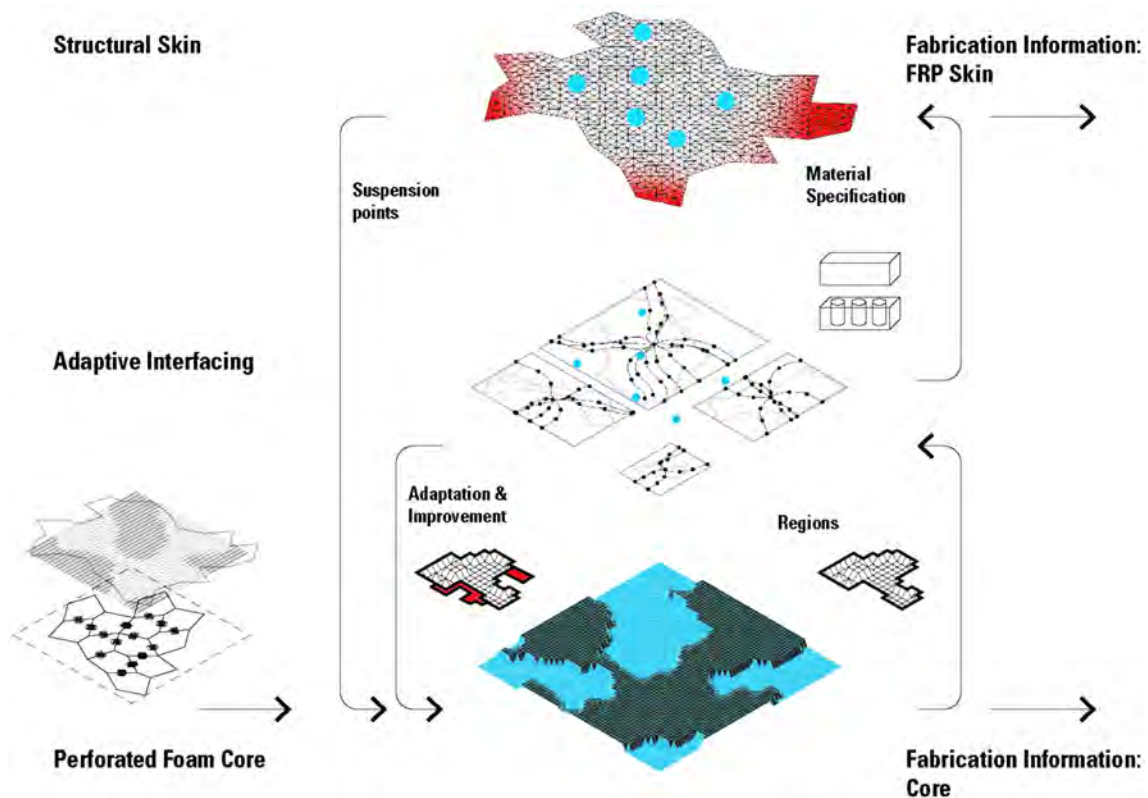


Figure 3.11: The different system times of the continuous simulation and the Discrete Event Simulation. (Tamke et al., 2014)

Discrete Event Simulation (DES) in the context of building design

In some aspect, the Metagraph produced by Woods Bagot (Figure 2.29) or the cross-practice collaboration diagram proposed by Dimitrie Stefanescu (Figure 2.15c) can be seen as sorts of Discrete Event Simulation frameworks in which data is shared between multiple, decentralized models that are triggered by the inputs of other models. Conceptually, the Building Information Generation strategy developed by can also be interpreted as a sort of DES workflow where the digital chain is interrupted when the data is transferred and stored within the Rhino models. The user needs to manually bake the geometry, apply onto it a grasshopper script in order to produce the next data generation. A DES model enhances such experience by triggering data transfer through more automated processes.

Discrete Event Simulation (DES) for Product Lifecycle Management (PLM)

DES could also enhance the current Product Lifecycle Management (PLM) processes of a building. The PLM concept generally corresponds to the management of the entire lifecycle of a product from conception, through engineering design and manufacture, to service and disposal of manufactured products. In the building industry, researchers and professionals tend to merge the concepts of **BIM** and PLM by proposing different alternatives generally labeled as “**BIM 4D**” (**BIM** + the time/schedule dimension), “**BIM 5D**” (**BIM** 4D + the cost dimension), “**BIM 6D**” (**BIM** 5D + all other project-related lifecycle information) etc. However, at the end of the day, all those acronyms remain primarily conceptual, meaning that there exists no official technical implementation to tackle PLM related challenges. Recent research has also been looking at Discrete Event Simulation in order to solve the latter. Indeed, the whole construction and PLM processes of a building can be seen as discrete events that are dependent from each other, in the same way as one looks at air traffic at an airport or banking systems, for example. Recent research also looked specifically at the DES for the modeling of a mobile crane movement under dynamically changing site layout ([ElNimr et al., 2016](#)). Also, during the conception and realization of the Brock Commons Tall Wood House building project (Figure 3.12), the third party company CadMakers developed a complete 3D model which not only contained all geometrical data but also embedded Discrete Event Simulations that could be run while the building itself was erected on the construction site ([Acton Ostry Architects Inc., 2016](#)). Those different research projects and industry-based examples show that traditional modelling practices can be rethought through the scope of DES.



Figure 3.12: ([Acton Ostry Architects Inc., 2016](#))

3.1.4.5 Conclusion

In this section, different projects from academia focusing on Multi-Scalar Modelling strategies have been described and analyzed. In their own specific ways, these different projects envision through Multi-Scalar Modelling future design thinking that could be transposed into practice, by investigating speculative digital infrastructures capable of carrying complex information through multiple scales. Each project described in this section has been built and relies on four design thinking strategies, already described above summarized here:

- **Approximating the models by defining the required levels and resolutions.**
- **Defining the coarsening and uncoarsening strategies to navigate across the previously defined levels and resolutions.**
- **Setting up connected pipelines, design workflows and handshake techniques to deploy coarsening and uncoarsening strategies.**
- **Deploying simulation frameworks and optimization strategies across the previously defined pipelines and design workflows.**

The present thesis argues that transferring those design thinking strategies – already used in academia – to the building scale would strongly benefit the [AEC](#) realm. Although the academic realm might be familiar with these design thinking strategies and methodologies, the construction industry is facing other sorts of problems, linked more specifically to interoperability and data management concerns – as seen in Chapter 2. Therefore, those design thinking methodologies would still need to be revisited and enhanced in order to take into account the specific challenges faced by the building realities. The next subsection highlights the current discrepancy that exist between the different design methodologies employed in academia and the main challenges currently faced by the [AEC](#) sector, before proposing an hybridization between the speculative design thinking methodologies coming from academia and the current strategies employed in practice to tackle design problems on an everyday basis.

3.2 Existing Discrepancy between Academia and Practice



Figure 3.13: This figure highlights the difference between the material specific and seamless design integration of research pavilions and demonstrators in academia (left), and the heterogeneous design landscape of the building industry (right): element clashes, intricate material interfaces, large diversity of employed materials, etc.

The different research projects, demonstrators and their inherent modelling methodologies described in the previous section represent quite well current research in academia, looking at how we can integrate material behaviour, structural analysis and spatial features through all scales and phases of a design and construction process. These research projects have proven to successfully enable such integration, which however answers very specific problems inherent to a particular material behaviour or structural system. Indeed, the developed design workflows and interoperability bridges are custom-made, often conceptualized and designed very quickly – to meet short deadlines – and therefore rarely re-usable for other research investigations that deal with different sets of materials and processes.

The demonstrator scale also largely differs from the building scale, which deals with a much more diverse range of elements, material interfaces and detailing. It is therefore difficult to transpose directly the same design methodologies investigated within the academic realm to digital design workflows for architecture, which often fail at containing the whole digital chain across tightly interconnected modelling pipelines. Those need to be instead clearly segregated across different submodels (see section 2.3). Indeed, the way that data is generated and classified varies drastically between academia and practice. In the former case, data generation and classification are usually highly intertwined and integrated within each other: the dynamic data tree or DAG hardly allows the user to operate any sort of changes (Davis, 2013) onto the data classification itself without affecting the whole design process. In the latter case, in practice and industry, the data generation and its classification are more independent from each other, and are staged in a discrete manner through multiple files, modelling environments (Van Der Heijden et al., 2015), multiple trades and layer tables. Here, the user is free to operate classification changes whenever it is necessary during the design process. Classification features (such as names and attributes) can be introduced, developed and refined before, during and after the data generation. Although these modelling practices used in industry have proven to be robust enough to carry complex architectural projects through all the post-tender phases, it has been explained in the previous chapter that they also come with their sets of limitations as well (residual models, duplicate files, poor management interfaces, etc.). In order to overcome the latter, custom tools and workflows for enabling better interoperability and data

management need to be developed (see section 2.4.2), as well as neutral formats to enhance interoperability between the different existing software platforms.

The advent of the computer within the architectural design domain made architects and designers rethink how material and design can be perceived (Menges and Ahlquist, 2011). Indeed, it is acknowledged that architecture entertains profound links to technology and recognizes the creative and transformative power of new technology platforms (Ramsgaard Thomsen, 2016). In the past couple of decades, a growing number of universities and institutes have been establishing state of the art teaching programs, robotic facilities and research clusters (Knippers et al., 2018). Through these different platforms, the academic community has developed strong speculative design thinking and envisioned how future modelling techniques could be deployed into practice. However, these strategies have been tested internally within academia and not deployed yet across the AEC industry. The difficulty of deploying knowledge produced in academia into practice has been highlighted and discussed by both Nathan Miller and David Stasiuk during the second episode of the Proving Ground podcast “Prove It”. First, Nathan Miller introduced the divergent interests between the two realms:

“One of the larger problem area is the disconnect that exists between academia and professional practice. It almost seems that there exists a bit of a bubble created around digital fabrication in academia, to the point where these experiments are esoteric and driven by just the interest of the researcher. Only very rarely we start to see a lot of these things make their way into practice, which is hindered by a risk aversion that comes with the idea that designers are able to construct and develop schemes made of components embedded with complex geometries. In this case, the interface with the manufacturer or the fabricator still needs to be solved, as contractual obligations usually prevent it.” (Miller et al., 2019a)

David Stasiuk continued the discussion by warning the academia to take more into account industry constraints in order to better bridge the knowledge produced in academia into practice:

“There are different constraints that exist in practice and in academia. The constraints of really building a building that people will occupy need to take into consideration building code, client needs, and other real life constraints. Academia is typically unburdened by and uninterested in those constraints. The interest in research is on almost anything but those constraints, which is probably in the long or mid run problematic to find a way to bring the knowledge from academia into practice.” (Miller et al., 2019a).

Seizing the opportunity of collaborating with two industrial partners – Design-to-Production and BuroHappold – that are very well aware of the current modelling and data management challenges in the AEC industry, the present thesis attempts to merge these two domains by translating the Multi-Scalar Modelling techniques described in the previous section to the industry sector and confront them with construction realities and industrial design problems and challenges.

Both ends against the middle

This current discrepancy between academia and practice is very important to grasp and acknowledge in order to reformulate what Multi-Scalar Modelling means in the context of building design. Indeed, Multi-Scalar Modelling should not only take into account the concepts currently applied in academia, such as the event-based approach that was applied to the Transmissive Assemblies research project described above in this section, but it should also focus in closing the poor interoperability gaps that still persist within the AEC industry by proposing a robust digital infrastructure that can carry fabrication data and in defining new methods to scale up complexity and deal with late changes in the design process. In other words, both the bottom-up approach often chosen in academia to tackle design problems and the top-down strategies employed by the industry to solve daily challenges should meet in order to propose a coherent digital infrastructure to collaborate. Such approach, defined as “middle-out” or “both ends against the middle” has been already been proposed to deal with participatory design problems (Twidale and Floyd, 2008). The following section therefore aims at merging both ends (the approach in academia and the one in industry) by redefining, re-articulating and expanding the current employed Multi-Scalar Modelling methodologies in academia described above with industry related concerns. These redefined methodologies will take into account the context of building design, its challenges and the different findings from Chapter 2.

3.3 Redefining and Expanding the Existing Methodologies of Multi-Scalar Modelling for the Context of Building Design

A large-scale and complex architectural project is always subject to be undertaken by multiple trades and disciplines simultaneously, dealing with similar and/or overlapping data that crosses each other. Similar parts of the building could be handled by different companies, depending particularly of the particular typologies of those parts. This complex web of information between the different trades involved in the project produces a non-linear generation process of data, which can grow faster within one area (one specific building's component type) than another. Indeed, if two data types are managed by two different companies, it is most probably that the digital production chain will evolve differently. For example, there can be some delay in the manufacturing process for one specific object, while the fabrication of another might happen much quicker. Figure 3.14 represents this dilemma and highlights the fact that the design space (in which the computational designer can act) is much more flexible at early stages that late stages, when the information is more massive, disparate and decentralized.

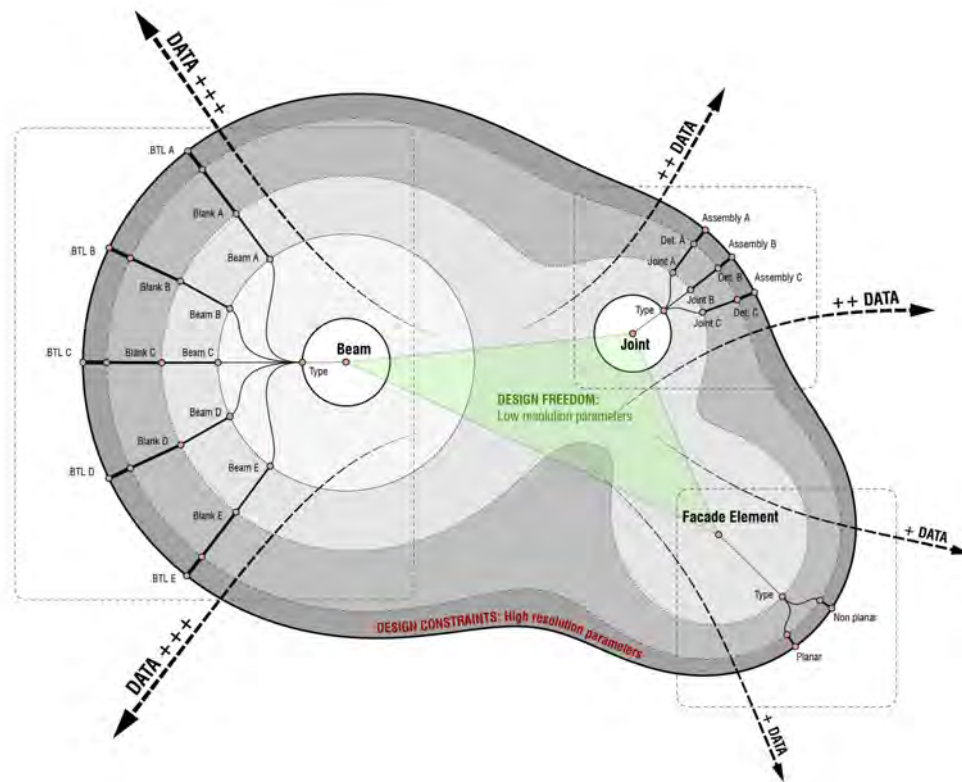


Figure 3.14: Information overload at late stages in the design process: representational diagram of the non-linear evolution of data generation during the conception and realization of a large-scale and complex architectural project

Integrated vs. segregated strategies

From an architectural research perspective, Multi-Scalar Modelling approaches aim to describe *integrative* design solutions containing a very precise hierarchy of objects that can link data and physical matter across scales and resolutions. This linkage can be operated through “*handshaking*” techniques (Winsberg, 2010), which enable the passing of information through different simulations. Integrative models are in general more performative, but less adaptive to changes than *non-integrative* models.

From an industrial perspective, any complex architectural project might throw errors and exceptions within its own design system. A certain amount of clashes might start appearing between multiple types of elements during the conception phase. The meeting of two different types generated within two distinct parametric systems coming from two different trades might sometimes lead to an exceptional and unforeseen contradiction that often need to be rectified manually, since it would take usually more time to develop a third parametric system meant to resolve a unique exception. The latter design process can be described as *non-integrative*. Unlike the *integrative* approach, a *non-integrative* design workflow allows for mistakes and let the user solve it outside the current parametric model (manually, through the help of additional computational processes or by hybridizing both methods). *Non-integrative* design workflows could be analyzed through the use of graphs, layers and attributes that map hierarchical information on the respective objects they refer to. Here, the *Post-rationalization*, *Pre-rationalization* and *Embedded rationality* (Aish et al., 2012) design methods are important to understand whether a project is *Integrative* or not. In any case, these design methods embed – after, before or during the conception process – geometrical constraints in order to ease further fabrication processes.

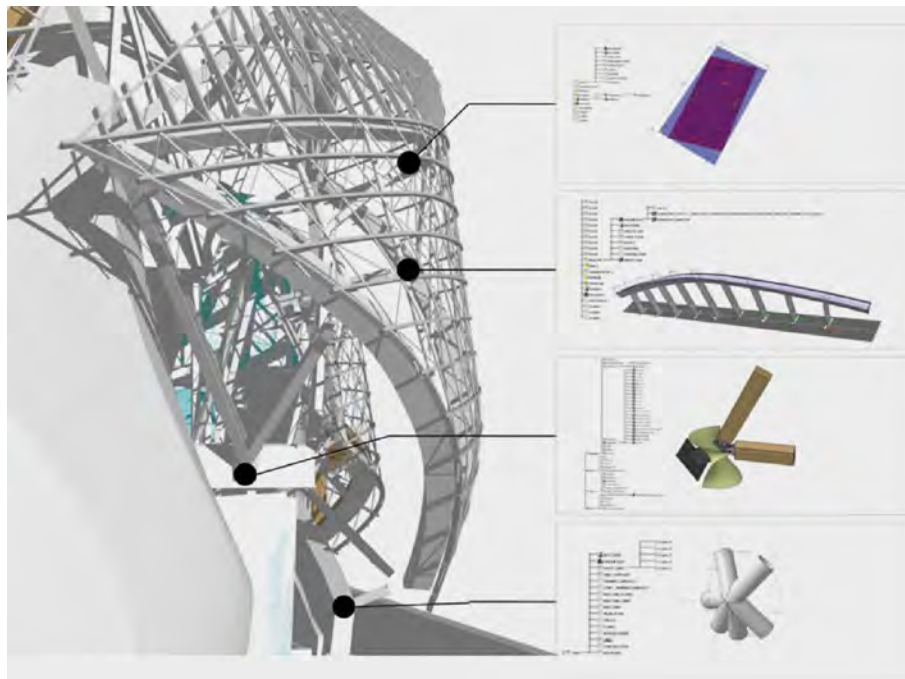


Figure 3.15: A digital interface that encapsulates the different objects composing the *Fondation Louis Vuitton*.

One of the biggest challenges of Multi-Scalar Modelling in the context of building design is to consider the conception of very complex and large-scale architectural projects as part of a *continuous environment* where different scales can relate to each other, even though the design system of such projects is very often *non-integrative*. Within the recently built project of the *Louis Vuitton Foundation* conceived by Frank Gehry, different types of elements meet each other through multiple levels of resolution and provoke sometimes non-predictable material interfaces. Large glass panels are supported by glue-laminated timber beams that are themselves supported by steel profiles, which are connected to the building's steel reinforced concrete core. By considering those geometrical and material complexities that emerge from the chosen design system, this architectural project could be labeled as *non-integrative* and *post-rationalized*. However, the latter has been conceived through *Implicit Material Optimizations* (Nolte and Witt, 2014) which have been integrated within the design model after the main design decisions have been made. A negotiation between the described material systems and the design intents could be solved by encapsulating the different objects within a unique interface (Figure 3.15). In this case, the initial design model/prototype has been “distributed across many machines, courtesy of the cloud model server”

([Nolte and Witt, 2014](#)). Since the design components, designed both by design consultants and engineers, were stored on a centralized database, it was possible to access, modify and update them as needed, by taking into account structural and spatial requirements.

From those previous realizations, the Multi-Scalar Modelling methodologies need to be revisited and extended in order to apply them into practice:

- **Segregating the Levels**

Although well distinguished in academia, levels and resolution need to be clearly segregated within the workflow. This does not mean that they can't communicate, but the user needs to be able to isolate them, temporarily, in order to run optimization onto them when required.

- **Encapsulating the Workflows: Hierarchizing the Code**

The workflow needs to be clearly organized across multiple infrastructural levels of code in order to segregate the process into distinct pipelines.

- **Differentiating Modelling and Simulation**

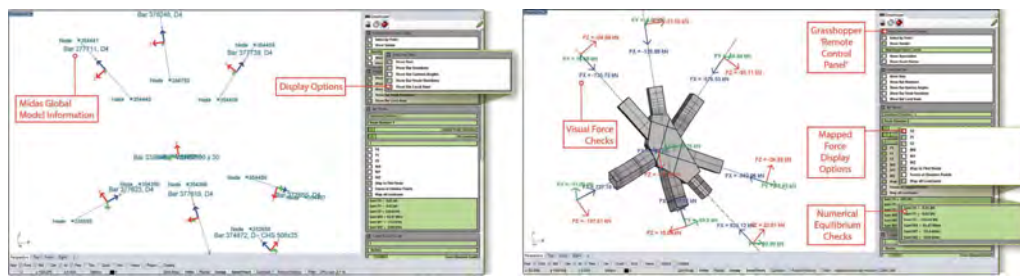
Setting up a modelling workflow across scale is already a challenge. The distinction between Multi-Scalar "Encoding" (as the encoding of objects across scales) and Multi-Scalar Simulation (as the addition of specific simulation frameworks for each levels) is therefore necessary to break down the complexity. Simulation features and frameworks can be added on top of the Multi-Scalar Model once the different models, pipelines and their inherent connections are clearly defined.

- **Curating the Workflows and Enabling Feedback: Management Interfaces and Handshake Techniques**

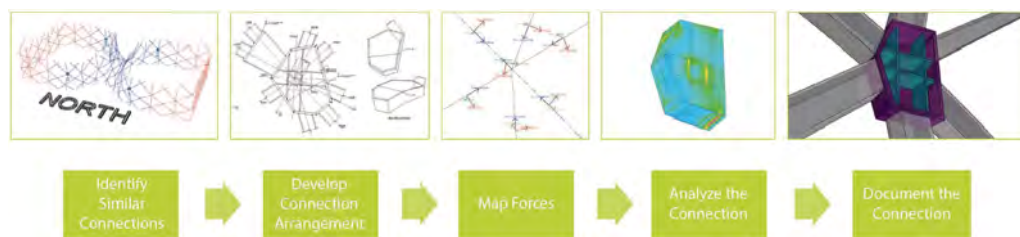
Once all the points above have been tackled, it is possible to enable feedback strategies between the different levels of resolution. For each architectural projects, depending on the clients needs and the required software, the overall workflow needs to be curated by developing custom nodes, data management and interoperability interfaces that support the digital infrastructure of unique architectural design projects.

The next subsections will elaborate each of the above points by introducing their related methodologies enabling a Multi-Scalar Modelling framework for building design.

3.3.1 Segregating the levels



(a) Visualisation and numerical check of mapped forces at each complex nodes of the architectural project City of Dreams. (Piermarini et al., 2016)



(b) Design process of a complex node: from the abstract network to full detailed resolution. (Piermarini et al., 2016)

Figure 3.16: Modelling methodologies and workflows employed by the structural design team at BuroHappold during the conception and realization of the City of Dreams architectural project designed by Zaha Hadid Architects. (Piermarini et al., 2016)

In the context of building design, each level of a Multi-Scalar Model would present a unique set of constraints to solve. A global optimization task would not be efficient to find solutions at all levels at the same time. It would be more efficient for each optimization task, at different defined levels, to run independently from each other (without however disabling totally the communication that should still happen between those) in order to find a final satisfying output that ideally takes in account all predefined constraints. This workflow scheme has been used to realize the different academic demonstrators described in section 3.1.3 and could be summarized as follow, from the lowest level to the highest one:

- **Level 1: Topological optimization**
At low level, a topological optimization takes into account material behaviour and its structural integrity.
- **Level 2: Fabricability optimization**
At low level, fabricability optimization takes into account different fabrication processes (e.g. the machine morphospace and its manufacturing precision).
- **Level 3: Local design optimization**
At intermediate level, the local design optimization takes into account a building component's design space which can be modelled through different levels of resolution. Its local stability can be tested against structural analysis methods.
- **Level 4: Global design optimization**
At high level, a global design optimization analyses the design space enabled by the Multi-Scalar Model.
- **Level 5: Global structural analysis**
At high level, a global structural analysis verifies the structural integrity of the resulted global design.

The distribution of different simulation and optimization tasks at particular isolated levels has also been a strategy adopted by BuroHappold for the calculation of forces and the design of complex nodes of the Morpheus Hotel realized in Macau, China. Figure 3.16b presents the step-by-step design and simulation process to break down the large amount of geometrical complexities spread across the overall building structure to a particular node. The following paragraph elaborates on the notion of level hierarchy and goes further by describing existing design methodologies employed in practice (such as the “Node-to-Code” approach) allowing the encapsulation of workflows and the hierarchization of code.

3.3.2 Encapsulating the workflows: the “Node-to-Code” approach

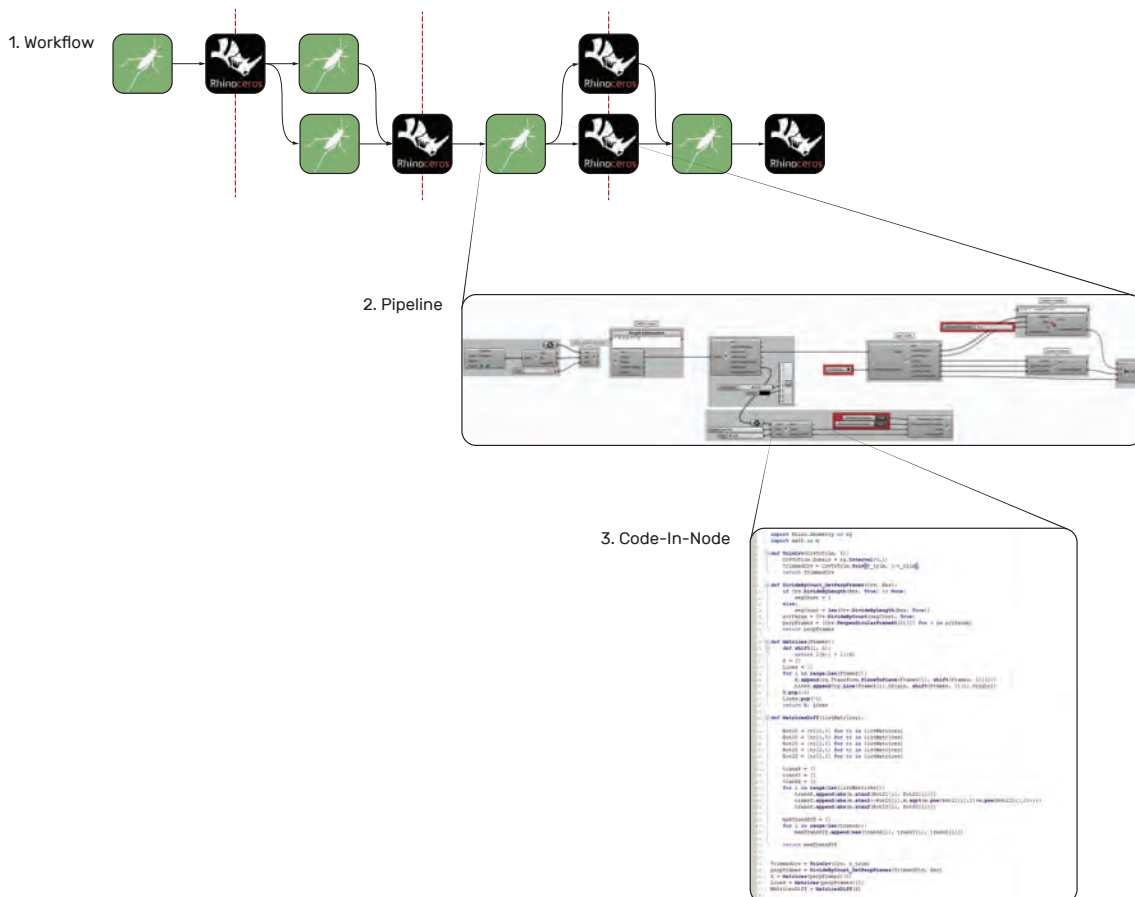


Figure 3.17: This diagram conceptually highlights current design strategies in which code, nodes, pipelines and workflows are interlocked within each other, contributing here to the *Separation Of Concerns*.

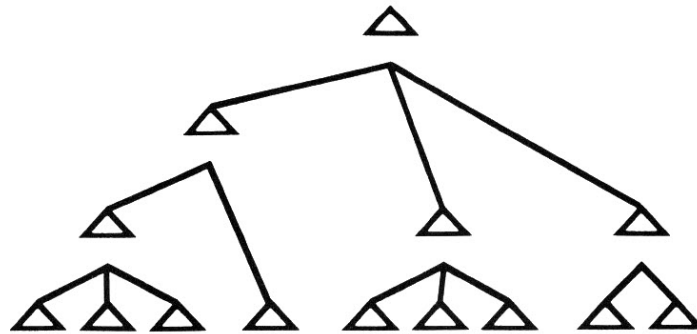
As already mentioned in Chapter 2, the overall workflow tend to grow very rapidly and is confronted to a scaling problems in the context of building design. Indeed, during the design process of an architectural design project, as the Cyclomatic Complexity (CC) of the DAG is constantly growing, the user will experience some difficulties in adding/removing/modifying parametric features at different scales during modelling and within the same design workflow. To reduce the CC, different solutions have been developed and proposed by many software packages, all based on McCabe's idea of modularizing subfunctions and restructuring (or refactoring) code methods (Davis, 2013). For example, the Grasshopper environment has an integrated cluster option allowing the user to wrap multiple components within a same global node. A part of the overall design workflow is thus transformed into a subfunction, or isolated pipeline, where the input and output are clearly defined and assumed to be always consistent. This method drastically decrease the CC of the main DAG and helps the user to “forget” about non-crucial parts of the main DAG. Similarly, scalable tools

for design computation have been introduced within the Dynamo design framework (Aish, 2013). Here, the Node-to-Code function allows the user to replace many graph nodes by a single one, containing the lines of code corresponding to the algorithm operated by the previous nodes. The author justifies the usage of such procedure by the increasing visual intricacy of the graph during the design process:

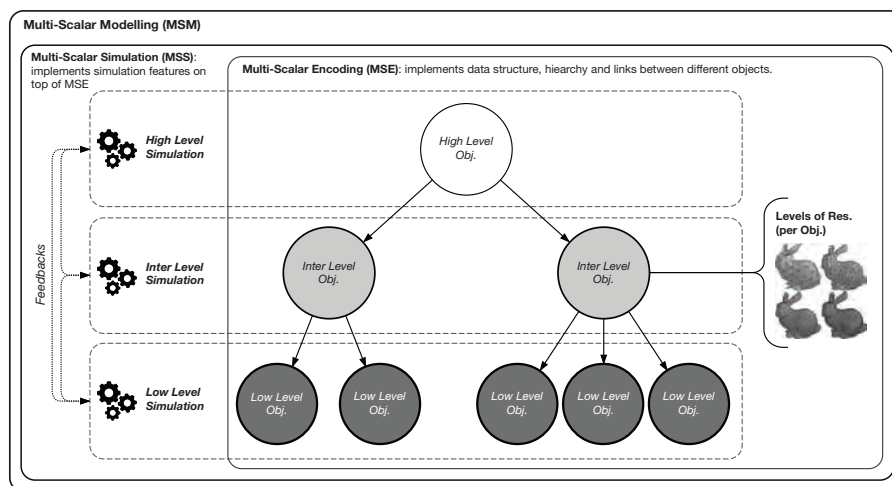
“Indeed the visual complexity of the graph may become overwhelming and counter productive. It is exactly at this point where the application should encourage the novice programmer to make the transition from graph node diagramming to scripting: literally from node to code.” (Aish, 2013, 88)

Organizing the DAG or overall workflow by clustering and/or grouping pieces of algorithms together is seen here as a good strategy to reduce complexity and increase flexibility of the design process. The next section emphasizes the need of distinguishing modelling from simulation as the two can merge, operate and act separately at different times in the design process.

3.3.3 Differentiating “Multi-Scalar Encoding” from “Multi-Scalar Simulation”



(a) Alexander’s tree-like design structure consisting of sub-sets, or sub-models.



(b) Multi-Scalar Modelling: from Multi-Scalar Encoding to Multi-Scalar Simulation

Figure 3.18: Bi-directional workflows, coarsening and uncoarsening strategies.

In Architecture and more generally within the AEC realm, we have seen in Chapter 2 that solely the notion of modelling at different scales is a challenge in itself, when it comes to design a large-scale and complex architectural project. Integrating seamless simulation processes between levels within the same framework would be even more challenging, so the distinction between the modelling and simulation frameworks should be established, although they should ultimately work together and

collaborate. The terminologies “Multi-Scalar Encoding” (as the process of implementing a modelling design workflow accross scales) and “Multi-Scalar Simulation” (which extends the concept of Multi-Scalar Encoding by introducing simulation layers and features that can inform and pass additional information between different levels) are introduced here, in order to give nuance to the broader topic of Multi-Scalar Modelling. Existing scientific research also made the distinction between Multiscalar Modelling and Simulation ([Barth et al., 2009](#)).

- **Multi-Scalar Encoding (MSE)**

MSE implements data structure, hierarchy and optional geometrical links between different objects without simulating any sort of optimizations (structural, material or geometrical). The model resulted from Multi-Scalar Encoding could be compared to other existing models, like Google Map/Earth or other types of Building Information Models that link geometrically or abstractly different scales, without performing any simulation onto or between them. This encoded data structure can be, in some cases, recognized as a **DAG** that could *serve as a picture of a designer’s view of some specific problem* ([Alexander, 1973](#), 81), see figure 3.18a.

- **Multi-Scalar Simulation (MSS)**

A Multi-Scalar Simulation is extending the Multi-Scalar Encoding paradigm, by providing additional layers of simulation features propagated across different scales, which can then steer others based on their own results and outputs. Figure 3.18b presents a Multi-Scalar Modelling environment composed of a series of simulation frameworks acting at different levels (or scales) which extends the initial Multi-Scalar Encoding diagram illustrated in figure 3.18a.

3.3.4 Curating the workflows and enabling feedback

As it has been developed in Chapter 2, custom interfaces also need to be developed in order to enable interoperability across software platforms and the curation of data-rich geometrical objects. In the context of building design and large-scale projects, enabling feedback on large data sets and intricate 3D models needs to be curated very carefully. As it has been explained previously in section 2.3, the processes operating on very large-scales are usually segregated and the models are staged. Therefore, enabling feedback requires to rigorously set up a very clean digital infrastructure that is both able to segregate the levels, encapsulate pipelines and workflows in a hierarchical manner and differentiate modelling from simulation.

As explained previously in section 2.4.3, such infrastructure could be deployed by defining a neutral data format that would improve interoperability and data exchange different software platforms. A database-driven approach would be adopted, instead of a model-driven one ([Ringley, 2017](#)). To enable further simulation frameworks to communicate on the top layer of this digital infrastructure, management interfaces and handshake techniques needs to be developed, allowing the expert user to operate different actions, such as filtering, searching, cross-referencing, sending, pausing, streaming sets of data to different trades.

3.3.5 Conclusion

The present section introduced the concept of Multi-Scalar Modelling which originated within the academic realm (E, 2011), and attempted to transfer its main principles to the AEC industry by transferring the most relevant concepts composing its theoretical framework (such as the “handshaking” techniques) to a more practical level and by observing emerging modelling methodologies within existing practices in industry. Although Multi-Scalar Modelling strategies have been successfully implemented in several architectural design projects, it remains an important challenge to transfer those academic findings to larger scale and more complex projects who need more scalable and robust digital infrastructures to exchange digital models seamlessly and collaborate onto the same project. The 9-point list below regroups and merge both Multi-Scalar Modelling methodologies used in academia (listed in section 3.1.4) and redefined for industry (listed in section 3.3). It also takes into account the different strategies and custom solutions that emerged from practice (listed in section 2.4.2):

- **Defining a neutral data format to facilitate data-rich object customization, transfer and interoperability.**
- **Approximating the models by defining the required levels and resolutions.**
- **Segregating the levels.**
- **Defining the coarsening and uncoarsening strategies to navigate across the previously defined levels and resolutions.**
- **Setting up connected pipelines, design workflows and handshake techniques to deploy coarsening and uncoarsening strategies.**
- **Encapsulating the workflows.**
- **Deploying simulation frameworks and optimization strategies across the previously defined pipelines and design workflows.**
- **Differentiating Modelling from Simulation.**
- **Curating the workflows, improving Interoperability and enabling feedback through the development of custom AEC applications.**

Through this 9-point list, it is crucial to both understand the existing problems in AEC highlighted in the previous chapter and the Multi-Scalar Modelling framework originated from the academic realm, so that the next digital experiments can be carried both with a strong practice-based mindset and relevant theoretical grounding. Based on this list, the next section will focus on the research methodology employed to carry the thesis work. It will highlight how the employed methodology enabled to find a right balance between the academic and industrial realms during the development of the different carried digital experiments which will finally serve as main demonstrators to answer the thesis research questions.

3.4 Research Methodology

The research project follows an experiment-based method (developed in the next subsection) and is built upon 18 different experiments (mapped in figure 3.19) which are categorized into three clusters:

- The first cluster, named “**Modelling-based experiments**” is composed of 9 different modelling experiments that are listed in section 3.4.3.1 and developed in chapter 4.
- The second cluster, named “**Schema-based experiments and search interfaces**” is composed of 6 different schema-based experiments that are listed in section 3.4.3.2 and developed in chapter 5.
- Finally, the third cluster, named “**Cross-practice collaboration experiments**” is composed of 3 different cross-practice collaboration experiments that are listed in section 3.4.3.3 and developed in the last section of chapter 5.

The present section discusses the experiment-based research methodology that has been employed to carry these different clusters of experiments. This section is divided into three subsections. The first subsection introduces the overall research methodology (research *through* design), its context and precedents, as well as the main overall approach that has been chosen to carry the different experiments throughout the thesis. The second subsection describes the evaluation methods which have been deployed to assess the experiments. Finally, the third second subsection is drawing a more detailed map of the carried experiments and explains the different means employed to compose the experiments and the differences that lie in their individual approaches within particular and different contexts (industrial or academic).

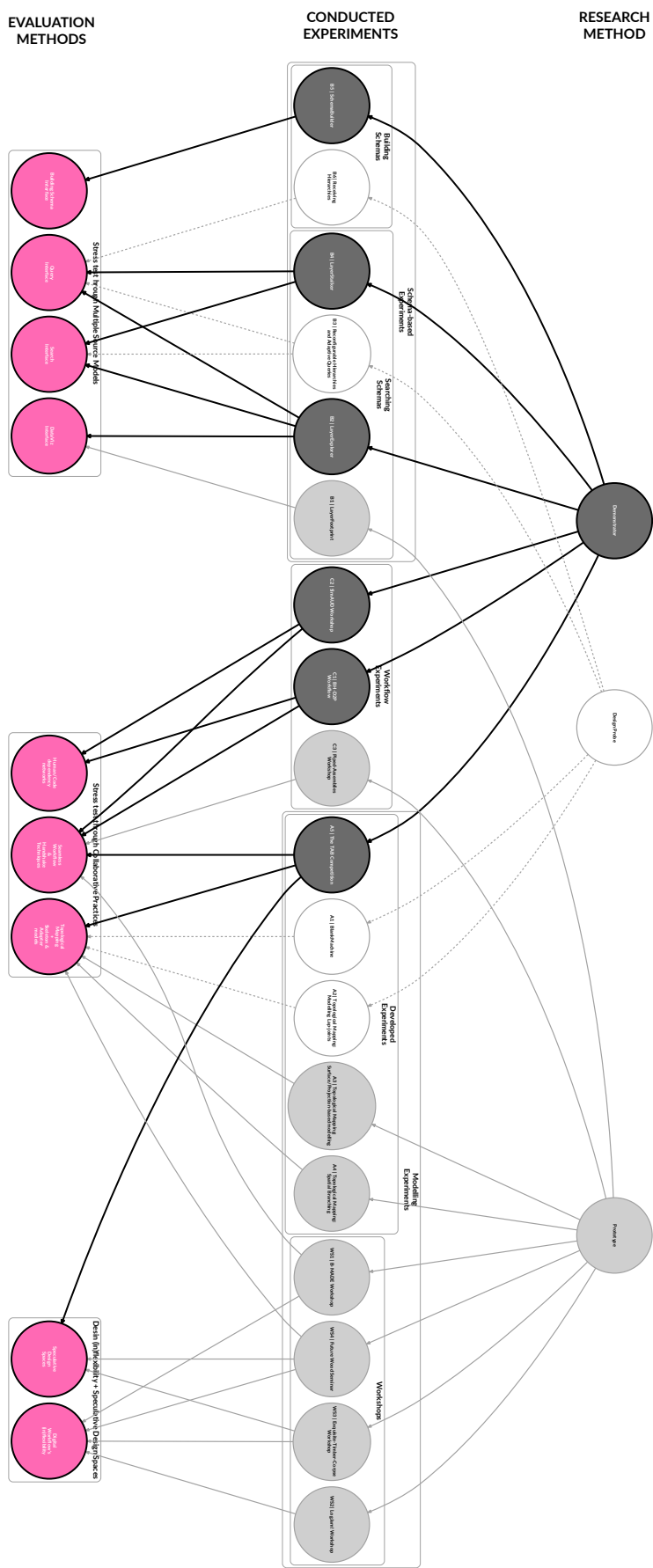


Figure 3.19: This diagram maps the different carried experiments in the central column and classifies them according to their belonging type (design probe, prototype or demonstrator) on the left and evaluation methodology on the right.

3.4.1 Research through Design

The present dissertation's hypothesis asks whether or not the Multi-Scalar Modelling paradigm described in chapter 3 and known in academia could be transferred and help solving industry related issues, by revisiting and extending it through the consideration of practice based challenges. Therefore, the employed research methodology aims at transferring the academic concept of Multi-Scalar Modelling to the industrial realm.

A series of experiments has been conducted to verify the thesis hypothesis. An "experiment" is defined in the Oxford English Dictionary as the following: *"A scientific procedure undertaken to make a discovery, test a hypothesis, or demonstrate a known fact."* As the present thesis interrogates digital design workflows in architectural practice through the Multi-Scalar Modelling spectrum, one of the particularities of the experiments carried out through the thesis work is that they remain within the digital realm. Therefore, instead of being materially-driven, the experiment are both digitally-driven and design-led. Referring to (Ramsgaard Thomsen and Tamke, 2009), Jørgen Hauberg defines design as *"a reflective practice in which critical assessment, comparability and evaluation takes place through sketching, through the continual weaving between problem and solution in an iterative movement between inquiry and proposal."* (Hauberg, 2012). Here, design is meant not only as a prescribed activity, but also as an industrial undertaking within practice.

Digital prototyping

The thesis employs a research methodology that inscribes itself within the Research *through* Design paradigm (Frayling, 1993) and is embedded within a digital prototyping culture. Citing (Petric and Lindsay, 2009), Ramsgaard Thomsen and Tamke consider "digital prototyping" as a valid method to test and verify a "product or process":

"Digital prototyping, or virtual prototyping, is a shared method of digitally designing and investigating a product or process. The aim is to design, iterate, optimize, verify, and visualize a product as it is being created, often by a multi-disciplinary and large design team." (Ramsgaard Thomsen and Tamke, 2015, 52)

Citing (Wong, 2006), the authors also argue even further that "digital prototyping" could replace and be an alternative to physical prototyping:

"digital prototyping is seen as an alternative to physical prototyping and has fully replaced physical prototyping in some fields. Aviation industry companies like Lockheed Martin ceased using physical prototypes in the 1980s; instead, the digital model has become the single place for conducting required verification, simulation and testing." (Ramsgaard Thomsen and Tamke, 2015, 52)

Taking also the aviation industry as an example, Smith observes as well the digitalization of the prototype, the elimination of paper-based drawings and the emergence of the "3D Mockup" as a replacement to the physical prototype:

"Boeing realized too, that the 3D model was the master source of the design and that the elimination of 2D drawings was in its future. They coined the terms "3D Mockup" and "Paperless Manufacturing", with the concept that the majority of the work from design through to the finished part was in a 3D computer digital format. [...] Boeing stated that their new 777 was slated to be the first aircraft designed and manufactured using a three-dimensional computer database entirely. They even went so far as to digitally model a human figure which simulated the feasibility of assembly, access, and maintenance procedures for the aircraft. Boeing's use of CATIA would grow to reach 10,000 workstations by 1996." (Smith, 2017).

In this context, the present thesis is composed of digital probes, prototypes and demonstrators which attempts to improve current digital processes used in industry.

From the design probe to the prototypical application

The different digital experiments carried out during the present research project can be classified and hierarchized across three different categories: “design probe”, “prototype” and “demonstrator”. Those categories have been defined by (Ramsgaard Thomsen and Tamke, 2009, 3) and distinguish different levels and modes of evaluation and evidence. The authors described these three different modes of evaluation as the following:

- **The design probe**

The design probe is an early stage speculative investigation that defines the design criteria and anticipates further research development:

“a design-led investigation allowing speculative inquiry, theorization and the setting out of design criteria.” (Ramsgaard Thomsen and Tamke, 2009, 3)

While the design probe is not as mature and developed as a prototypical application on demonstrator, its role is crucial as it sets the initial research directions leading to further development of the latter.

- **The prototype and the prototypical application**

The prototype – or *material* prototype – is defined by the authors as the following:

“[...] a materially-led investigation allowing exploratory testing, of craft and material behaviour. The prototype answers and develops the design criteria of the design probe” (Ramsgaard Thomsen and Tamke, 2009, 3)

As mentioned above, most of the work in the present thesis is composed of modelling experiments that remain in the digital realm, except for the experiment “A5 - *The Tallinn Architecture Biennale Installation Competition*”, during which several physical prototypes have been produced. In this particular case, the physical prototype can be tested against its digital twin. However, as all the other experiments remain digital, the definition of “prototype” needs therefore to be revisited here in order to include and consider digitally-driven experiments as valid prototypes as well. In this case, a prototype would be defined as an “*investigation allowing exploratory digital testing. The prototype answers and develops the design criteria of the design probe.*” Some prototypes developed in this thesis are defined as “prototypical applications”. Those are application-driven, as their development led to the creation of a particular digital application answering a specific design problem. Contrary to a software application which is robust enough to be deployed through multiple machines and shared amongst different parties, a prototypical application remains at an experimental stage, mature enough to test it against different models or scenarios, but not enough developed to be considered as a proper software application.

- **The demonstrator**

Finally, a demonstrator deploys the prototype (or prototypical application) and tests it under real world constraints. It is defined by the authors as follows:

“[...] an application-led investigation allowing interfacing with real world problems and constraints” (Ramsgaard Thomsen and Tamke, 2009, 3)

Through the present thesis work, a demonstrator is acting in a similar way as the prototype (or “prototypical application”) but deals exclusively with real world data. For example, the cross-practice collaboration experiment “C1 | *Sharing Schemas through Speckle and BHoM: a Speculative Design Workflow between Design-to-Production and BuroHappold*” acts as a demonstrator as it deals with data provided by the consultancy and engineering practices Design-to-Production and BuroHappold Engineering. The demonstrator therefore attempts to solve specific existing problems encountered in practice by providing tailored solutions that emerged through both the design probes and prototypical applications.

3.4.2 Evaluating the experiments

As mentioned earlier, the present thesis attempts to redefine existing modelling practice in industry (chapter 2) through the proposed Multi-Scalar Modelling framework described at the beginning of this chapter. Ultimately, the different modelling experiments and prototypical applications that compose this framework attempt to improve the UX by enhancing flexibility through the whole design process and proposing new ways to manipulate, search, share and build complex geometrical data sets. Evaluating flexibility – or more generally, the UX – through the design process can be problematic as it also relies on subjective inputs (man-machine interaction). It would be possible to analyze digital experiments through data analytics and statistics as in most of the thesis experiments conducted by (Davis, 2013). However, this approach does not inform the researcher or the reader on how the final user experience has been improved. Davis himself warned about *“methodological challenges in developing a convincing understanding of flexible parametric model design”* (Davis, 2013, 10), as the UX is hardly quantifiable and might overrule statistical data analysis. The author highlights here the existing discrepancies that lie within the objective, idealized evaluation of an experiment and its failure to capture the issues of *“complexity, of magnitude and avoiding its bastard chaos.”* which persist in practice (Dijkstra, 1972, 7). As suggested by Davis, it is therefore necessary to dive into practice and conduct experiments from within. (Schön, 1983) describes such strategy as *“reflection-in-action”*. Therefore, specific evaluation methods will be deployed to test the experiments within collaborative practices. Each modelling experiment originates here with a very specific research question, that will be evaluated through the completion of the related experiment, in which the findings, concepts and developed prototypical applications are deployed within an industrial context, in order to test the user experience of AEC practitioners. Except for the “design probe”, which – as mentioned above – is a speculative investigation that is further tested through a further “prototype” and/or “demonstrator”, the conducted experiments that fall in the categories of “prototype” or “demonstrator” will be evaluated through three different “in-action” methods which attempt to answer for each case a specific research question. These methods rely on “stress testing” the experiment through different means. The Oxford English Dictionary defines a “stress test” as a *“test of cardiovascular capacity made by monitoring the heart rate during a period of increasingly strenuous exercise.”* in the medical realm, and more generally as a *“test designed to assess how well a system functions when subjected to greater than normal amounts of stress or pressure.”* In the realm of industrial software development, “stress testing” is a software testing activity which evaluates and verifies the stability and reliability of the software’s system by testing it under heavy load conditions and beyond the limits of normal operation (Weyuker, 1998). As the present thesis work does not pretend to undertake robust and industrial software development, the term “stress testing” refers here to the deployment and testing of a prototypical application or prototypical design tools against multiple models and/or case scenarios. In the present thesis, stress testing also refers to the act of continuously designing a prototype until the latter works as initially intended. As a designer, **a part of the design process is to continuously stress test the model or experiment until the latter fulfils the desired requirements.**

The three “in-action” methods mentioned above are listed below:

- **Stress testing the design workflow’s (in)flexibility**

The first evaluation method introduced here relies on the subjective user experience, which measures whether or not the speculative probe or prototype allows design flexibility and enhance design possibilities. Those qualities are difficult to assess objectively and will be therefore investigated in practice to evaluate the initial design probes and to determine concepts and tools which can be further developed through the next prototypes and demonstrators.

- **Stress testing prototypical applications against existing multiple source models**

Several modelling experiments and demonstrators – LayerExplorer, LayerStalker and SchemaBuilder – result from the development of prototypical applications. These applications have been tested through their deployment and stress tests against different building models to either extract and visualize specific data (as for LayerExplorer and LayerStalker), or to build custom schemas (SchemaBuilder). Furthermore, the prototype LayerFootprint (which

preceded the development of LayerExplorer and LayerStalker) has also been stress tested against multiple Rhino3D model to evaluate the inherent complexity of the model's respective layer tables.

- **Stress testing prototypical design tools and methodologies through collaborative practices and speculative workflows**

Other modelling experiments, prototypes and demonstrators were stress tested through different design tools and methodologies (previously developed within former design probes or prototypes) by deploying them through the establishment of collaborative practices which investigate speculative design workflows between different people, actors or trades during the design process.

The above listed experiments have been mapped in figure 3.19 which depicts for each experiment its belonging category (design probe, prototype or demonstrator) in the left column and its related evaluation mode in the right column.

3.4.3 Mapping the experiments

Different means of composing experiments have been previously identified in “*Ways of Drifting – 5 Methods of Experimentation in Research through Design*” (Krogh et al., 2015). In this article, the authors identify by connecting design experiments five different means to produce knowledge and conduct experiments, inscribed within the Research-through-Design (Frayling, 1993) paradigm: the *Accumulative*, *Comparative*, *Serial*, *Expansive* and *Probing* approaches. Each of these different means of composing experiments have been extracted individually from each one of the cited original thesis works which have respectively be composed by one of these means. Instead of focusing on one of these means in particular, the present thesis connects and hybridizes those different concepts to conduct experiments differently according to particular contexts. Before developing further on how such hybridization inscribes itself within the present thesis work, the different means of conducting the experiments are explicated below:

- **Accumulative**

Derived from each carried experiment, the *accumulative* approach builds iteratively (by layering, stacking) an increasing depth of knowledge into the next generation of experiments. Therefore, the *accumulative* learns constantly from the precedents in an empirical manner.

- **Comparative**

The *comparative* approach explores the subject “*by means of a number of design cases working from or towards a shared platform of comparison.*” (Krogh et al., 2015). In this case, different experiments are conducted separately and in different contexts, although with the purpose of proposing a common framework for further analysis and comparison.

- **Serial**

The *serial* approach focuses on the order in which the experiments are carried out, and how they mutually influence each other though the chosen research sequence or logic order: “*[...] knowledge production in the serial approach is achieved on the basis of insights gained into the relationships between design experiments that proceed chronologically.*” (Krogh et al., 2015).

- **Expansive**

Unlike the *serial* approach, the *expansive* one does not follow “*strict successive or linear orders or directions to follow. [...] Rather than deepening [...] knowledge of a domain, this approach widens [the] perspective and extends the concerns [that designers] should include in [their] praxis.*” (Krogh et al., 2015). The expansive approach therefore attempts to broaden the research perspective by investigating multiple domains in parallel.

- **Probing**

The *probing* approach reveals itself after comprehending the different concepts and approaches listed above: “*Yet, it is only when we examine probing in relation to the other four approaches that its methodological value for design research can be fully grasped.*” (Krogh et al., 2015). Generally,

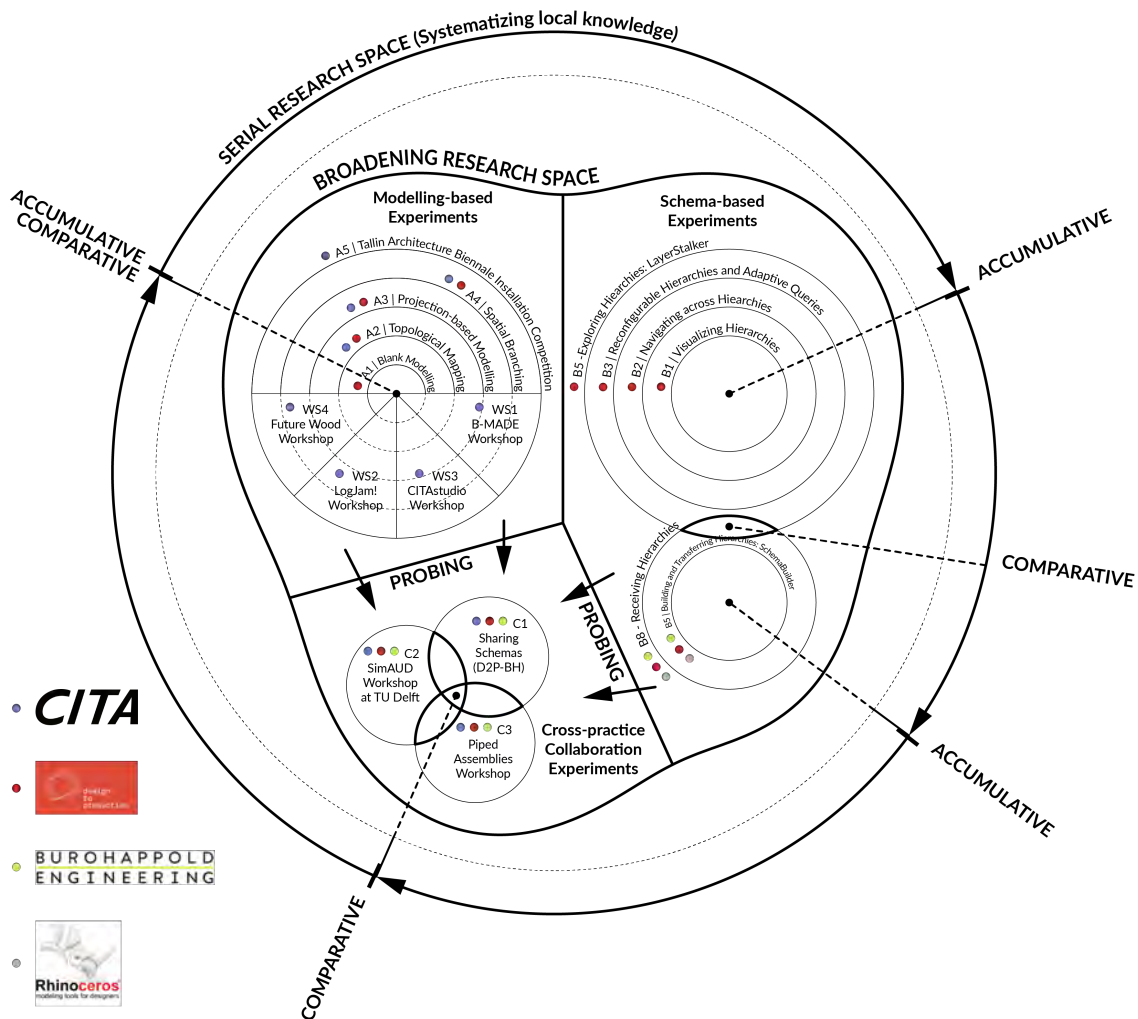


Figure 3.20: The experiments carried out during the thesis work are mapped on this diagram, following different approaches and means of conducting experiments, such as the *accumulative*, *comparative*, *serial*, *expansive* and *probing* research through design approaches described by (Krogh et al., 2015). Those different approaches take into consideration the different context within which the experiments took place: in an industrial context at Design-to-Production, BuroHappold or McNeel or in academia at CITA.

the probing approach exploits “opportunities and [explores] design ideas as they emerge through design work.” (Krogh et al., 2015). The keywords used by the authors to define the choice of the experiments are “illogical”, “artistic” and “impact oriented”.

Intertwining academia and practice research perspectives

In (Krogh et al., 2015), the listed approaches are seen as different and isolated alternatives to conduct experiments. In contrast, the present thesis use them in tandem, as the work has been shaped through a back-and-forth dialog between academia and practice, necessitating versatile means of conducting experiments to adapt to different contexts. As described in the introduction, the author pursued secondments both at Design-to-Production, BuroHappold and McNeel during five distinct periods totalizing respectively 7 weeks at Design-to-Production, 5 weeks at BuroHappold and 8 weeks at McNeel. The research perspectives were therefore constantly balanced between the respective modes of thinking of each industry and academic supervisors. In such context, the *comparative* and *accumulative* approaches were used to both broaden the research perspectives when meeting with each individual and to direct the overall research towards a converging goal by comparing and merging the different view points.

The next section will describe and map how the different series of thesis experiments have been carried, taking into account both the five means of conducting experiments described by (Krogh et al., 2015) and the constant dialog between the academic and industrial contexts.

In the present thesis, experiments are carried to test the hypothesis of a Multi-Scalar Modelling paradigm for building design.

The series of experiments carried out during the thesis work borrows its research approaches from each of the ones described above. Experiments have been carried out both in an accumulative, comparative, serial, expansive and probing manner, depending on the specific context. On a global level, the experiments have been performed following the serial research approach which successively introduced three new research perspectives and contexts: academia, practice and an hybridization of the two. Whereas the academic realm provided the conceptual framework of Multi-Scalar Modelling (as described in chapter 3), the industrial domain grounded the research by analyzing existing practices that already push the boundaries of digital workflows in architecture (as described in chapter 2). The carried experiments therefore entertain a back and forth relationship between these two domains and attempt to produce knowledge by extracting the respective concepts and methodologies that would contribute to the general concept of Multi-Scalar Modelling applied to building design and AEC.

3.4.3.1 Modelling-based experiments

Grounded within the academic realm, a first series of experiments (detailed in chapter 4) has been achieved following the accumulative logic, during which the knowledge gained from the previous experiment has been used to pursue the next one. These experiments ground themselves both in the context of Multi-Scalar Modelling in academia, and through the observation of free-form timber modelling practices at Design-to-Production. These different experiments are listed below:

- **A1 | BlankMachine: Propagating the blanks**
- **A2 | Topological Mapping: Modelling Lap Joints**
- **A3 | Topological Mapping: Surface/Projection-based modelling**
- **A4 | Topological Mapping: Spatial Branching**
- **A5 | The Tallinn Architecture Biennale Installation Competition**

By analyzing the modelling complexities at every scale (from the lower to the higher one) in the design process of a free form timber structure, each of these experiments unlocks, discovers a new level of complexity during modelling and unravels the experiment's internal limits. This leads to the establishment of the next experiment that attempts to manage a higher level of complexity. In parallel to the modelling experiments described above, the author taught and participated in four different workshops that helped him to contextualize and test the knowledge and findings gathered within the precedent series of experiments, following here the *comparative* research approach described above.

- **WS1 | B-MADE Workshop: Fabrication Constraints and Assembly Logistics**
- **WS2 | LogJam! Workshop: Structural Optimizations and Performance**
- **WS3 | Exquisite-Timber-Corpse Workshop: Collaborative Practice**
- **WS4 | Future Wood Seminar: Speculating Design Spaces**

3.4.3.2 Schema-based experiments and search interfaces

A second series of schema-based experiments (detailed in chapter 5) has been pursued, grounded this time in an industrial context and following the accumulative research approach as well. Instead of focusing on the topic of Multi-Scalar Modelling (as understood in the academic context), these experiments broaden the thesis research questions and the Multi-Scalar Modelling domain by integrating the constraints and challenges faced by the building industry. This series of experiments focuses on the topics of data management, visualization, query, manipulation and flexibility during the post-tender phases of a complex architectural design project. Here, the *accumulative* research

approach and the knowledge gained throughout the series lead to the development of two experimental web-based applications, such as *LayerFootprint*, *LayerExplorer*, *LayerStalker* and *SchemaBuilder*:

- **B1 | Visualizing Hierarchies: LayerFootprint**
- **B2 | Navigating across Hierarchies: LayerExplorer**
- **B3 | Reconfigurable Hierarchies and Adaptive Queries**
- **B4 | Exploring Hierarchies: LayerStalker**

In parallel, another series has been conducted, focusing this time on the building, sharing and receiving of standard and open custom schemas that are able to aggregate complex data sets through different scales. It leads to the development of an experimental web-based application named “SchemaBuilder”.

- **B5 | Building and Transferring Hierarchies: SchemaBuilder**
- **B6 | Receiving Hierarchies**

Here, the *accumulative* approach is employed as well, although the *comparative* approach has been used to compare the findings from both the present series (B5-B6) and the previous one (B1-B4). Indeed, those two lists share common areas of interests looking at schemas and open-standards, although the first series of experiment (B1-B4) focus on the exploration of complex, hybrid and intertwined schemas, whereas the second series (B5-B6) explores new ways of building them from the bottom-up.

3.4.3.3 Cross-practice collaboration experiments

Finally, the last series of experiments (developed in section 5.3) attempts to merge the findings of the two series developed above by extracting the relevant concepts and methods that could contribute to the theoretical framework of Multi-Scalar Modelling for Building Design and AEC. Here, as the opportunity of conciliating both the academic and industrial realm emerged through the precedent series of experiments, the *probing* research approach has been used. The list of experiments conducted here is constituted of both a case study conducted between the author, Design-to-Production and BuroHappold, as well as two different workshops lead by the author in which have been introduced tools and concepts coming from the academic (the Multi-Scalar Modelling perspective) and industrial realms (schema-based workflows for building design).

- **C1 - Sharing Schemas through [Speckle](#) and [BHoM](#): a Speculative Design Workflow between Design-to-Production and BuroHappold**
- **C2 - SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows**
- **C3 - Piped Assemblies Workshop at CEDIM**

In the present section, the five different research methods identified by ([Krogh et al., 2015](#)) – the *Accumulative*, *Comparative*, *Serial*, *Expansive* and *Probing* approaches – have been used in various contexts and at different levels, as illustrated in figure 3.20.

3.4.4 Conclusion

The research methodology described in this chapter grounds itself in the Research through Design (RtD) paradigm (Frayling, 1993). Depending on the research context – whether it is pursued within an academic or industrial context – different research methodologies have been deployed, as described by (Krogh et al., 2015): the *Accumulative*, *Comparative*, *Serial*, *Expansive* and *Probing* approaches. Furthermore, the thesis experiments have been evaluated through different modes belonging to specific categories defined and established by (Ramsgaard Thomsen and Tamke, 2009): the *Design probe*, the *Prototype* and the *Demonstrator*.

Each pursued experiment attempts to answer a specific research question that will be evaluated as well against different *in-action* methods described above: the model's (in)flexibility and design potentials, the experiment's stress tests against multiple 3D models and through "*in-action*" collaborative practices and speculative workflows.

The next chapter will describe the first series of modelling-based experiments and conducted workshops, and the last chapter will describe the second and third series of experiments (respectively schema-based and cross-practice collaboration experiments). All the experiments will be tested through the three "*in-action*" methods (Schön, 1983) described above.

Chapter 4

MODELLING-BASED EXPERIMENTS: FREE-FORM TIMBER STRUCTURES AS A SERIES OF CASE-STUDIES

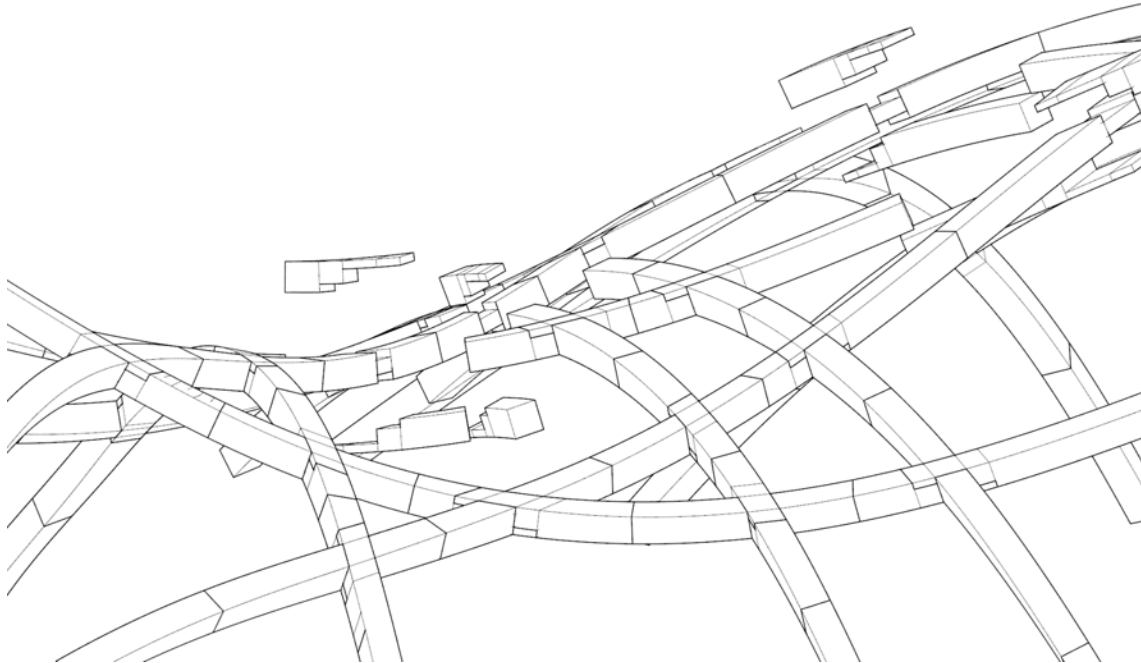


Figure 4.1: Discretization of a complex timber structure into n lap joints.

The present chapter focuses on modelling-based experiments and is divided into two main sections.

The first section studies the topological mapping of complex timber structures through the scope of existing modelling practices at Design-to-Production. The author interrogates here the application of the Multi-Scalar Modelling paradigm through a first series of modelling-based experiments aiming at improving Design-to-Production's existing practices. A majority of the modelling-based experiments described in this section first analyzes an existing precedent on which Design-to-Production was involved, extracts specific design methodologies that were used and the challenges that were tackled during the project, and proposes alternative solutions – either conceptual and/or practical – in order to enhance the existing design workflows. For each case study, speculative digital experiments aim at enabling seamless connected modelling pipelines and workflows across different scales, focusing both on modelling and data management concerns.

The second section aims at *applying* the Multi-Scalar Modelling strategies in the course of various workshops, seminars and research projects undertaken in the framework of InnoChain's teaching program. Therefore, those experiments focus on the dissemination, application and speculation of the gathered knowledge produced in the first section. The last modelling experiment of this second

series describes a collaborative work with Tom Svilans (ESR 2) which focused on a design development and proposal for the Tallinn Architecture Biennale Installation Competition in 2017. Finally, the present chapter concludes by learning both from the case studies and the carried digital experiments, extracting and analyzing the most prominent remaining challenges that will be further tackled in the last chapter.

4.1 Topological Mapping of Complex Timber Structures

This first series of experiments focusing on the topological mapping of complex timber structures has been achieved following the accumulative logic (Krogh et al., 2015), during which the knowledge gained from the previous experiment has been used to pursue the next one. These experiments ground themselves both in the context of Multi-Scalar Modelling in academia, and through the observation of free-form timber modelling practices at Design-to-Production.

4.1.1 From blank modelling to global networks: extracting and navigating through the different levels of resolution

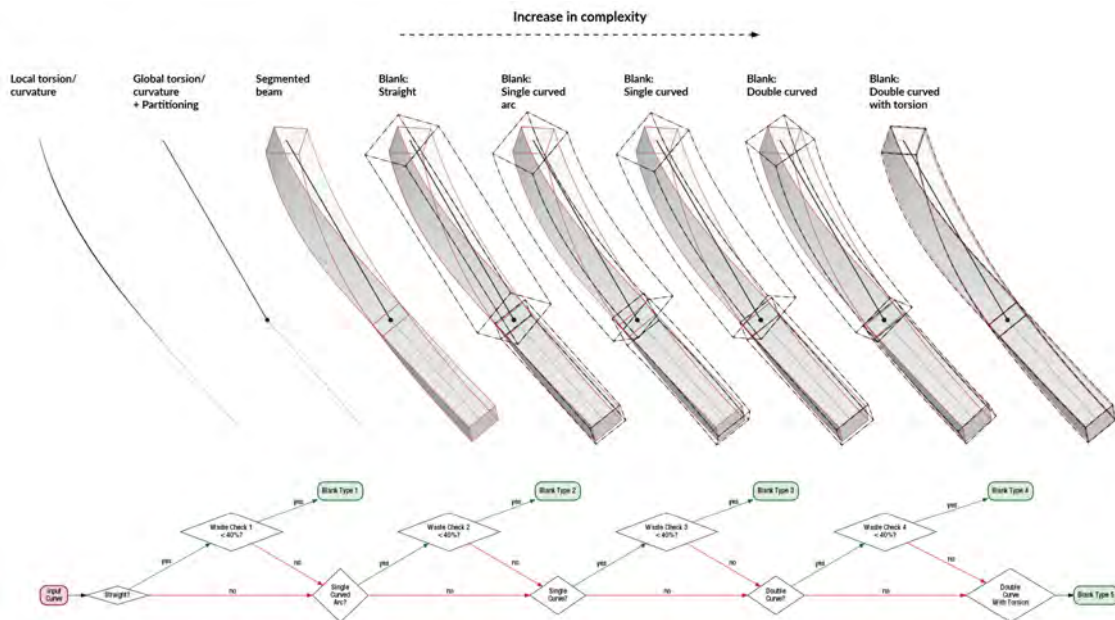


Figure 4.2: This If-Then-Else flow diagram represents the selective blank modeling process used at Design-to-Production.

This first experiment explores the blank modelling processes used at design to production and proposes a digital prototype that helps to automatize the creation of blanks at high level in the design process. Before describing the modelling experiment, it is necessary to understand the existing modelling process used at the consultancy practice Design-to-Production and to extract its inherent logic.

4.1.1.1 The modelling techniques developed and employed at Design-to-Production

Collaborating very often with timber manufacturers on challenging and complex wooden projects, the consultancy practice Design-to-Production has become expert in delivering data to fabricate geometrically intricate wooden members. Usually, each timber beam needs to be modelled along with its corresponding “blank”. A blank is the glue laminated raw material that is formed before milling, and has been defined by Design-to-Production according to the following: *“Once the beam segmentation is set, blanks are created. The term “blank” refers to the raw glue-laminated timber*

piece, which is milled down to the final shape of the segment. In free form projects, single curved or doubly curved blanks approximate the final piece's curvature. They are produced by bending and gluing raw lamellas on a set of pre-shaped supports.” (Usai and Stehling, 2017)

Depending on the geometrical complexity of the initial beam segment, five different types of blanks have been identified by Design-to-Production for custom fabrication strategies: “Straight”, “Single-curved”, “Single-Curved Arc”, “Double-Curved without Torsion” and “Double-Curved with Torsion”. For each type of blank, a specific computational design workflow is applied to generate the 3D model necessary for further fabrication. The following paragraphs describes the different working methodologies as well as the selective process that Design-to-Production runs in order to define which specific type of blank would best fit a particular timber beam element.

- **Method 1 - Straight Blank**

If the beam is considered as being straight, the blank will be modelled by offsetting non-uniformly the beam's faces (top, bottom, left and right).

- **Method 2 - Single-Curved Arc Blank**

If the beam is considered as a single-curved arc, Method 1 will be applied first by approximating the single-curved arc into a straight line. If the curvature and the percentage waste are too important, the blank will be calculated by offsetting non-uniformly the beam's faces (top, bottom, left and right).

- **Method 3 - Single-Curved Blank**

If the beam is considered as single-curved, Method 1 will be applied first by approximating a single curve into a straight line. If the percentage waste is too important for Method 1, Method 2 will be applied by approximating a single curve into a single-curved arc. If the percentage waste is too important for Method 2, the blank will be calculated by offsetting non-uniformly the beam's faces (top, bottom, left and right).

- **Method 4 - Double-Curved Blank**

If the beam is considered as doubly curved, Method 1 will be applied first by approximating a double curve into a straight line. If the percentage waste is too important for Method 1, Method 2 will be applied by approximating a double curve into a single curved arc. If the percentage waste is too important for Method 2, Method 3 will be applied by approximating a double curve into a single curve. If the percentage waste is too important for Method 3, the blank will be calculated by offsetting non-uniformly the beam's faces (top, bottom, left and right).

- **Method 5 - Double-Curved Blank w/ Torsion**

If the beam is considered as doubly curved with torsion, Method 1 will be applied first by approximating a double curve (w/ torsion) into a straight line. If the percentage waste is too important for Method 1, Method 2 will be applied by approximating a double curve (w/ torsion) into a single curved arc. If the percentage waste is too important for Method 2, Method 3 will be applied by approximating a double curve (w/ torsion) into a single curve. If the percentage waste is too important for Method 3, Method 4 will be applied by approximating a double curve (w/ torsion) into a double curve (w/o torsion). If the percentage waste is too important for Method 4, the blank will be calculated by offsetting non-uniformly the beam's faces (top, bottom, left and right).

The workflow described here aims at finding the most suitable blank type, fitting both fabrication, structural and economical criteria. It is often hard to find the right balance between those different benchmarks, as the best fitting blank satisfies structural requirements (good alignment of the fibers), but is harder to fabricate and subsequently much more expensive. On the other hand, the cheapest blank hardly satisfies structural requirements (poor alignment of the fibers) and generates more material waste. Referring to the Seine Musicale project as an example, Hanno Stehling and Fabian Scheurer from Design-to-Production explain: *“Beam segments for structures like the one discussed are usually CNC-milled from a mixture of straight, single- curved and double-curved glue-laminated timber blanks. The decision of which type of blank to use is a trade-off between structural strength, material cut-off and lamination costs.” (Stehling et al., 2017)*

4.1.1.2 A1 | BlankMachine: Propagating the blanks

Experiment A1 | BlankMachine: Propagating the blanks Category: design probe



Video link to the experiment

Related research question: “How can early stage data be leveraged to improve late stage design decisions?”

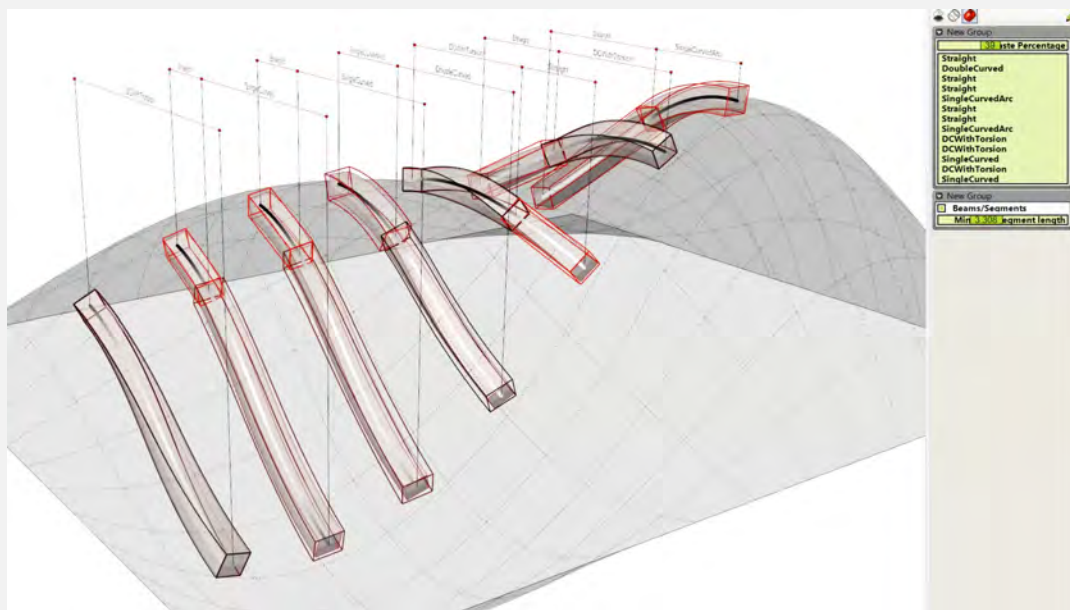


Figure 4.3: The BlankMachine's UI (right panel) enables the computational designer to specify with a slider a maximum waste percentage that will steer the blank's geometry (left), following the If-Then-Else modelling pipeline described in figure 4.2. Furthermore, the user can choose whether or not the beams should be segmented into multiple parts according to their respective degree of complexity.

Description

This present experiment asks whether or not early stage data can be leveraged to improve late stage design decisions. This aspect is important to consider within the Multi-Scalar Modelling framework, as high-resolution objects depend ultimately on more abstract, lightweight representations that have been designed at earlier stages. The case study described above is depicting the modelling techniques developed by Design-to-Production to calculate the appropriate wooden blank for each unique timber element. This process is still segregated into different sub-processes that need to be carefully curated by the consultancy practice. The present experiment therefore attempts to bridge the gap between early stage and late stage design workflows by taking into account abstract parameters at the earliest stages that would inform later stage design decisions.

The experiment consists of a digital prototypical tool aiming at automatizing the creation of blank types. This digital prototype bases itself on the above described methods to select a particular blank type. The selection process happens here through an If-Then-Else modelling pipeline (described in figure 4.2), which starts from the simplest blank type and examines if

such type suits well for the fabrication of the considered wooden member, according to the resulted waste percentage. The general rule says that if the chosen blank generates more than 40 percent of waste, one should try to fit the beam within a next slightly more complex blank. This rule goes on until the material waste finds itself below 40 percent. For the same building component (Figure 4.2), it is observed that while one part would probably be fabricated with a double-curved blank with torsion, another part could already be very nicely approximated with a simple straight blank. The computational designer can therefore separate the concerns and apply different rationalization strategies for different parts of a same beam.

User Interface

Within the developed prototypical interface named “BlankMachine” (Figure 4.3), the computational designer is able to select a particular beam segment and generate its corresponding blank based on a specific waste percentage. The higher the latter is, the simpler it is to fabricate the blank, reducing subsequently its cost. Reciprocally, the lower the waste percentage is, the more complex it is to fabricate the blank, increasing its cost. On the right panel, the user can specify with the help of a slider the maximum tolerated waste percentage. For each selected wooden member, the implemented algorithm will take this input value into account and generate a blank that will respect the maximum waste percentage set by the user, following the If-Then-Else modelling pipeline described in figure 4.2. The UI integrates a panel displaying the category of each generated blank: “Straight”, “Single-curved”, “Single-Curved Arc”, “Double-Curved without Torsion” or “Double-Curved with Torsion”.

The algorithm analyzes the local curvature between every frame forming the element’s twist. Based on this initial curvature analysis, a K-Means clustering algorithm has been implemented to find and partition the beam into sub-segments depending on their levels of complexity: while some sub-segments can be rationalized using the simplest blank modelling methodology, others (the most twisted ones) will be fabricated with the most expensive and complex blank modelling algorithm. Through the interface, the computational designer is also able to specify whether the blanks should be generated from the full selected wooden member or from its subsequently generated sub-segments. Finally, another slider also allows the user to set a minimum length for the generated sub-segments.

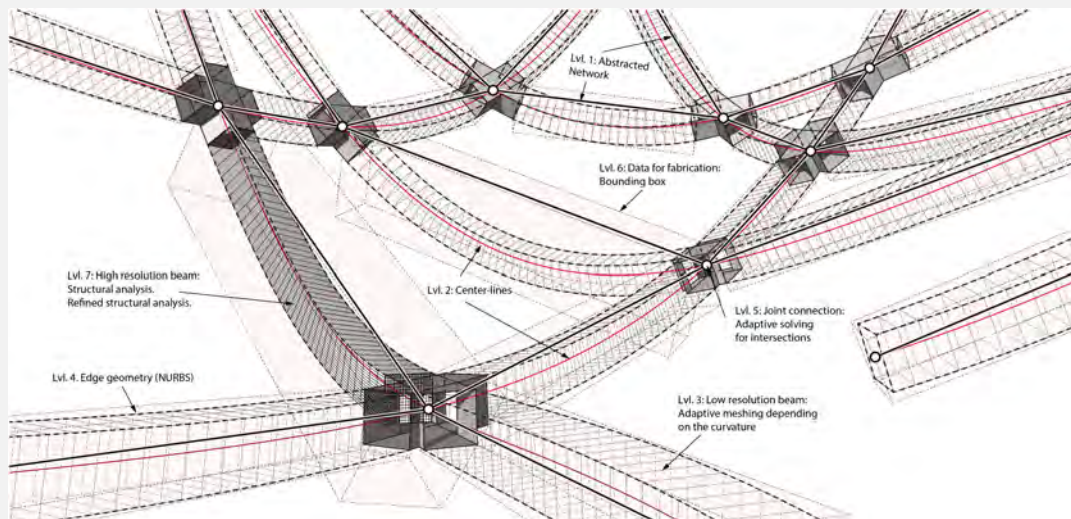


Figure 4.4: The different extracted levels of resolution.

Results

Through several stress tests against different glue-laminated timber beam models generated by the author, this prototypical tool proved to be useful to generate blanks, steer their geometries at high resolution by manipulating low level parameters. It gave the end-user

constant feedback on the generated blank type and corresponding waste percentage. Here, early stage data has therefore been leveraged to improve late stage design decisions. Once the selective blank modelling process has been defined to detect the blank of a particular timber element, one can start to instantiate the algorithm onto multiple elements and across an entire structure (Figure 4.4). Through the *accumulative* research approach defined previously in section 3.4, the next experiment will take into consideration the blank modelling knowledge gained during the development of the BlankMachine prototype while focusing this time on a higher level, through the modelling of a large network of interconnected glue-laminated timber beams.

Future work

The next modelling experiment tackles the scale of the global structure by operating topological mapping of complex wooden joints between different timber beam elements. Before describing the modelling experiment, it is also necessary here to understand the context by analyzing specific case studies, such as the the spa and health center Solemar-Therme in Bad Dürkheim and the Expo French Pavilion modelled by Design-to-Production which dealt with many different levels of resolution across the digital chain, from the global design scale to the blank modelling processes. This analysis will serve here to transition to the next digital experiments that focused on multiscale modelling workflows for the design of free-form timber structures.

4.1.2 Topological mapping in timber assembly

The second experiment described in this section acts as a design probe: a speculative investigation that defines the design criteria and anticipates further research development. The present experiment tackles the scale of the global structure is tackled by taking into account multiple levels of resolution. Before detailing the modelling experiment, one needs to understand what are the different existing levels that needs to be taken into consideration during the modelling process of free-form timber structures. This will be illustrated by both the spa and health center Solemar-Therme in Bad Dür rheim and key projects modelled by Design-to-Production, such as the France's Expo 2015 Pavilion, the Nine Bridges and the Cambridge Mosque projects.

4.1.2.1 The spa and health center Solemar-Therme in Bad Dür rheim

The spa and health center Solemar-Therme in Bad Dür rheim is a very good example that illustrates how different levels of resolution have been “aggregated” on top of one another through the design process of a free-form timber structure, from early design stages to data production for fabrication. During the form-finding process, the shell surface was discretized into a network of straight center lines which represented abstractly the solid geometries that were later generated for fabrication purposes. Klaus Linkwitz and Diederik Veenendaal have described each design step of this project ([Linkwitz and Veenendaal, 2014](#)):

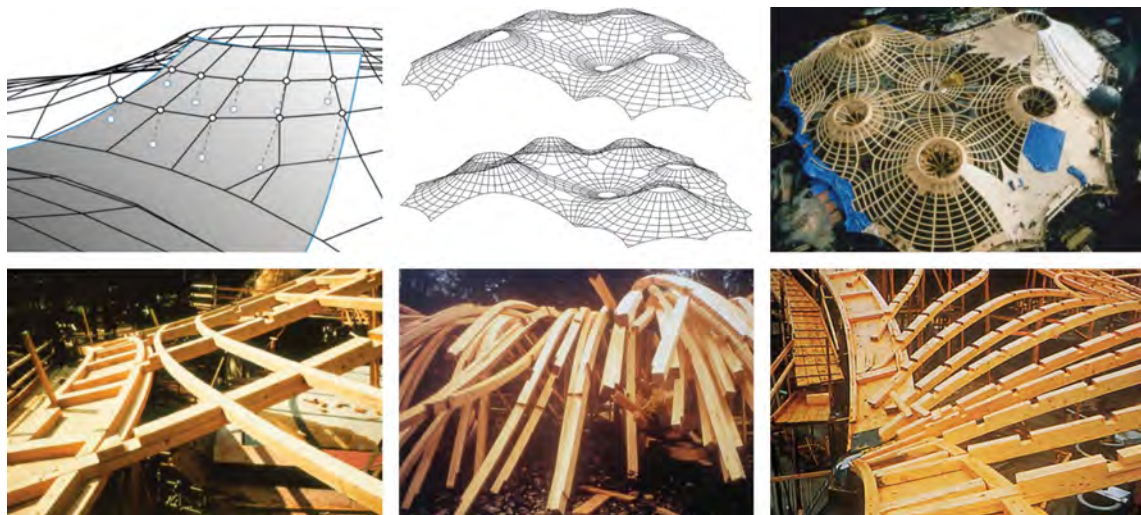


Figure 4.5: From left-to-right and top-to-bottom: target surface with projected initial coordinates and resulting solution from nonlinear FDM (Force Density Method), the final equilibrium shape, construction site with all ribs assembled and placement of covering boards and sheeting, overlaid with the annular ribs, prefabricated strongly doubly curved glulam beams ready for transport to site, and meridian ribs with cross lap joints, attached to the edge beam. ([Linkwitz and Veenendaal, 2014](#))

- **Level 1: Abstracted Network**

It is possible to discretize the shell's surface as a spatial net, a network of straight center lines. *"This wireframe does not yet allow a unique definition of the three-dimensionally curved glulam ribs."* ([Linkwitz and Veenendaal, 2014](#))

- **Level 2: Center-lines**

To calculate the beams center-lines, normal and tangent vectors along the ribs were derived from consecutive sets of discrete points along their paths. *"Alternatively, this information can be calculated by first tracing splines along the points of each discretely represented rib as an intermediate step."* ([Linkwitz and Veenendaal, 2014](#))

- **Level 3: Approximated tridimensional representation (meshing)**

From the previously generated center-lines, one can approximate – by means of offsets – the global geometry of each glulam beams. Their respective curvature can be analyzed (locally and globally) and adaptive meshing techniques can be used to model each element. Color gradient can be mapped on these geometries by measuring local or global parameters such as the planarity property of each mesh face.

- **Level 3(bis): Brep representation - Edge geometry**

In parallel to Res.3, it is possible to calculate by means of multiple offsets the edges and the solid geometry of each glulam members.

- **Level 4: The joint condition**

Joint techniques are particularly crucial when it comes to the assembly process. *“[...] it is important that all their contact surfaces are planar, such that an adequate fit is achieved in assembly. The assembly on the construction site will be more economical if all joints are also prefabricated, avoiding any additional labour on site. The CAD modeling of the cross lap joints and the monitoring of their prefabrication were therefore of the utmost importance.”* (Linkwitz and Veenendaal, 2014)

- **Level 5: Data for fabrication (blanks and lamellas)**

From each generated solid geometry, one can calculate the minimal bounding box of each member in order to mill and fabricate it numerically: *“In order to calculate the patterns of the ribs for fabrication, the coordinates were transformed to a local coordinate system for each rib. The first axis was defined by the two end points of the rib segment. The other two orthogonal axes were then defined by a best-fitting plane of the three-dimensionally curved rib. This local coordinate system allowed a convenient set-up of the manufacturing tools, minimizing the required heights above the manufacturing floor.”* (Linkwitz and Veenendaal, 2014)

- **Level 6: Material performance**

It is possible to investigate a much higher resolution by analyzing the material performance of a glue-laminated timber beam. One can modify and explore different types of parameters such as: type of glue used, lamination techniques, introducing new material into the process (such as glass-fiber, for example), etc.

Identifying and enumerating the different resolutions that usually emerge during the design process of a free-form timber structure was very helpful in order to conduct first abstract investigations on multi-scalar modeling techniques. Figure 4.6 drafts a first Multi-Scalar Model for free-form timber structures, laying out four main different levels and resolutions that should be taken into account in the design process:

- **High Level**

At high level, different global geometrical analysis can be deployed, ranging from graph manipulations and partitioning for assembly strategies (K-Means, cycles detection, neighboring analysis) to geometrical optimizations and simulations (dynamic relaxation for Bending radii maximization, geodesic path search onto a master surface for blank type optimization, etc.), global structural analysis (manipulation, addition and removal of supports), and global design decisions (manipulation of the design targets such as the master surfaces control points).

- **Intermediate Level**

At intermediate level, the focus is placed on regional assembly strategies and analysis through the subdivision of the entire structure into multiple node “patches”, allowing partial geometrical description, and the extraction of forces from the global abstract network.

- **Low Level**

At low level, each wooden node is fully geometrically described and detailed following an initial typology detection based on local and neighboring conditions. The user is then able to manipulate the resolution settings and detect the corresponding blank type. Finally, local structural analysis can be run by means of forces coming from the intermediate or global high levels (or global networks).

- **Material Level**

At the material level, the machine's morphospace ([Menges, 2012](#)) and material behavior can be further analyzed through the local fabrication constraints. Fabrication tolerances can also be evaluated through scanning technologies and comparison between the scanned point cloud and the original 3D model.

This first draft for a theoretical Multi-Scalar Model also takes into account feedback strategies that could happen from and to any level: The Material-to-Local feedback would introduce physical and fabrication constraints, the Local-to-Intermediate and Material-to-Global feedbacks would inform on the readjustment of the nodes locations, the Intermediate-to-Global feedback would enable the repartitioning of the global graph, and the Local-to-Global feedback would re-inform on the global segmentation strategy.

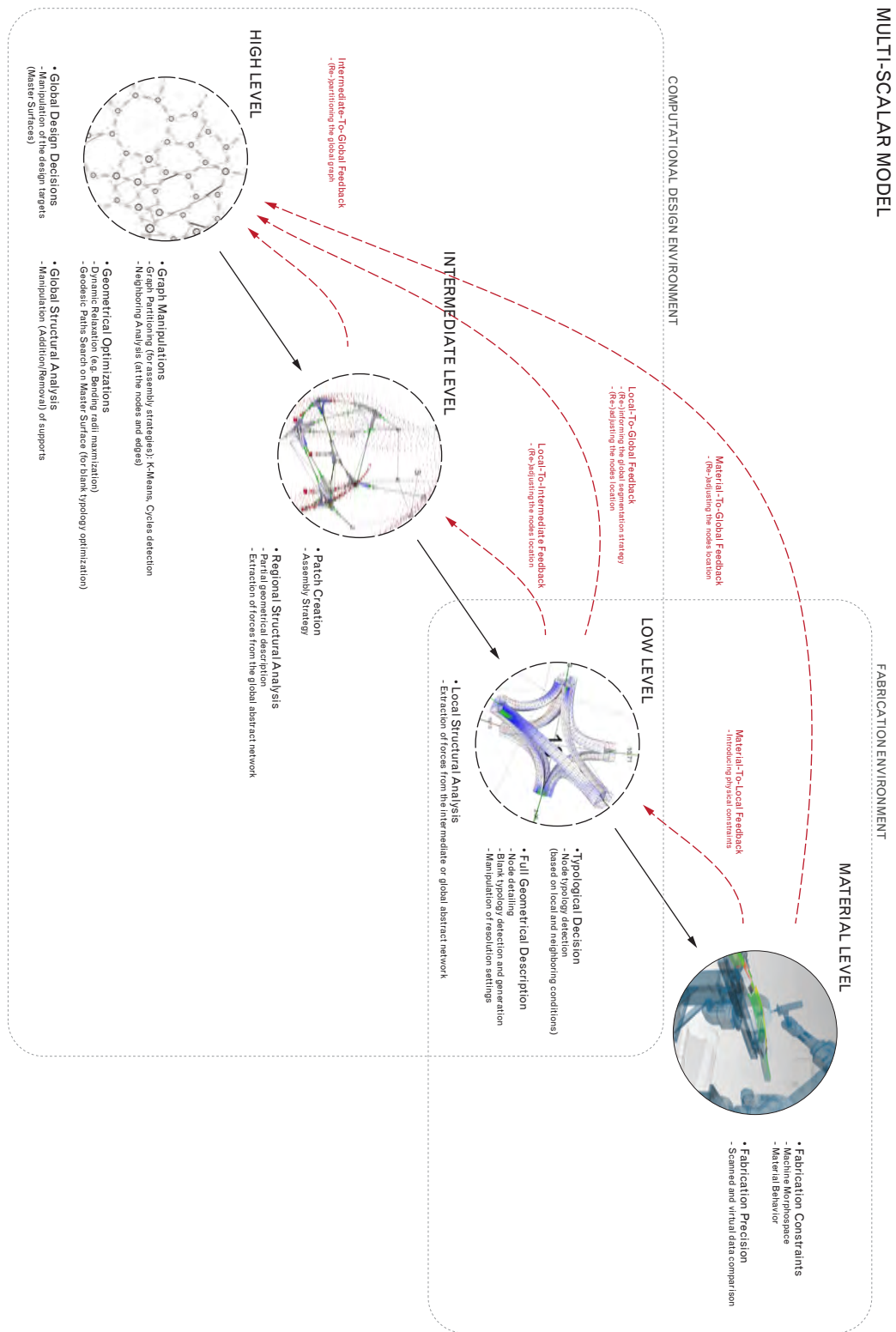


Figure 4.6: From high level to material level and feedback strategies.

4.1.2.2 Design-to-Production's expertise

Most of the projects tackled by Design-to-Production are composed of a complex web of wooden members interfaced with unique connection details. It is therefore important to create a clear abstract two-dimensional map of the construction project to locate and track each single detail and geometrical data sets from conception to fabrication. In this particular case, the main challenge here is to transfer the 3D geometrical data to a 2D map so that the representation of the latter is as simple as it can be in order to understand it, and as faithful as it can be in order to match with the reality.

- **The Expo French Pavilion**

During the post-tender phase of the French Pavilion exhibited at the EXPO 2015 in Milan, information exchange between the trades often relied on two-dimensional representations of the timber structure. Excel-tables or simple diagrams eased the access of complex data for both the engineer and the fabricator:

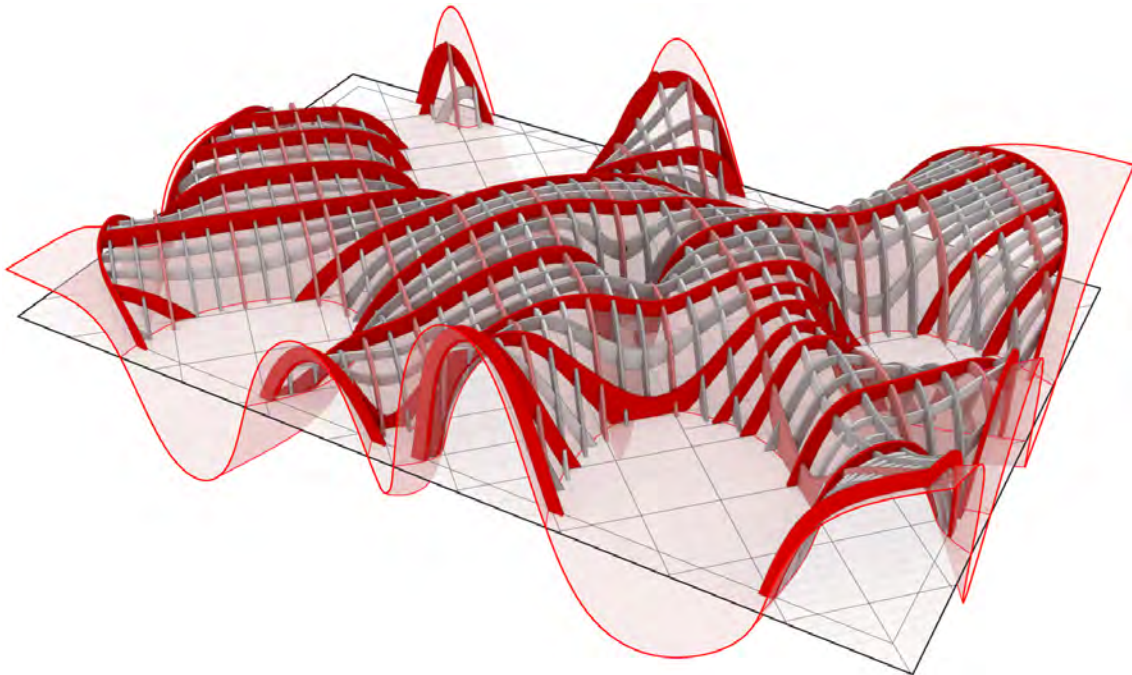
“When satisfied, they exported a table containing the unique name and the type of joint for either end of the segment, which was then imported into the fabrication model by the 3D-planner and the joint types were updated accordingly. Automatically generated 2D-drawings were used for checking and documentation.” (Scheurer et al., 2015)

To enable such process, it was necessary to keep a consistent information flow between the 3D model (Figure 4.7a) from which subsequent 2D information can be extracted, and its corresponding topological map (Figure 4.7b) which served as a managerial, location interface of each joint for the different trades. This topological map usually can be compared to the “building’s skin” which has been completely flattened on the XY plane. It is then much easier for each trade to refer to it and map the project’s complexities.

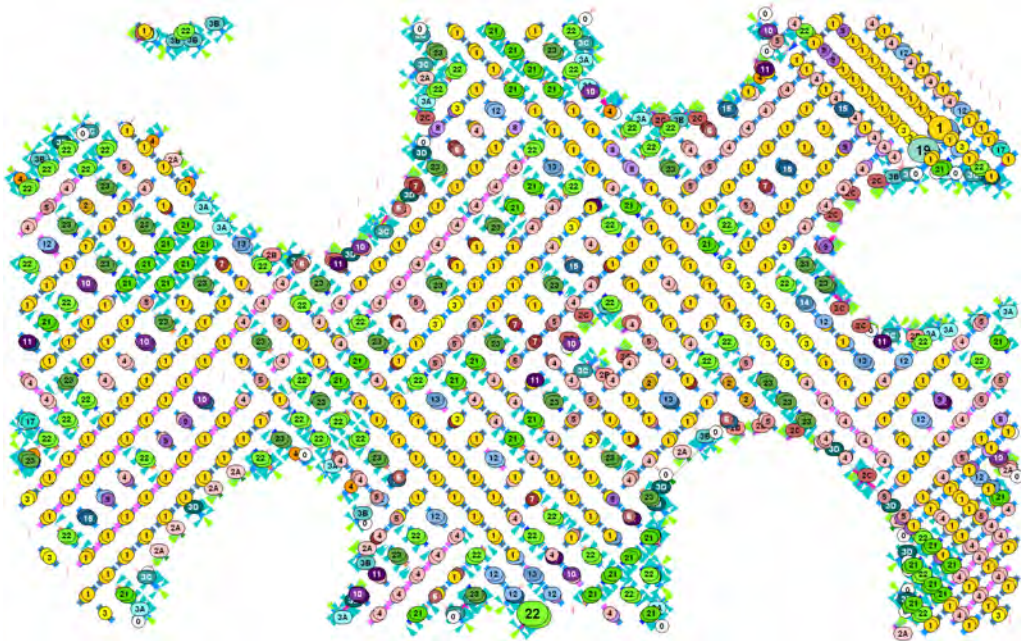
- **The Nine Bridges and the Cambridge Mosque Projects**

Both the Nine Bridges and the Cambridge Mosque projects are topologically very resemblant. Indeed, their corresponding global master surface that serves as a guideline to generate the production data could be subdivided into different sub-modules which are all almost identical, helping therefore during both the conception and fabrication process by reducing the amount of modelling time to deliver the necessary data for production. In such cases, a topological map also eased the understanding of the 3D model. In both projects, a few differences existed between the different sub-modules, requiring local manual modifications (adding and removal of a couple of elements) for each one of them.

Following the initial case studies described above (the Solemar-Therme in Bad Dürkheim and the key projects modelled by Design-to-Production) and findings, the next section will focus on different Multi-Scalar Modelling experiments, from graph modeling techniques to interactive UI prototypes.



(a) 3D-CAD model of the France's Expo 2015 Pavilion displaying the entire structure network made of single curved timber elements. Image courtesy of Design-To-Production.



(b) Two-dimensional mapping of the different types of joints required for structural integrity, communicated by the engineer in charge of the project.

Figure 4.7: 3D-CAD model and two-dimensional map of the different types of joints of the France's Expo 2015 Pavilion.

4.1.2.3 A2 | Topological Mapping: Modelling Lap Joints

Experiment A2 | Topological Mapping: Modelling Lap Joints

Category: design probe

Related research question: “How can the computational designer be able to navigate more easily between different scales during the design process?”

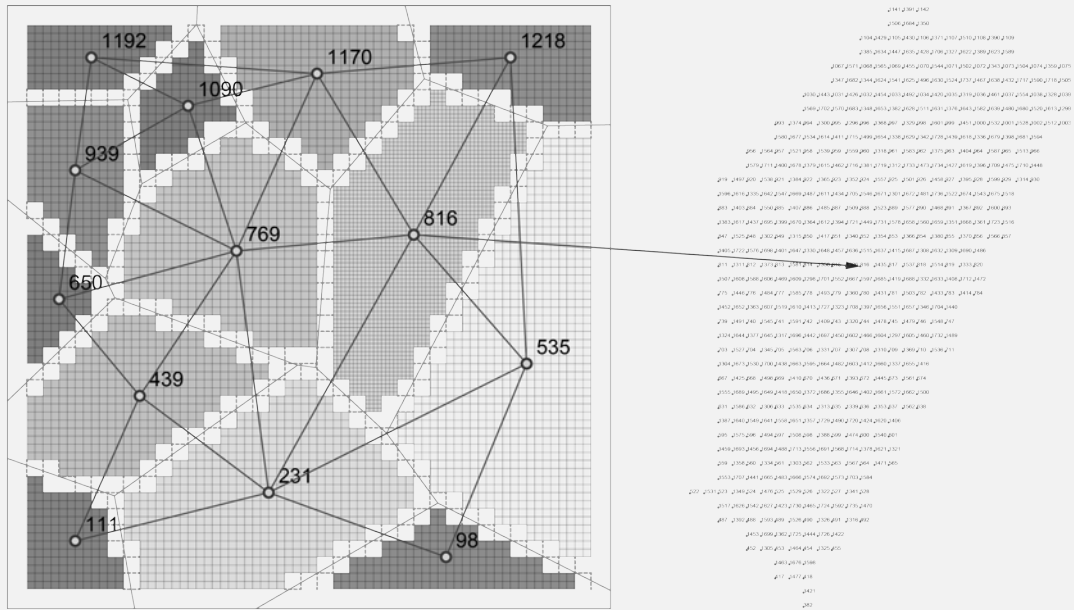


Figure 4.8: Data partitioning using the block model function from the NetworkX python library.

Description

Similarly to the previously described modelling experiment “A1 | *BlankMachine: Propagating the blanks*”, the present modelling experiment acts as a design probe — “a design-led investigation allowing speculative inquiry, theorization and the setting out of design criteria.” (Ramsgaard Thomsen and Tamke, 2009, 3) — and bases itself on existing modelling practices and precedents. The present experiment grounds itself in the above described digital design workflows employed both at Design-to-Production and by Linkwitz and Veenendaal, which tackle the rationalization and modelling process of a free-form timber structure across all levels of resolution, from design to fabrication. Although capable of delivering building data at late stages, these existing modelling practices remain quite laborious as the related inherent discretized design steps hardly communicate between each other. Therefore, the present experiment attempts to provide specific design methodologies enabling the computational designer to link the different scales and resolutions across the whole digital chain. Here, the employed design methodologies ground themselves on existing architectural design research projects — (Nicholas, 2016), (Nicholas et al., 2017) and (Ramsgaard Thomsen et al., 2015) — conducted at the Centre for Information Technology and Architecture and described previously in section 3.1.3. Through the inter-linkage of multiple computational design pipelines, these research projects investigated how one could navigate more efficiently between different scales and resolutions. The modelling processes used within these research projects relied on graph-based models and third-party libraries to pass information between different frameworks and operate targeted queries across the different scales. Similarly, the present experiment implements a graph-based modelling workflow to navigate through multiple levels of resolutions. Furthermore, it challenges the design scale as it tackles the modelling process at

a building scale, contrasting therefore with the above mentioned existing precedents that successfully handled the smaller scale of the demonstrator.

First, initial investigations have been carried out with the **NetworkX** Python library which was previously used and implemented within computational design pipelines by (Ramsgaard Thomsen et al., 2015). This library is described a “*Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.*” It provides to the programmer or computer scientist the necessary tools and algorithms to apply graph theory for the study of the structure and dynamics of infrastructure networks between multiple levels of resolutions. Figure 4.8 investigated this aspect on a very abstract level. A coarsened, low resolution graph contains higher data resolution at each of its nodes, which can be accessed using the **block model partitioning algorithms** from the NetworkX library. Indeed, the latter contains different functions enabling the creation of subgraphs or block-models which keep in memory the location of every single dataset. This presents great potential for enabling direct interaction, feedback and update between scales.

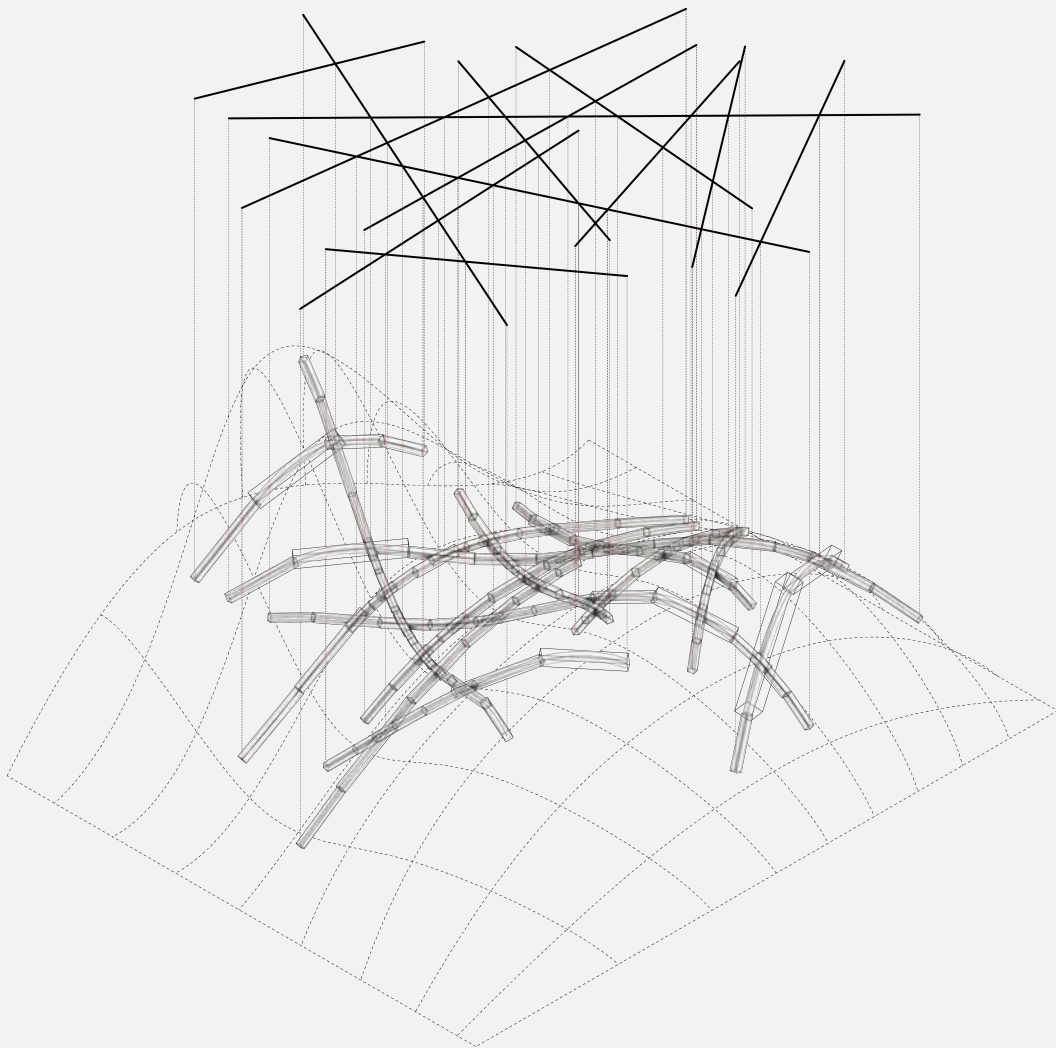


Figure 4.9: Projection-based modelling onto a master surface. Query of different nodes corresponding to particular wooden lap joints within the 3D model.

After identifying the different resolutions that can be found within the design process of a free-form timber structure (as in the above described digital design workflows employed both

at Design-to-Production and by Linkwitz and Veenendaal), further investigations on multiscale modelling techniques have been made using the Python NetworkX library. A modelling workflow has been developed in which the designer is able to manipulate early stage data (the design of a network of intersected lines onto a 2D plane) which directly informs later design stages (here, the projection of those intersected lines onto a 3D free-form surface, their subsequent offsets into three-dimensional elements, and the calculation of their respective lap-joint connections). Whenever the user manipulates the high-level pattern by adding or removing a line from the intersected network, the latter is converted into an undirected graph through the Python NetworkX library which keeps track of the semantic relationship between this graph and the generated 3D geometry. Here, the abstract network acts as a skeleton model that serves as Skeleton Model described in section 2.5.3: *“Skeleton Models represent the minimum amount of information (wireframe elements: points, lines, planes, etc.) to generate more intricate geometrical datasets, and can serve as a point of contact to control two (or multiple) different (sub-)models.”* (Hirz et al., 2013, 303)

The present experiment attempts to enable direct feedback between a 3D model (Figure 4.10) of intertwined timber beams connected through lap joints and a 2D graph (Figure 4.11) mapping information relevant for the further fabrication of the generated free-form timber elements. This topological graph allows the user to get a quick overview of the complexity of the structure: the darker the segments are represented, the more twisted are their corresponding timber elements. The graph also informs both graphically and numerically on the length of each segment: the longer the latter will be in the original 3D model, the longer its corresponding edge will be represented within the graph, along with its numerical value. Finally, the user also has the option to query directly from the graph specific joints and retrieve their corresponding neighboring nodes (Figure 4.11).



Figure 4.10: Generation of networked lap joints.

Results

The different features implemented within the present modelling experiment allowed the user to visualize complexity, search and query the model in a more intuitive manner than through

the more traditional Direct Modelling method (Pena De Leon, 2014). In this degree, the experiment demonstrated – through multiple stress tests against different free-form surfaces – the possibility of leveraging early stage data to improve late stage design decisions. Furthermore, the UX during navigation between the different scales through the design process has been improved, as the computational designer was able to seamlessly pass relevant information for further fabrication from one representation to another.

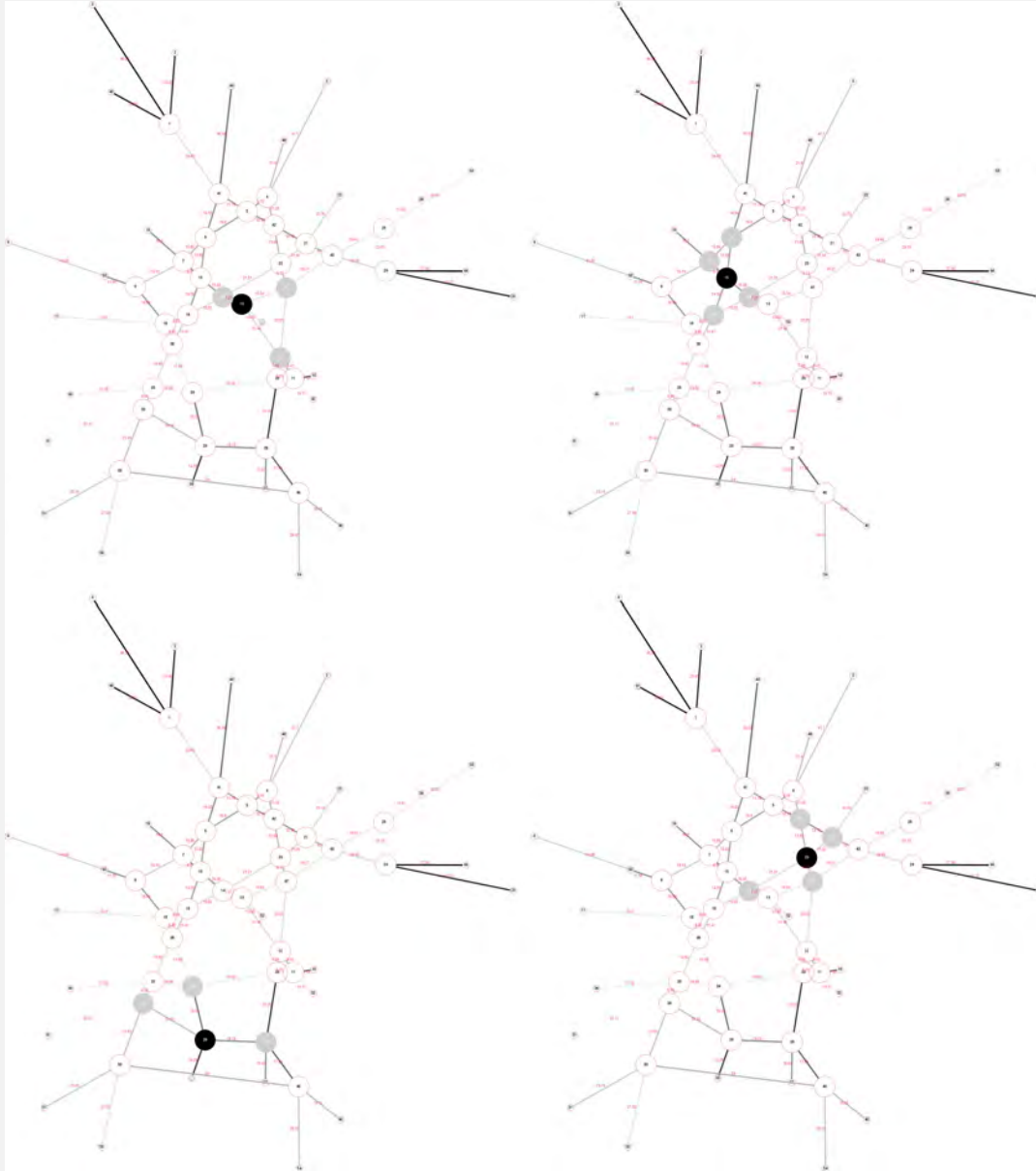


Figure 4.11: Query of different nodes corresponding to particular wooden lap joints within the 3D model.

Limitations and future work

Although enabling seamless navigation between multiple scales, the present experiment restraints itself to the design space that has been proposed here to test the graph modelling workflow with NetworkX. Indeed, the present computational pipeline solved a very particular design problem by allowing only two beams to intersect into one particular lap joint type. Therefore, the next modelling experiments will act as digital prototypes and will attempt to demonstrate that the same graph modelling methodologies employed within the present

experiment can be further deployed and stress tested against more intricate joint topologies and modelling frameworks. In this aspect, the initial findings gathered so far will be helpful to carry out the next digital experiments that introduce more geometrical and topological complexities within the design process.

4.1.3 A3 | Topological Mapping: Surface/Projection-based modelling

Experiment A3 | Topological Mapping: Surface/Projection-based modelling Category: prototype



[Video link to the experiment](#)

Related research question: *“How can the computational designer be able to navigate more easily between different scales during the design process?”*

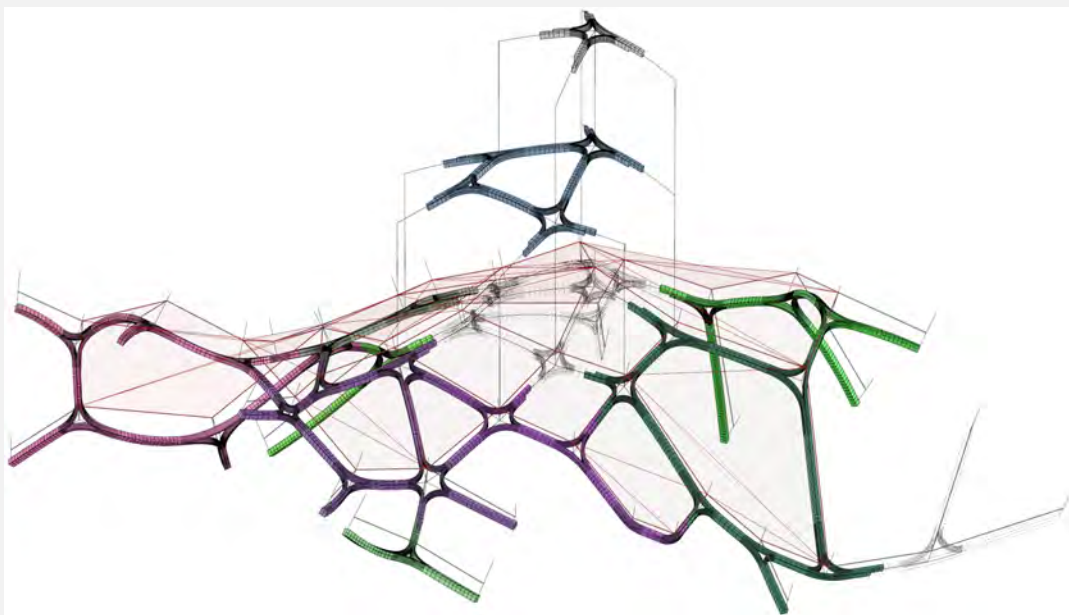


Figure 4.12: Interactive modelling of a branching structure propagating onto a master surface.

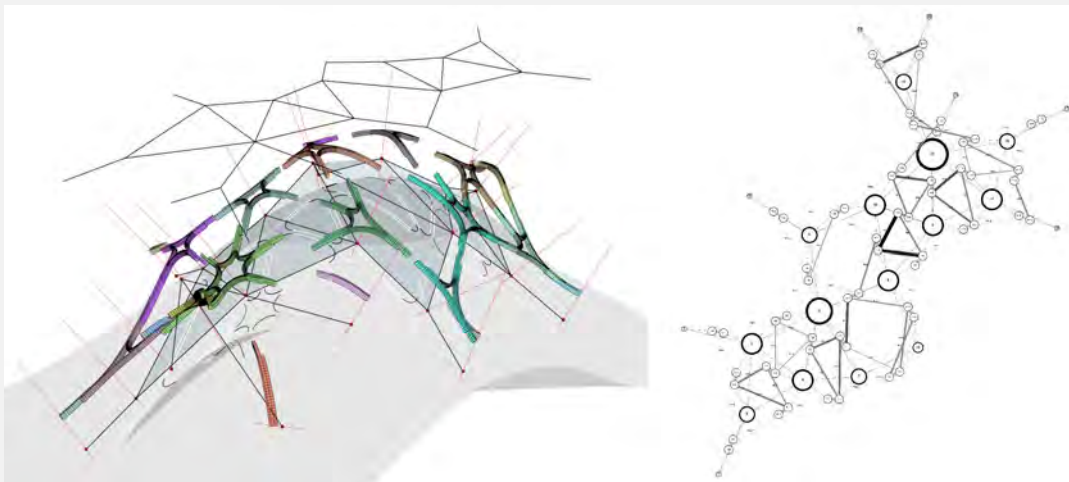


Figure 4.13: Relationship between the 3D model and its corresponding assembly graph.

Description

The present experiment continues to follow the accumulative research approach defined by (Krogh et al., 2015) and described previously in section 3.4. It takes into account the knowledge gained in the previous experiment “A2 | Topological Mapping: Modelling Lap Joints”, within which graph modelling strategies have been investigated to seamlessly pass information across multiple scales. The previous experiment only acted as a design probe and its computational design pipeline allowed only two beams to cross into one particular lap joint typology. In addition to the application of the same graph modelling methodologies developed and investigated in the previous experiment, the present experiment attempts to embed more complexities and levels of resolutions within the design process. Furthermore, it acts as a prototype, defined previously as “[...] a materially-led investigation allowing exploratory testing, of craft and material behaviour. The prototype answers and develops the design criteria of the design probe.” (Ramsgaard Thomsen and Tamke, 2009, 3)

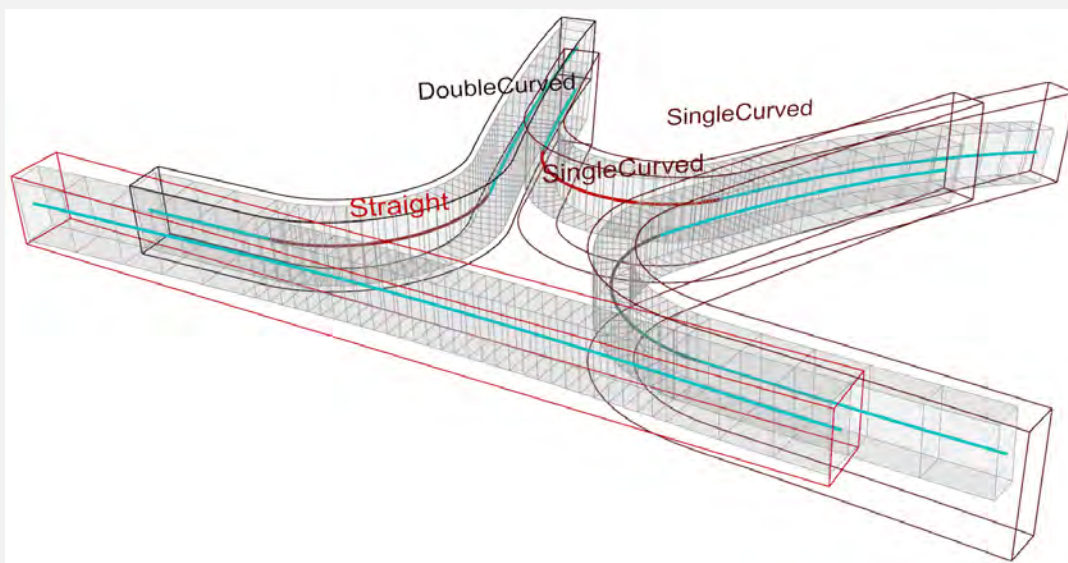


Figure 4.14: The modelling of a branching joint and the corresponding blank type for each of its glue-laminated timber beam elements.

The present experiment has been developed in order to take into account a specific type of joint that could branch and therefore adapt to different user-specified valence conditions. Contrary to traditional joint techniques where the node complexity increases by the number of elements connected to it, the reciprocal segmentation location has been displaced at the center of each edge. From this strategy results a specific type of node (Figure 4.14) from which all connected wooden members bifurcate along a specified radius in order to meet their respective neighbors (n). Thus, each glue-laminated timber beam splits and bifurcates in n directions.

Similarly to the previous experiment, the abstract network is here projected onto the master surface and generates information required for further fabrication. Maximum bending radii, segmentation strategies (per node or per group of nodes), full three-dimensional objects and glass panels could be easily generated, accessed and checked by the user by analyzing the corresponding two-dimensional graph, or Skeleton Model (Figure 4.13, right). Using cycle basis algorithms (Kavitha et al., 2009) from the Python NetworkX library, it is possible to detect the closed polygons which are situated within the graph. The expert user is therefore able to partition the abstract network into different subgraph (or subsets) that can be used for different purposes, from rationalizing facade panels to the definition of prefabricated modules to improve on-site fabrication strategies:

- **Glass panel generation**

Each closed polygon of the graph is analyzed and treated as a single element of the cladding system. Glass manufacturing techniques present various typologies: planar, cold bent, single curved and double curved. Depending on the planarity, the size of the polygon and its number of edges, different subdivision strategies enable the generation of the appropriate typology.

- **Prefabrication module management**

Each polygon can also be interpreted as prefabricated module of aggregated timber beam elements. A collection of multiple polygons can generate larger modules. Transportation and on-site assembly constraints would directly define the maximum size allowed for each module.

A particular node from the 3D model could be extracted and be further analyzed to generate the blank geometries of its corresponding members (Figure 4.14). Based on the same techniques and modelling methodologies described in section 4.1.1, it was possible to find the best fit for the member's blank type (straight, single curved or doubly curved). Such analysis could happen within a submodel that was segregated from the parent model. The latter focused on the global structure in its whole but without focusing particularly on fabrication data modelling. Here, the separation of concerns is therefore taken into account.

As described in *"Multi-Scalar Modelling for Free-form Timber Structures"* (Poinet et al., 2016) and illustrated in figure 3.18b, it is also possible to overlay Multi-Scalar Simulation frameworks on top of the modelling frameworks. In this modelling case study, the radius of each wooden member was crucial for further fabrication. Within the initial abstract network, the radius of each bifurcation is calculated by generating the best fit between the largest possible fillet allowed at the angle between two neighboring beams and the minimum bending radius defined by the material behavior of the chosen wooden species. In some cases, it might happen that the bending radius is too small to enable the glue-laminated timber element to take shape. Therefore, a dynamic relaxation has been implemented with K2^a in order to take into account a specified minimum bending radius across all the timber elements. Once relaxed, the graph is able to negotiate all angles so that the bending radius of all members is maximized throughout the overall network. Figure 4.15 displays the relationship between the original free-form timber structure (left) and its corresponding optimized version after relaxation (right). The angles displayed in red are the ones whose values are smaller than the angle specified by the computational designer, and the green ones are the ones whose values are larger than the same specified angle value. It can be observed that the number of red angles – the nodes with a level of curvature too high, causing fabrication issues – are diminishing after running the geometrical optimization through dynamic relaxation.

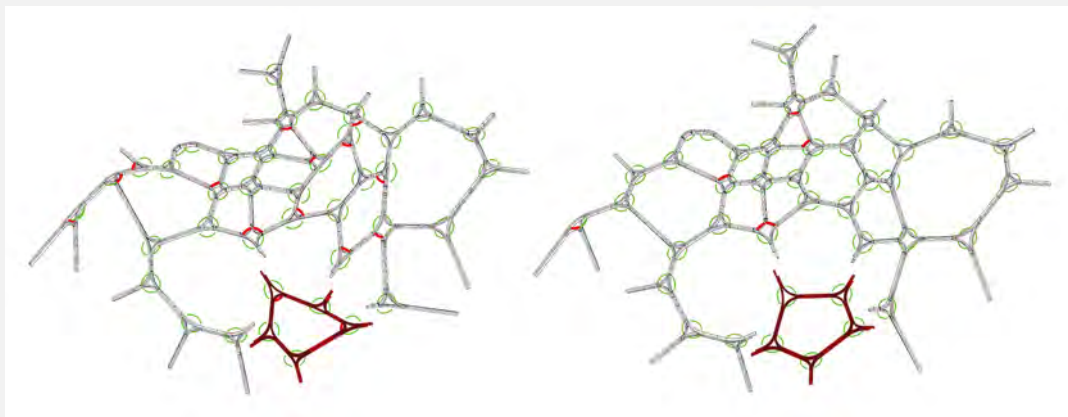


Figure 4.15: Geometrical optimization of the overall network through the maximization of the radius of each glue-laminated timber element (left: original network, right: optimized network).

Equivalently to the first digital experiment described in the previous subsection, the 3D model is keeping its relation to a two-dimensional graph, before and after the dynamic relaxation (Figure 4.16). The graph was generated with both the Python library NetworkX and the Graph Visualization Software GraphViz, taking directly as inputs the data coming from the 3D model itself. GraphViz is defined as an “*open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks.*” (<https://www.graphviz.org/>). Here, the generated graph displays visual information about each member: the geometrical complexity of each beam is represented through a gray gradient and the size of each element informs the graph’s layout and size. Furthermore, the number of neighbors meeting at each joint is also graphically represented by the scale of the node’s radius within the graph. Finally, the graph informs on the reciprocity of the overall structure and displays at the center of each edge which wooden member overlaps with its corresponding reciprocal element. Thanks to these different display features, the computational designer is able to better keep track of the evolution of the 3D model and to maintain a good general understanding of the reciprocal structure’s complexity and thus at early design stages.

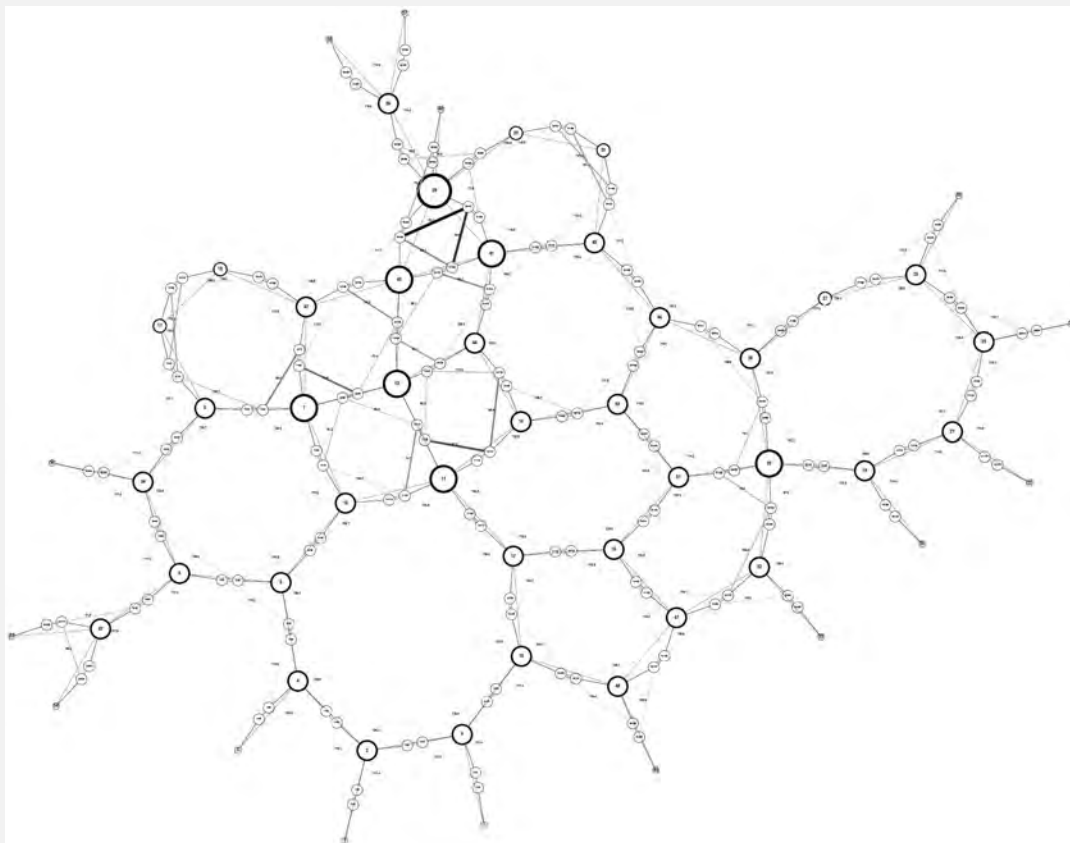


Figure 4.16: Two-dimensional graph corresponding to the geometrical 3D model, before and after relaxation.

Results

Through multiple stress tests against different source models, free-form surfaces and complex joint types, the present experiment also demonstrated both the possibility of leveraging early stage data to improve late stage design decisions, and a frictionless user experience during the design process, while navigating between the different scales. The computational designer was able to seamlessly pass relevant information for further fabrication from one representation to another.

Limitations and future work

Although digital flexibility and the understanding of geometrical complexity have been improved through data visualization tools and graph analyses at early design stages, the digital pipelines still depend very much from each other. Indeed, every computational pipeline developed in this experiment depends from a single reference system – a simple [NURBS](#) master surface – which controls the overall digital workflow. The next experiment attempts to overcome such issue by multiplying the reference systems and enable the user to set multiple master surfaces as design drivers. In such case, the modelling of each node is processed according to its local situation.

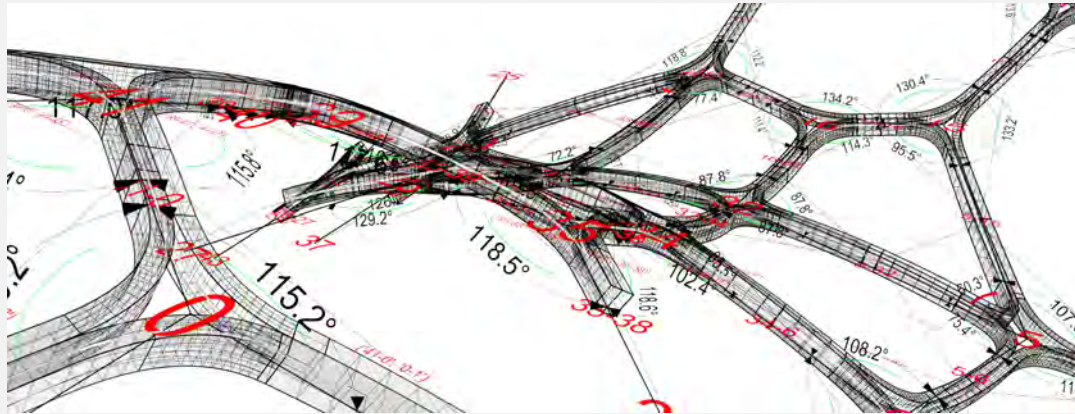


Figure 4.17: Overlay of different data sets coming both from the global abstract network, corresponding graph and nodes typologies.

^aK2 (or Kangaroo 2) is a popular live physics engine for interactive simulation, form-finding, optimization and constraint solving developed by Daniel Piker. It consists of a solver library and a set of Grasshopper components.

4.1.4 A4 | Topological Mapping: Spatial Branching

Experiment A4 | Topological Mapping: Spatial Branching Category: prototype



Video link to the experiment

Related research question: “How can the computational designer be able to navigate more easily between different scales during the design process?”

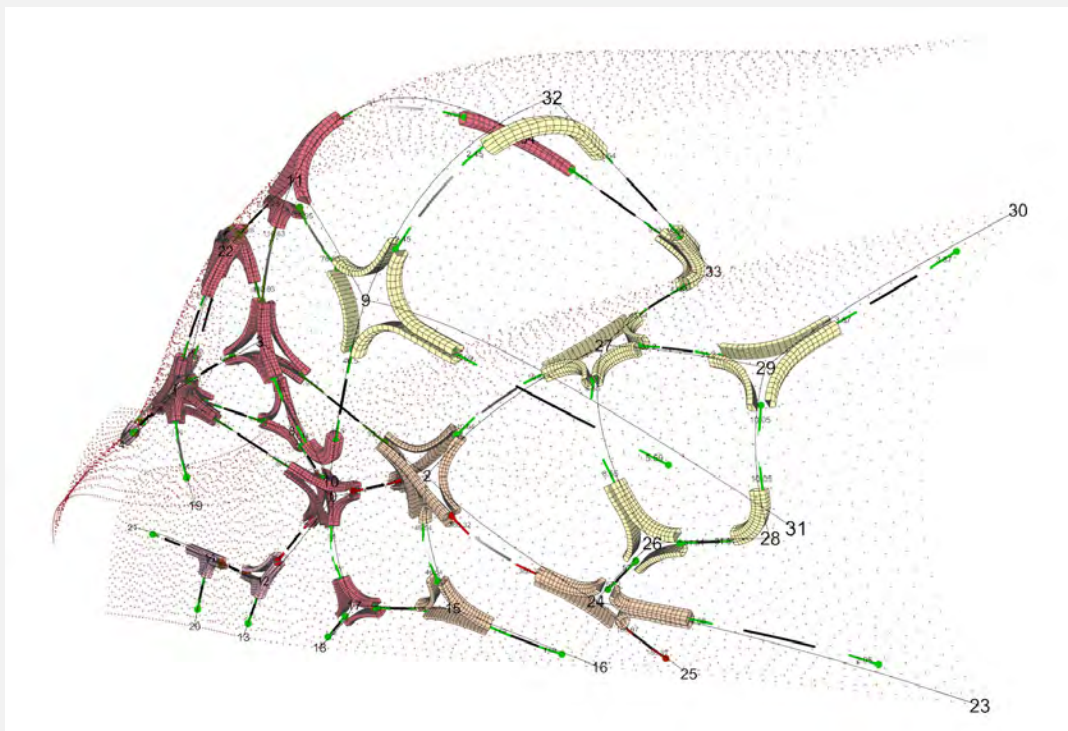


Figure 4.18: Interactive modelling of a branching structure propagating spatially.

Description

The present modelling experiment (illustrated in figures 4.18, 4.18 and 4.21) continues to follow the *accumulative* research approach described earlier in section 3.4 and takes into account the knowledge gained in the previous experiments “A2 | Topological Mapping: Modelling Lap Joints” and “A3 | Topological Mapping: Surface/Projection-based modelling”. Following experiment “A3 | Topological Mapping: Surface/Projection-based modelling”, in which the design space was restricted to a single master surface, further complexity has been introduced through the implementation of multiple master surfaces that can locally guide the geometrical orientation of each free-form timber node element. In this case, the computational complexity of each node is treated separately 4.19, and depends on corresponding local properties (e.g. the number of valence, the existence – or not – of a specific master surface). Those independent local parametric systems are added on top of an abstract network (or Skeleton Model as in (Hirz et al., 2013)). This abstract network can be treated as global Undirected Graph where each vertex can be analyzed independently in order to retrieve

specific information, such as: valency, joint typology, specified attributes, etc. The Undirected Graph maps abstractly the different types of node whose full geometrical description required for further fabrication can be generated locally and independently from any sort of global computational rule, and thus through a DAG. The possibility of triggering a local DAG pipeline through selection of a specific node within a global Undirected Graph is enabling the user to treat each object as a local exception. Therefore, the classical full Directed Acyclic Graph workflow is replaced by an hybrid workflow strategy within which information can be beforehand globally filtered and analyzed through an Undirected Graph before computation of precise geometries through local DAG pipelines.

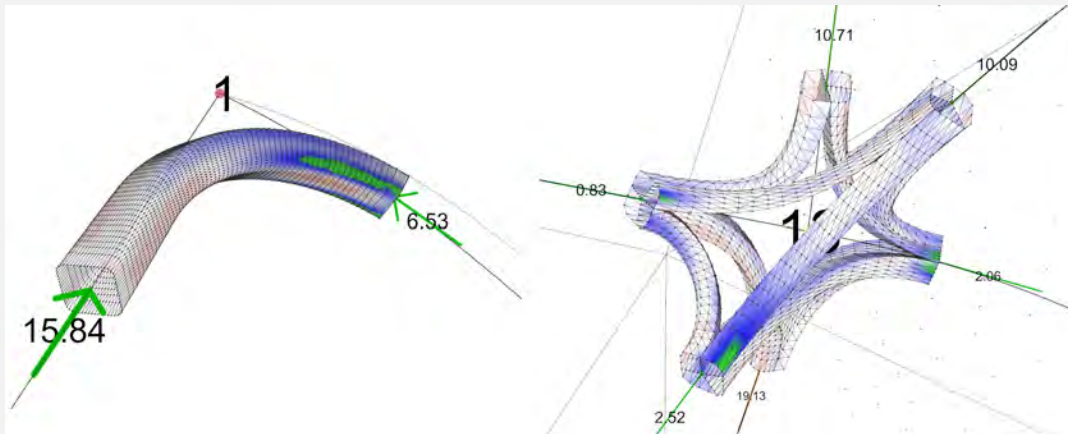


Figure 4.19: Nodes can be selected, extracted from the main database and their resolution can be increased in order to perform Finite Element Analysis.

Similarly to the modelling experiment “A3 / Topological Mapping: Surface/Projection-based modelling”, it is also possible to overlay Multi-Scalar Simulation frameworks on top of the modelling frameworks. Here, the user is able to select locally each node and run a local structural analysis to verify the stability of a particular wooden branching joint. The input forces are extracted from a global structural analysis that has been run on the global abstract network, acting here as a Skeleton Model. It is also possible to subdivide the whole structure through a K-Means algorithm and retrieve specific patches of nodes.

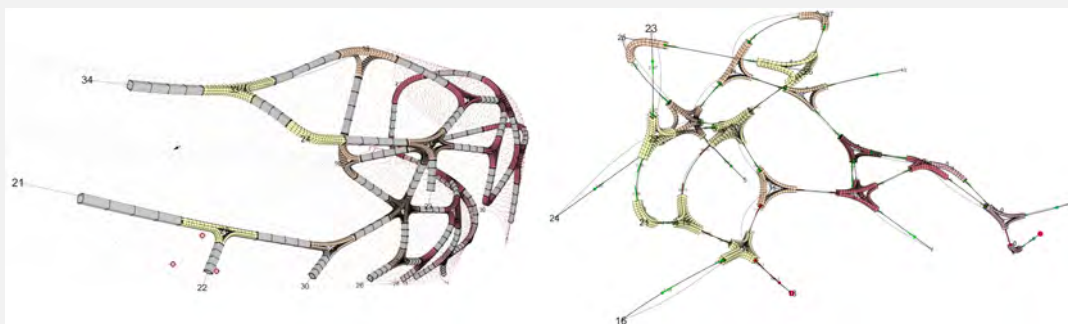


Figure 4.20: From the global model, the user is able to generate different sub-models, focusing on the beam and/or the node design components.

Results, limitations and future work

This modelling experiment was very useful to investigate practically the benefits of the Separation of Concerns (Dijkstra, 1982) within computational design pipelines and workflows. Although some of the methods applied here proved to be useful to improve data access, search and UX during the design process, it was quite challenging to craft every single design

exception in the design process, so that the node typologies could adapt to any particular situation (tangency to a master surface, high valence number, etc.). This particular problem has already been highlighted in section 2.2.3: finding the right balance between the integration of all exceptions in the design process and direct manual intervention is hard to find, as the former suppresses anomalies but keeps the data volatile and dependent on the input ([Scheurer and Stehling, 2011](#)).

Future work

The Multi-Scalar Modelling methods developed so far during the previous experiments have been tested and deployed through multiple external research activities, such as: three different workshops (both from the organizer and attendee perspective), the Future Wood seminar within the InnoChain training network and an architectural design competition. Those different research activities described in the next section will help to better assess the potential and limits of the carried modelling experiments described so far.

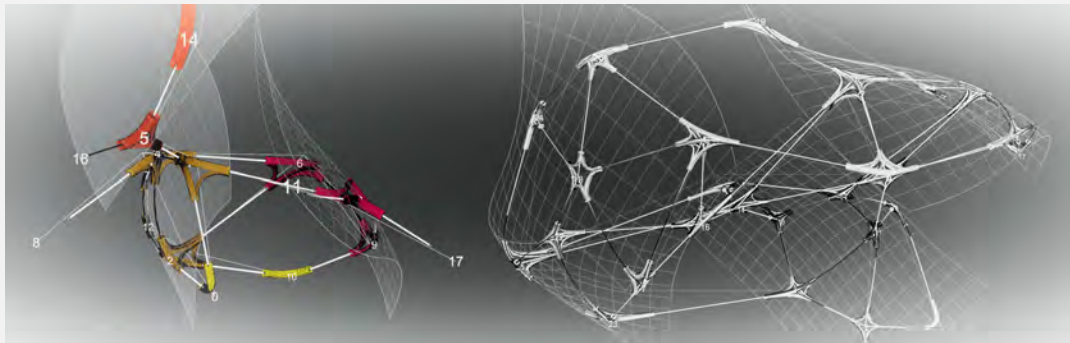


Figure 4.21: Interactive modelling of a branching structure propagating spatially.

4.2 Stress testing modelling-based strategies

In parallel to the modelling experiments described so far in the previous section, the author taught and participated in four different workshops that helped him to contextualize and test the knowledge and findings gathered within the precedent series of experiments, following here the *comparative* research approach, branching therefore from the *accumulative* research approach developed so far.

Teaching is used here as a mean of consolidating, formalizing and disseminating the PhD research. Teaching one- to two-week workshops for master students at KADK creates deadlines and a requirement to clean up the research into a format and scope that is suitable for the workshop topic and length. This helps the overall PhD research by creating checkpoint where development is temporarily stopped for re-structuring and tidying the design methodologies investigated so far. In the long run, this creates a tidier, more accessible knowledge base and a clear and documented research path, as well as a method for disseminating the research throughout the school and other programmes.

The present section attempts to apply and stress test the different modelling-based methodologies developed so far throughout the previously described modelling-based experiments, through direct physical/digital prototyping and workshops: “WS1 / B-MADE Workshop” (subsection 4.2.1), “WS2 / LogJam! Workshop” (subsection 4.2.2) and “WS3 / Exquisite-Timber-Corpse Workshop” (subsection 4.2.3). Furthermore, the Multi-Scalar Modelling framework has been stress tested through the Future Wood seminar organized in the context of InnoChain: “WS4 / Future Wood” (subsection 4.2.4). Finally, the Multi-Scalar Modelling methodologies have also been deployed and stress tested through the Tallinn Architecture Biennale (TAB) Installation Competition: “A5 / The TAB Installation Competition” (subsection 4.2.5). During those different research activities surfaced both potentials and limitations of the so-far developed Multi-Scalar Modelling techniques. These pros and cons will be highlighted in the next subsections for each one of the research activities.

4.2.1 WS1 | B-MADE Workshop: Fabrication Constraints and Assembly Logistics

Experiment WS1 | B-MADE Workshop: Fabrication Constraints and Assembly Logistics Category: prototype



Video link to the experiment

Related research question: “How can a Multi-Scalar Modelling framework allow the designer to work across different scales in order to take into account multiple constraints related to material, fabrication and structural performances during both early design and late stages?”

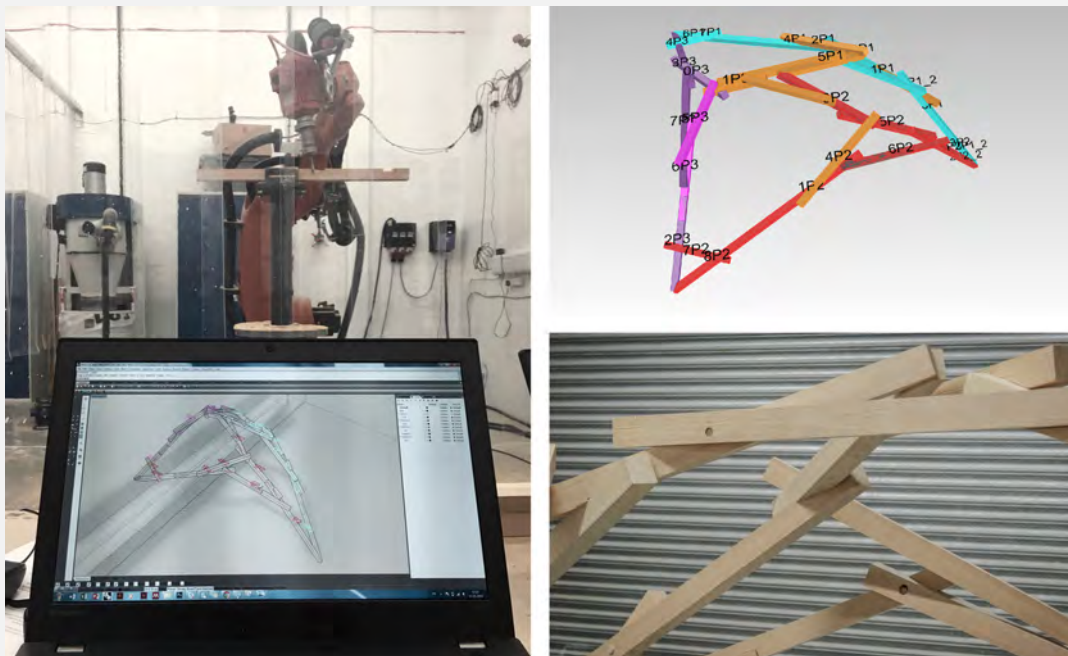


Figure 4.22: The design-to-fabrication workflow of the Sealion sculpture: robotic machining of each bespoke member (left), design model and assembly logistics (top right), physical demonstrator (bottom right).

Description

The present modelling experiment is conducted through the *comparative* research approach described in section 3.4. It takes as main point of comparison the Multi-Scalar Modelling methodologies developed in the first series of modelling of experiments described in the previous section by applying and stress testing them through the development of the present experiment.

The B-MADE workshop seminar was organized in the context of the InnoChain training program for all the Early Stage Researchers. It allowed the workshop participants to engage with robotic fabrication and wooden manufacturing techniques. The purpose of the course was to fabricate a wooden structure which would make full use of the robotic workflow and code library Robots^a developed by the workshop tutors Vincente Soler and Inigo Dodd. The tools developed for the workshop integrated lap joint detection and pocket path generation

that could be read by the robot for further fabrication. The workshop participants were then asked to design the structure by taken into consideration low level fabrication constraints. The author participated in the workshop along with Tom Svilans (ESR 2) and Kasper Ax (ESR 10). The design process was split into three different tasks – from low to high level – shared among the participants: global design, fabrication data model (machine-ready deliverables) and robotic fabrication (Figure 4.22). The global design model was constantly refined based on the fabrication data model to test it whether or not each individual bespoke wooden member could be further robotically fabricated. The B-MADE workshop seminar was therefore a very good opportunity to stress test the Multi-Scalar Modelling methodologies and graph modelling techniques investigated and developed so far, in order to facilitate the passing of information between the different levels of resolution throughout the whole design-to-fabrication process of a collaborative practice. Similarly to the digital experiments described in the previous section, graph modelling methods were used in order to design and keep track of each single wooden piece during the fabrication process. The different elements were finally assembled together with lap joints, butt joints and tenons, forming the overall structure, named “Sealion” (Figure 4.24). Figure 4.23 displays the overlay between the abstract network (or Skeleton Model) the fully generated joint geometries.

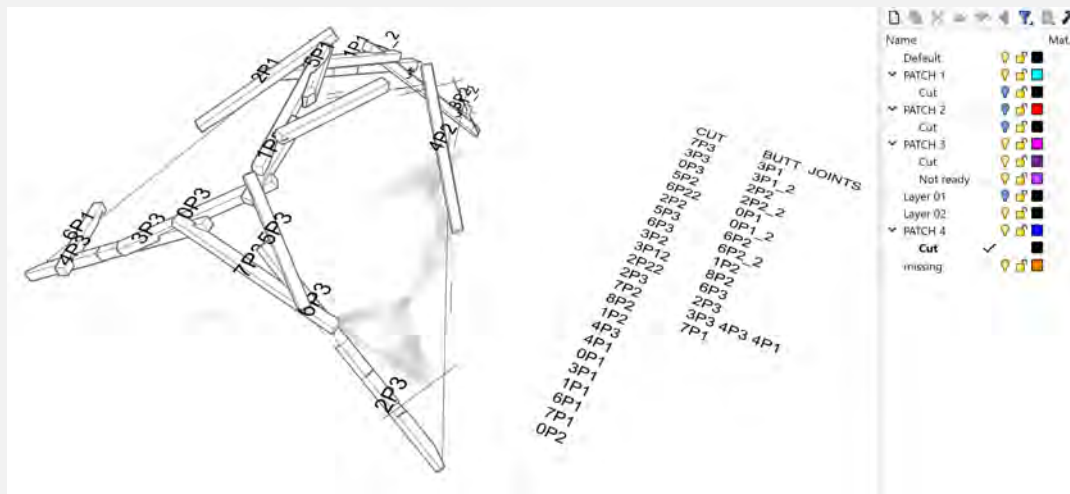


Figure 4.23: Overlay between the abstract network and the full 3D representation of each wooden member.

Results and limitations

Although the graph modelling techniques were useful at early stages to evaluate the abstract network and understand the joint complexities, it was not sustainable to keep a continuous flow of information between different models, levels and resolution until late stages. Indeed, the digital models were corrected back and forth multiple times between the team members until it was possible to generate machine code for every joint within the entire structure. The short workshop time-frame did not allow to invest time in elaborating a custom and continuous graph modelling pipeline through all the design phases of the project. Indeed, as already highlighted in section 3.2 the design workflows and interoperability bridges developed in the context of academic research projects are often custom-made, conceptualized and designed very quickly – to meet short deadlines – and therefore rarely re-usable for other research investigations that deal with different sets of materials and processes. Instead, many manual interventions were necessary, leading to the duplication of multiple residual models which only served to test whether or not fabrication was possible for specific joints. The Rhino LayerTable was also extensively used to classify the generated joints into production batches and track them during assembly (see Figure 4.23). These issues came at very late stages in the design process, during which many small but crucial adjustments were necessary to design and fabricate the entire timber structure.

Although the workshop team members managed to design and fabricate the *Sealion* structure, communication bottlenecks have been observed and were in fact one of the main issues within the design process. Therefore, instead of focusing exclusively on the investigation and development of graph modelling techniques allowing the passing of information between different levels of resolution, Multi-Scalar Modelling for building design should also focus on solving interoperability and data classification related issues.



Figure 4.24: Final assembly and physical model.

^a<https://github.com/visose/Robots/wiki>

4.2.2 WS2 | LogJam! Workshop: Structural Optimizations and Performance

Experiment WS2 | LogJam! Workshop: Structural Optimizations and Performance Category: prototype



Video link to [the workshop's overview \(left\)](#) and to [the modelling experiment \(right\)](#)

Related research question: “How can a Multi-Scalar Modelling framework allow the designer to work across different scales in order to take into account multiple constraints related to material, fabrication and structural performances during both early design and late stages?”

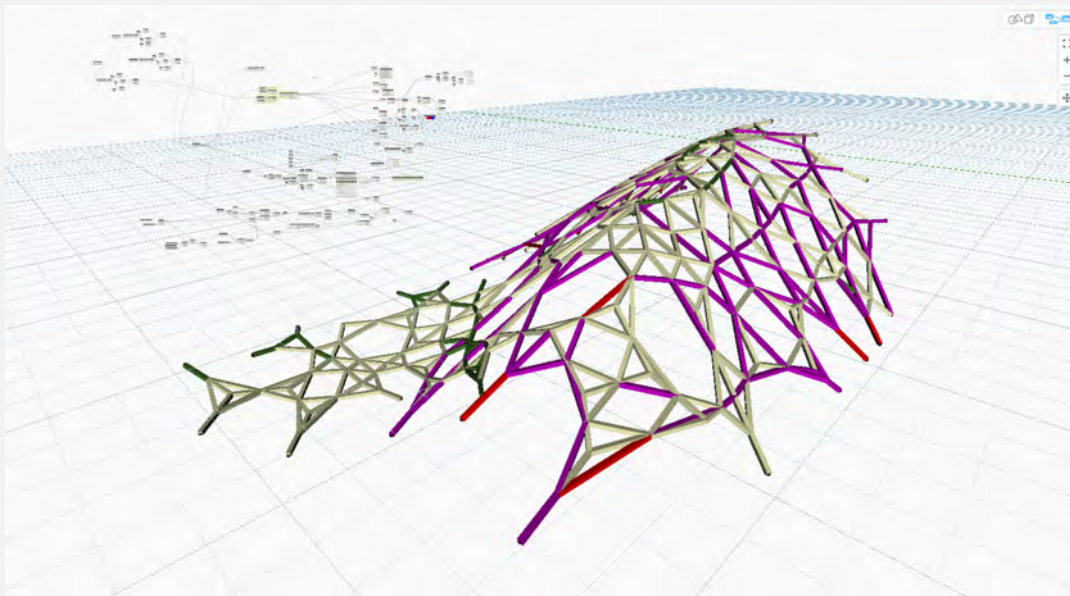


Figure 4.25: Parametric model of a spatial design integrating structural feedback.

Description

The present modelling experiment continues to follow the *comparative* research approach described in section 3.4. It takes as main point of comparison the Multi-Scalar Modelling methodologies developed in the first series of modelling of experiments described in the previous section by applying and stress testing them through the development of the present experiment.

The LogJam! workshop organized by Gustav Fagerström, Nick Cote and Michael Kirschner during the Smart Geometry workshop in 2016 proposed an exploration of inter-species optimization in construction of spatial structures through the use of different species of wood such as oak, pine, birch and spruce. Each of these species has different material behaviour and structural capacity. Through the design, optimization, analysis, assembly, and evaluation of a prototypical multi-species hybrid spatial structure, the workshop explored the role of computational design in wood composites through a the study of gradient materials in buildings.

This workshop was an opportunity to apply simulation frameworks onto the modelling pipelines by taking structural analysis workflows into account. The Autodesk Robot Analysis

package was used along with the *DynamoToro* plug-in to perform both parametric modelling and structural optimizations. After performing a structural optimization, a color gradient was mapped through the overall abstract network. Each generated color corresponded to a specific wooden species. Multiple model variations and design explorations have been developed in Dynamo in order to generate different kinds of spatial enclosures (Figure 4.25).

Results

The workshop team members finally agreed on a common demonstrator which could be fabricated during the four workshop days. This demonstrator is composed of three leg trusses that meet at six meters above the ground (Figure 4.26, right). Structural behaviour has been simulated (Figure 4.26, top left) before informing the fabrication data model (Figure 4.26, bottom left) within which the different types of wooden species were distributed and assigned to each member. Here, the setup of Skeleton Models and interfaces between the different kinds of models (structural, fabrication and simulation) was crucial in the establishment efficiency and fluidity of the design process. Unlike the previous modelling experiment *WS1 / B-MADE Workshop: Fabrication Constraints and Assembly Logistics*, it was possible to set up a continuous Multi-Scalar Modelling pipeline between the different sub-models as the joint typologies were relatively simple and did not necessitate any sophisticated robotic fabrication.

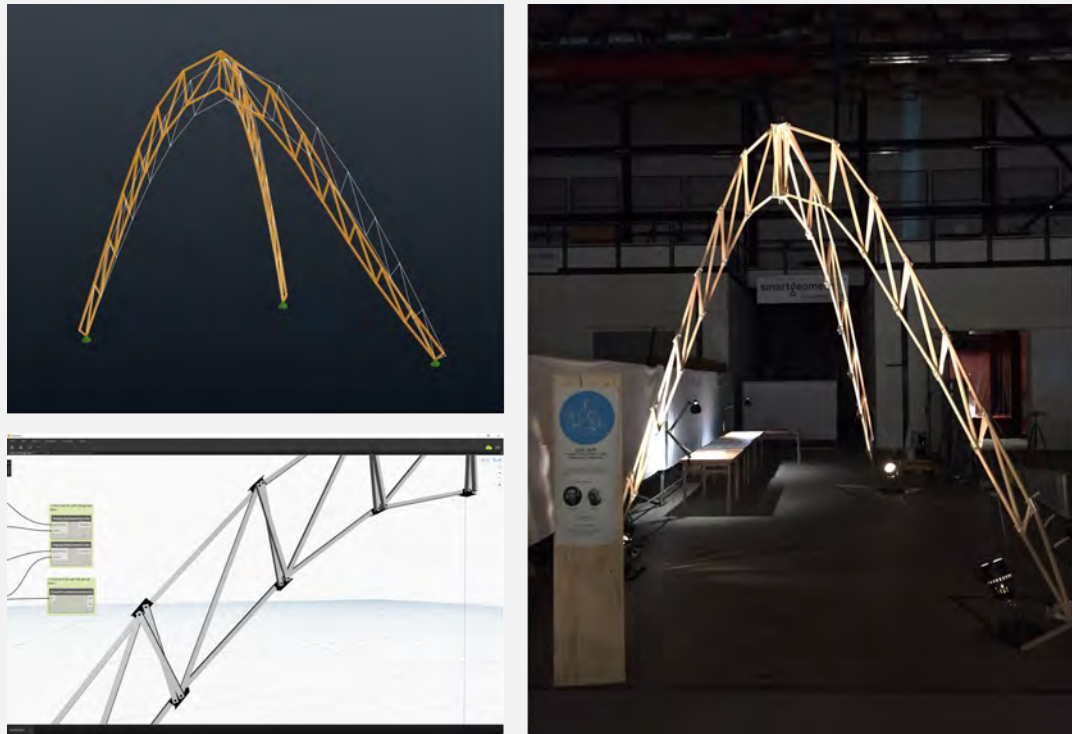


Figure 4.26: Design-to-fabrication workflow of the final demonstrator (bottom).

4.2.3 WS3 | Exquisite-Timber-Corpse Workshop: Collaborative Practice

Experiment WS3 | Exquisite-Timber-Corpse Workshop: Collaborative Practice Category: design probe

Related research question: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

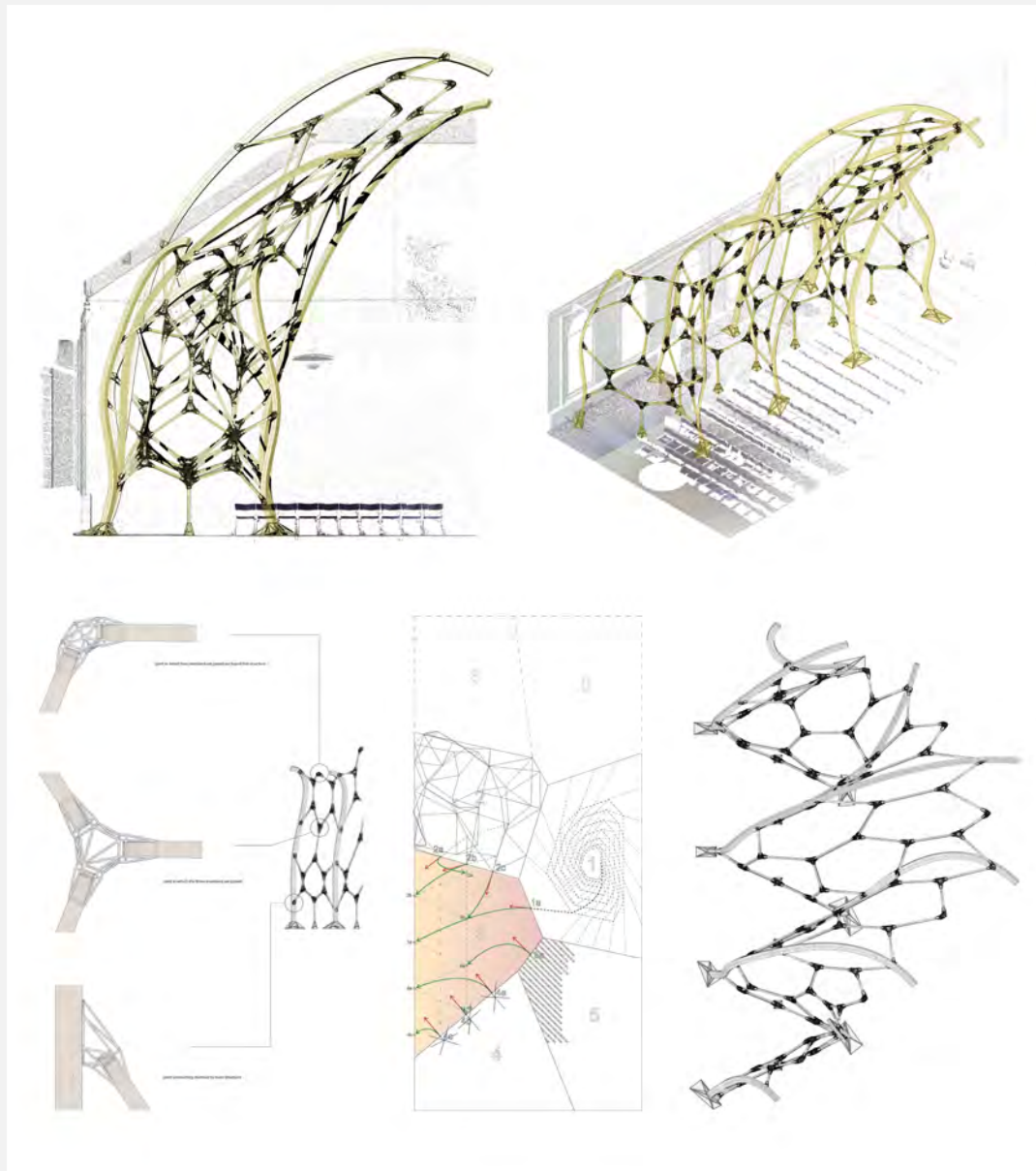


Figure 4.27: The speculative KADK canteen's roof structure developed by Emil Buchwald, Stian Vestly Holte, Daoyuan Zhu and Brian Cheung. From left-to-right and top-to-bottom: section (south), global axonometry, joint detailing, contextualization with the other design groups, top view.

Description

The present modelling experiment continues to follow the *comparative* research approach

described in section 3.4. It takes as main point of comparison the Multi-Scalar Modelling methodologies developed in the first series of modelling of experiments described in the previous section by applying and stress testing them through the development of the present experiment.

Based on the first material experiments and prototypes investigated during the first CITAstudio wood-workshop led by the author and Tom Svilans in September 2016, the workshop *“Exquisite-Timber-Corpse”* explored and speculated on various and extended design possibilities using glue/cross-laminated timber elements and different joinery techniques. The aim of the workshop is to understand and develop specific Multi-Scalar Modelling techniques through graph theory and analytics (using the Python library NetworkX) in order to hierarchize and cluster information through a design framework (or pipeline) for extracting three-dimensional information necessary for fabrication purposes. Potential feedback through the overall network will be introduced through geometrical and structural optimization strategies. The workshop tries to challenge the current conceptions of integrative digital workflows within computational design processes by hybridizing the models in order to introduce more exceptions onto the design landscape. Instead of generating all geometries based on one governing rule, the different objects are treated separately and depend on their local environment (similarly to the modelling experiment *“A4 | Topological Mapping: Spatial Branching”*). Thus, the present overall design framework should not be constrained by only one particular type of joint but by a multitude of design options that can propagate across the network, from surface-based geometries to complete free-form design species.

The main purpose of this workshop is to (re-)create the KADK canteen roof structure whose global design will result from the aggregation of many different local design frameworks developed by multiple student groups. Based on the original idea of the Exquisite Corpse (a participative design method by which a collection of words or images is collectively assembled), each group will elaborate their unique design-isolated environment that needs to connect with the respective neighboring groups, and thus by collaborating and negotiating the connected boundary conditions. Each group is asked to demonstrate the flexibility and design potential of their local parametric workflow by testing it against the global and collaborative design space. The latter should replace the truss supporting the KADK canteen's roof. Therefore, the global structural performance and the many connection details within and between the different groups need to be thought thoroughly.

Results and limitations

From the global design manipulation to the nodes detailing, global contextualization and neighboring negotiation (Figure 4.27), the teams managed to make a coherent design proposal and address all aspects described in the workshop's brief. Furthermore, the multi-scalar modelling pipelines and graph modelling methodologies developed in *“A2 | Topological Mapping: Modelling Lap Joints”*, *“A3 | Topological Mapping: Surface/Projection-based modelling”* and *“A4 | Topological Mapping: Spatial Branching”* were successfully implemented. However, the majority of the workshop participants had some difficulties in fully collaborating across external teams and to adapt their respective local boundary conditions to the ones from the neighboring teams. Instead, the focus was primarily placed locally and individually on each structural design. It became therefore very difficult at late stages to adapt each model to match respectively the neighboring conditions without affecting the initial design intent. As in *“WS1 | B-MADE Workshop: Fabrication Constraints and Assembly Logistics”*, communication bottlenecks have been observed at late stages during which many manual interventions were necessary to handle late design changes. This led to the duplication of multiple residual models which only served for adapting the external boundary conditions to the existing contexts. It has been discovered that communicating and passing information across external models and teams was a crucial point to in the design process.

4.2.4 WS4 | Future Wood Seminar: Speculating Design Spaces

Experiment WS4 | Future Wood Seminar: Speculating Design Spaces Category: design probe

Related research question: *“How can a Multi-Scalar Modelling framework allow the designer to work across different scales?”*

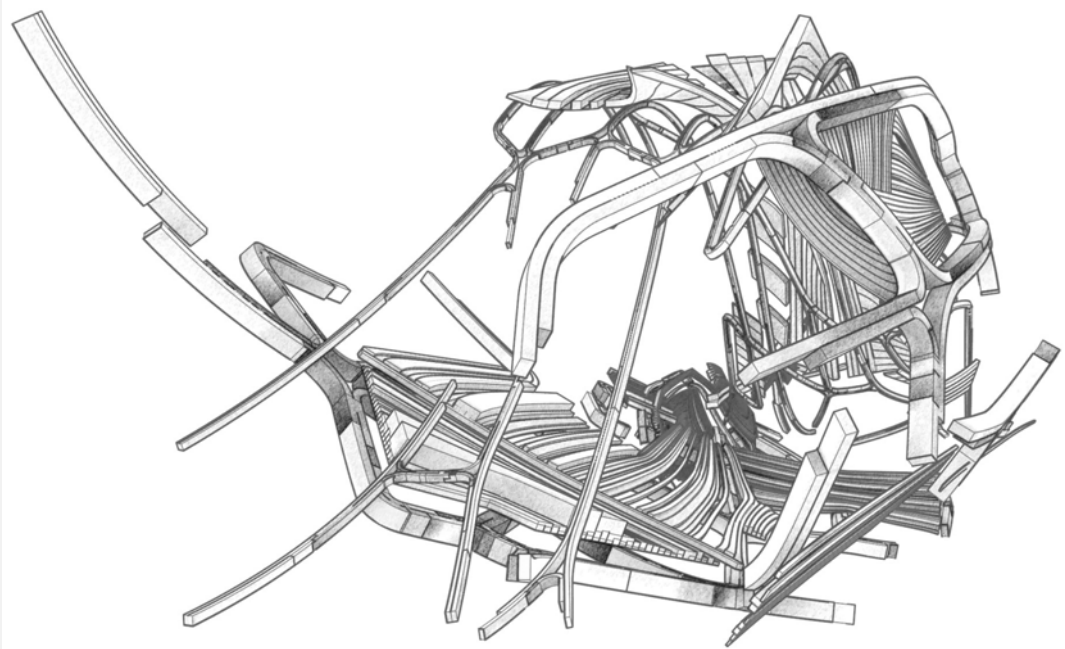


Figure 4.28: A speculative design proposal for a wooden enclosure.

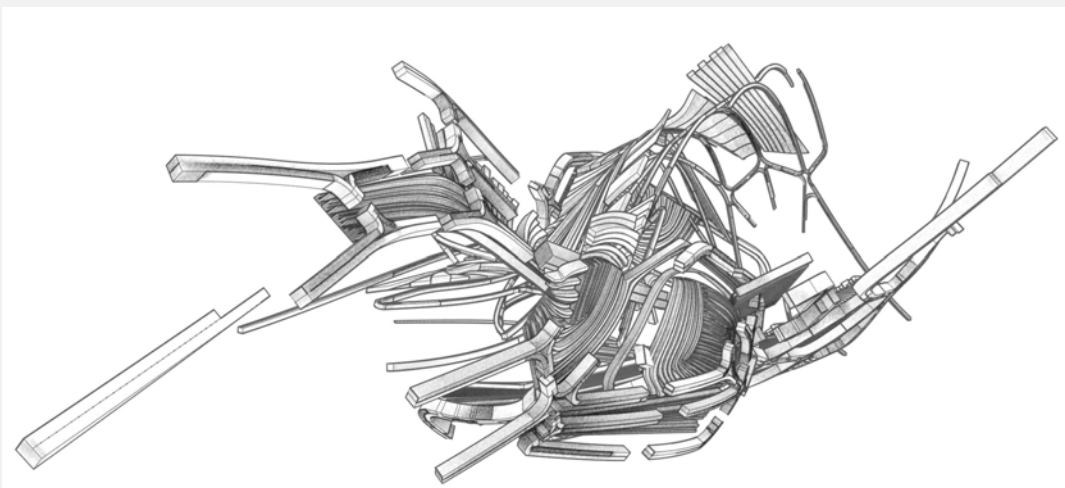


Figure 4.29: A speculative design proposal for a wooden enclosure.

Description

The present modelling experiment continues to follow the *comparative* research approach described in section 3.4. It takes as main point of comparison the Multi-Scalar Modelling methodologies developed in the first series of modelling of experiments described in the

previous section by applying and stress testing them through the development of the present experiment.

This workshop seminar offered an opportunity for three Early Stage Researchers, or ESRs (Kasper Ax, Paul Poinet and Tom Svilans), to collaborate around the shared topics of Multi-Scalar Modelling, engineered timber, and construction. The series of lectures at the beginning of the workshop offered a large variety of approaches and techniques in contemporary timber design, fabrication, and research. The workshop was an opportunity to contextualize the individual ESRs within this territory and speculate about the broader implications of each research project.

The first research project undertaken by Tom Svilans (ESR 2) focused around the integration of early stage design, material performance, digital fabrication technologies, and current industrial processes in laminated timber construction. It addresses the design-to-production digital chain and questions how new technologies and digital techniques can improve, hybridize, and re-think the current modes of laminated timber design and production.

The second research project led by the author (ESR 6) focused on the design exploration of free-form timber elements using one or multiple master surfaces in order to drive the global design. Another aspect investigated the potential of linking different scales together, from the main structural elements to secondary timber beams that could be aggregated in different locations for achieving different enclosures and spatial/architectural qualities.

The third research project undertaken by Kasper Ax (ESR 10), dealt with notions of assembly logistics and design for manufacture, wherein parameters that typically come at the very end of the design-to-production chain are considered and designed for well in advance, in tandem with other early stage design issues.

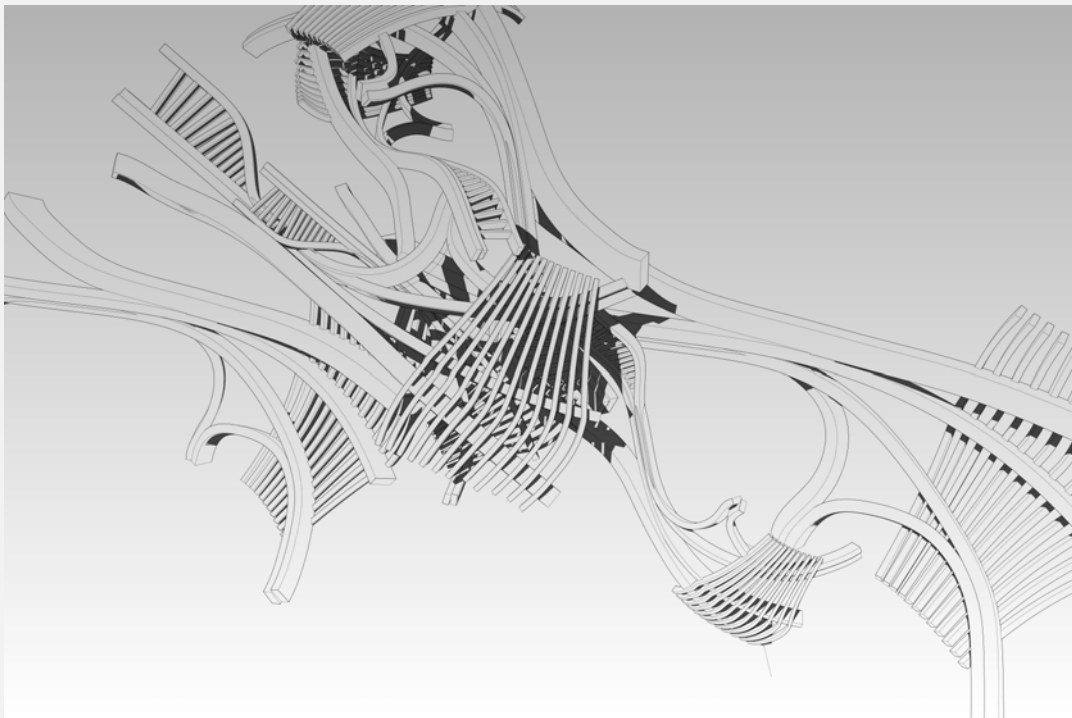


Figure 4.30: A speculative design proposal for a wooden enclosure.

The collaboration during the workshop was an attempt to find and assess the common links that exist between all three ESR projects and evaluate the feasibility of developing digital workflows between each project, with the ultimate aim of arriving at a shared goal - a physical demonstrator that could become a testing ground for all three projects.

Speculative design spaces

Using the Grasshopper environment and different custom script components (both in Python and C#), the three ESRs generated different design features, from two-dimensional and surface-based geometries to complete three-dimensional typologies. Different hierarchies have also been introduced through the different networks of timber elements by using different sizings, types of cross sections, dimensions and branching strategies. The workshop allowed the exploration of different design potentials, from integrated computational design workflows to more heterogeneous typologies. Taking precedents from existing large-scale glue-laminated timber projects such as the Centre Pompidou Metz and the Omega Swatch building by Shigeru Ban, the ESRs proposed a large speculative structural network that would host potential habitable spaces. This was a reconception of glulam members simply as structural components, imbuing them with programme and function beyond simply supporting structural loads. The glue-laminated timber as a composite performing element - not simply a structural member - was taken to the large-scale where it could be inhabited, interacted with, and used for services and utilities.

The different design iterations realized by the three ESRs allowed them to push the design boundaries beyond the sole consideration of material performance or pure structural design, by investigating how contemporary design tools and graph modelling methodologies – as developed in “A2 | Topological Mapping: Modelling Lap Joints”, “A3 | Topological Mapping: Surface/Projection-based modelling” and “A4 | Topological Mapping: Spatial Branching” – could enable the creation of speculative design spaces and explore the architectural potential of advanced timber constructions. Similarly to the modelling experiment “WS3 / Exquisite-Timber-Corpse Workshop: Collaborative Practice”, the Python library NetworkX was exploited to pass information between different levels of resolutions, from the abstract network designed interactively by the expert user, to the full detailing of each structural member.

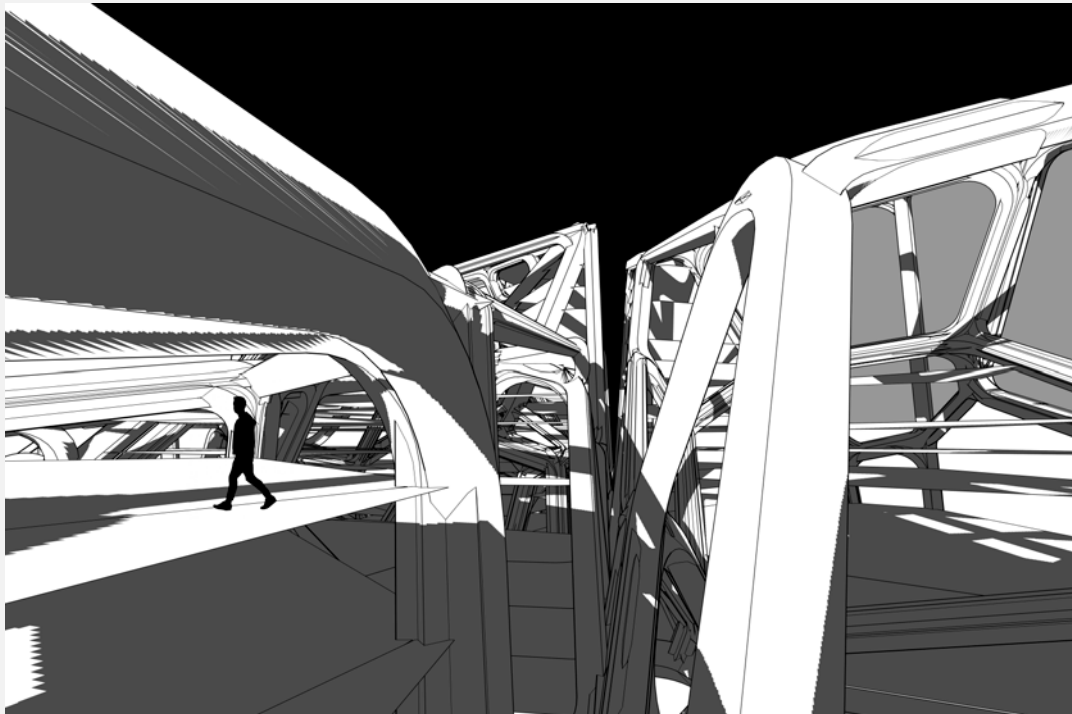


Figure 4.31: A speculative design proposal for a wooden enclosure.

Future work

This workshop served as a strong basis for the further development of a demonstrator made of

free-formed timber elements by the three ESRs involved in the described design explorations. While it was a good opportunity to speculate about the broader architectural implications of the involved construction methods and design models, the opposite end of the scale - the world of fabrication, tolerance, and individual elements - has been temporarily ignored. However, further work will be undertaken in order to refine and tune the design exploration by taking more into account material performance and fabrication constraints necessary for the completion of the mentioned demonstrator described in the next section.

4.2.5 A5 | The Tallinn Architecture Biennale Installation Competition

Experiment A5 | The Tallinn Architecture Biennale Installation Competition Category: demonstrator

Related research question: “How can a Multi-Scalar Modelling framework allow the designer to work across different scales in order to take into account multiple constraints related to material, fabrication and structural performances during both early design and late stages?”



Figure 4.32: Interior perspective of the design proposal for the Tallinn Architecture Biennale Installation Competition 2017.

Description

The present modelling experiment (illustrated in figures 4.18, 4.18 and 4.21) continues to follow the *accumulative* research approach described earlier in section 3.4 and takes into account the knowledge gained about Multi-Scalar Modelling strategies, (graph-modelling methods and Skeleton Models) from the previous experiments “A2 | Topological Mapping: Modelling Lap Joints”, “A3 | Topological Mapping: Surface/Projection-based modelling” and “A4 | Topological Mapping: Spatial Branching”. Furthermore, the present modelling experiment also takes into account knowledge gained about multiple design investigations carried through the previous experiments “WS3 | Exquisite-Timber-Corpse Workshop: Collaborative Practice” and “WS4 - Future Wood Seminar: Speculating Design Spaces”. These last experiment showcased the design potentials through the application of Multi-Scalar Modelling strategies for building design at early stages.

The present modelling experiment is the result of a common research elaborated by both the author (through the previous described digital experiments) and Tom Svilans (ESR 2). The research aims to elucidate and formalize the connection between material performance, Multi-Scalar Modelling, and early stage architectural design, in the context of free-form glue-laminated timber structures (Svilans et al., 2017). While the design and production of architectural objects deals with multiple scales and resolutions of data and information, this is

typically weighted towards either a top-down or a bottom-up approach. The ability to consider multiple scales at once and create bidirectional links between them offers the opportunity to even out this relationship and design more efficiently, thoroughly, and intelligently. This research uses Multi-Scalar Modelling methods to embed low-level material performance of glue-laminated timber into early stage design processes. This is situated in an applied architectural and industrial context, which looks at what early stage design means for multi-disciplinary architectural practices, and what material performance means for industrial timber fabricators. Through this demonstration, the research seeks to validate this design and employed Multi-Scalar Modelling methodologies as viable and productive alternatives to the more traditional design methods of producing these types of intricate structures.

Workflow and Multi-Scalar Methodologies

The implementation of the mentioned Multi-Scalar Modelling requires new ways of encoding and modelling the relationships between the scales. A graph-based, object-oriented approach is proposed. As such, a software library has been developed which facilitates the modeling and representation of glue-laminated timber assemblies through the different scales along with their material data, and which provides flexible production data such as workplanes and member-to-member relationships. This library and modeling workflow has been tested and demonstrated through the design and construction of a physical prototype. Through its conception and execution, the prototype shows how material properties and behaviours are managed throughout the whole process, and how fabrication parameters and constraints inform and alter the broader design through feedback strategies (Figure 4.33), similarly to the described methodology in figure 4.6. The next subsections highlight the design process through the different scales, from the macro to the micro level.

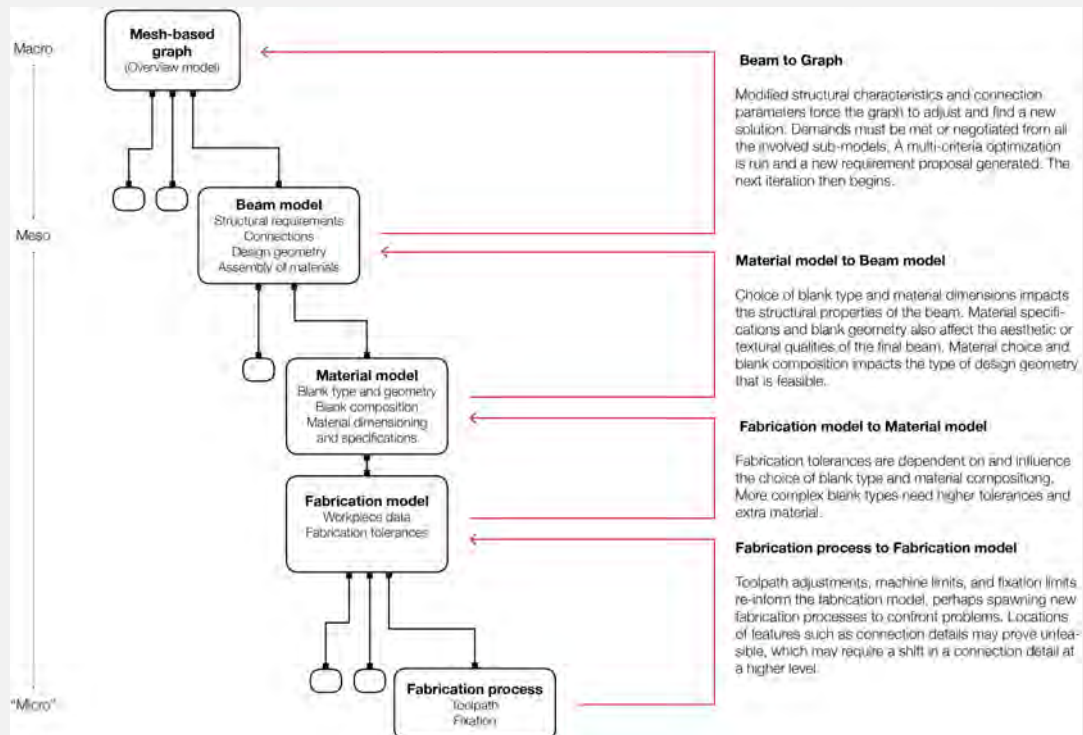


Figure 4.33: Multi-Scalar Modelling methodology employed during the design of the Tallinn Architecture Biennale Installation Competition design proposal. (Svilans et al., 2017)

4.2.5.1 Macro-level: global topological mesh and graph modelling

The global design has been manipulated and driven through an underlying topological mesh allowing the instantiation, mapping and directional control of the multiple wooden members composing the overall structure. Very early on, branching typologies were investigated in order to drive the overall design workflow. One of the main design challenges was to produce a consistent non-manifold topological mesh, required to increase the structural depth and spacial features at the footprints. Modelling investigations have been carried out in order to sort out the bifurcation strategies between the different wooden members.

Figure 4.34 highlights the global design workflow, from the setting of the global abstract network to the base mesh generation, branching pattern placement and mesh relaxation. The resulted geometry has been further processed through the incorporation of non-manifold modelling features (Figure 4.35) in order to allow more intricate branching and spatial qualities within the global design. Figure 4.36 displays the final design model, within which the wooden members have been mapped onto the relaxed non-manifold mesh.

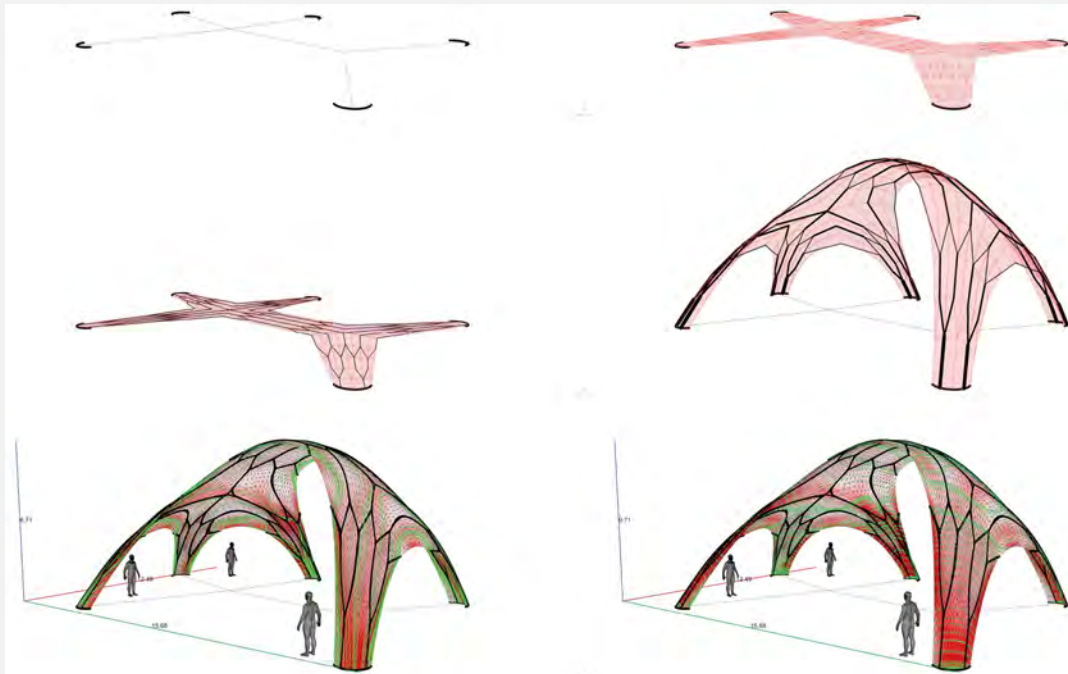


Figure 4.34: Branching pattern placement onto the global relaxed topological mesh.

A graph corresponding to the global design (Figure 4.38) was generated (with the help of GraphViz) at each generation to allow the computational designer to get rapid visual feedback on the structure's topology and wooden connections. The graph informs on the directionality of the wooden bifurcations, the degree of centrality and the thickness (or the number of laminates) of each timber element. The graph also allowed "regional" retrieval from the overall wooden timber structure in order to process each element for further fabrication.

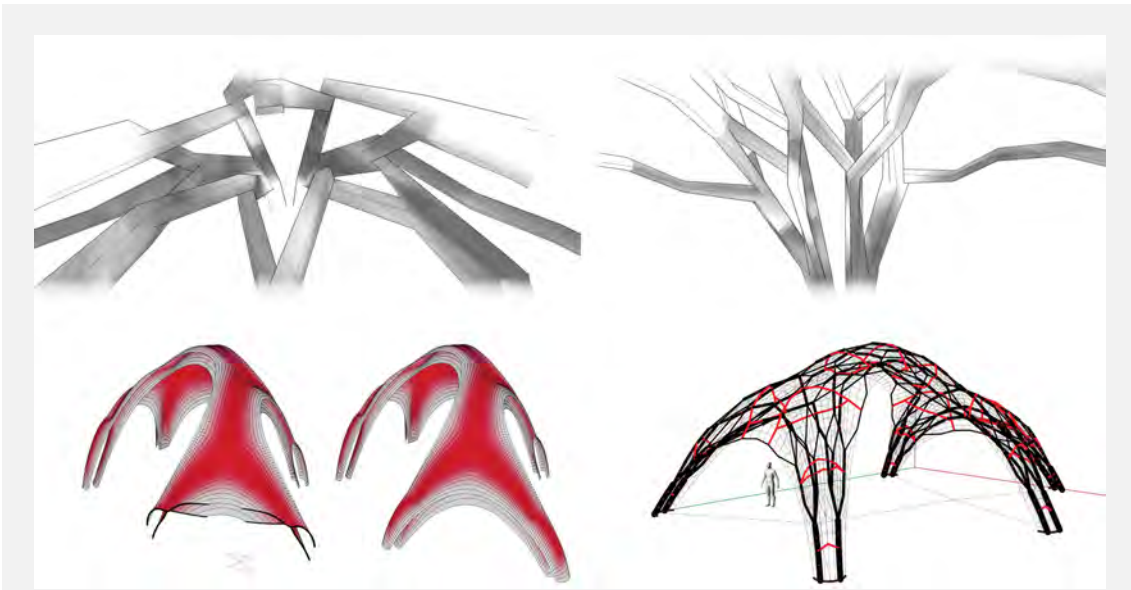


Figure 4.35: Global design process. Modelling features have been investigated to manage the splits and bifurcations of the different wooden members.

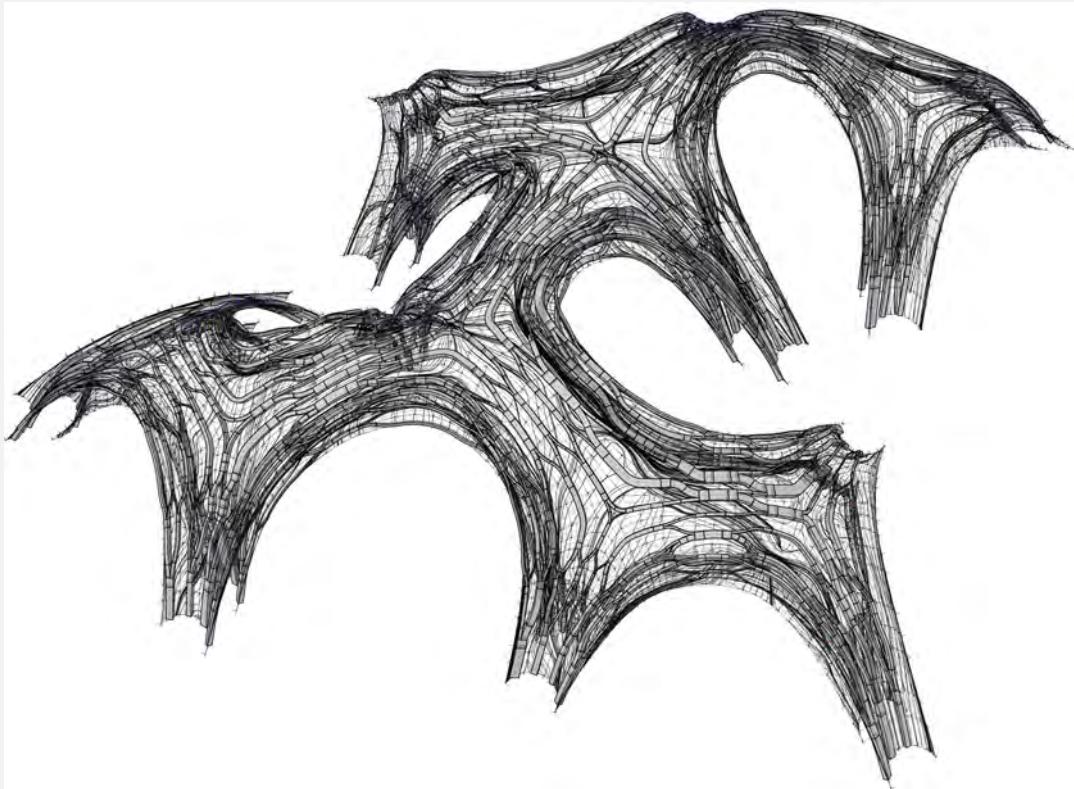
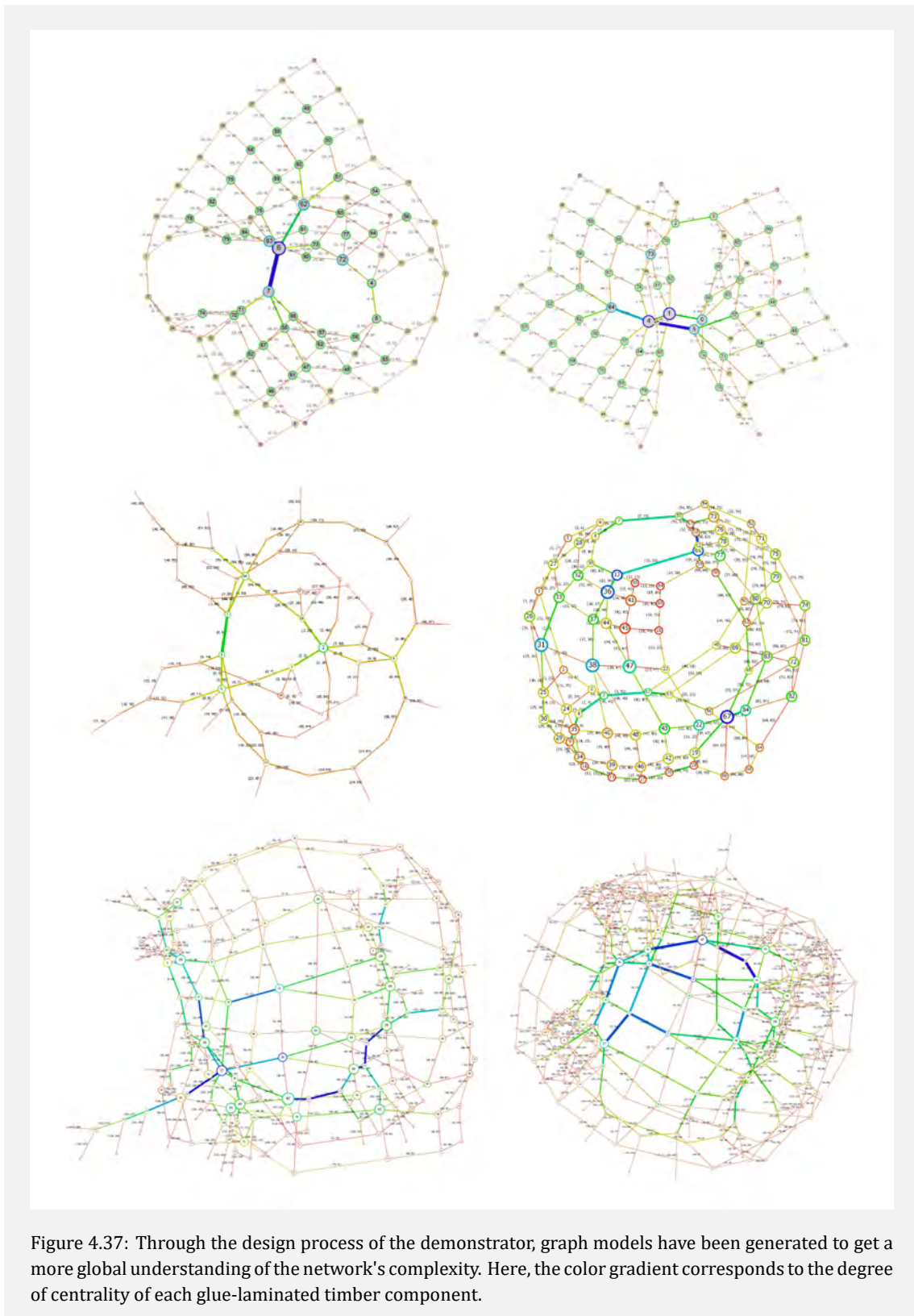


Figure 4.36: Final design and mapping of the different wooden members onto the non-manifold topological mesh.



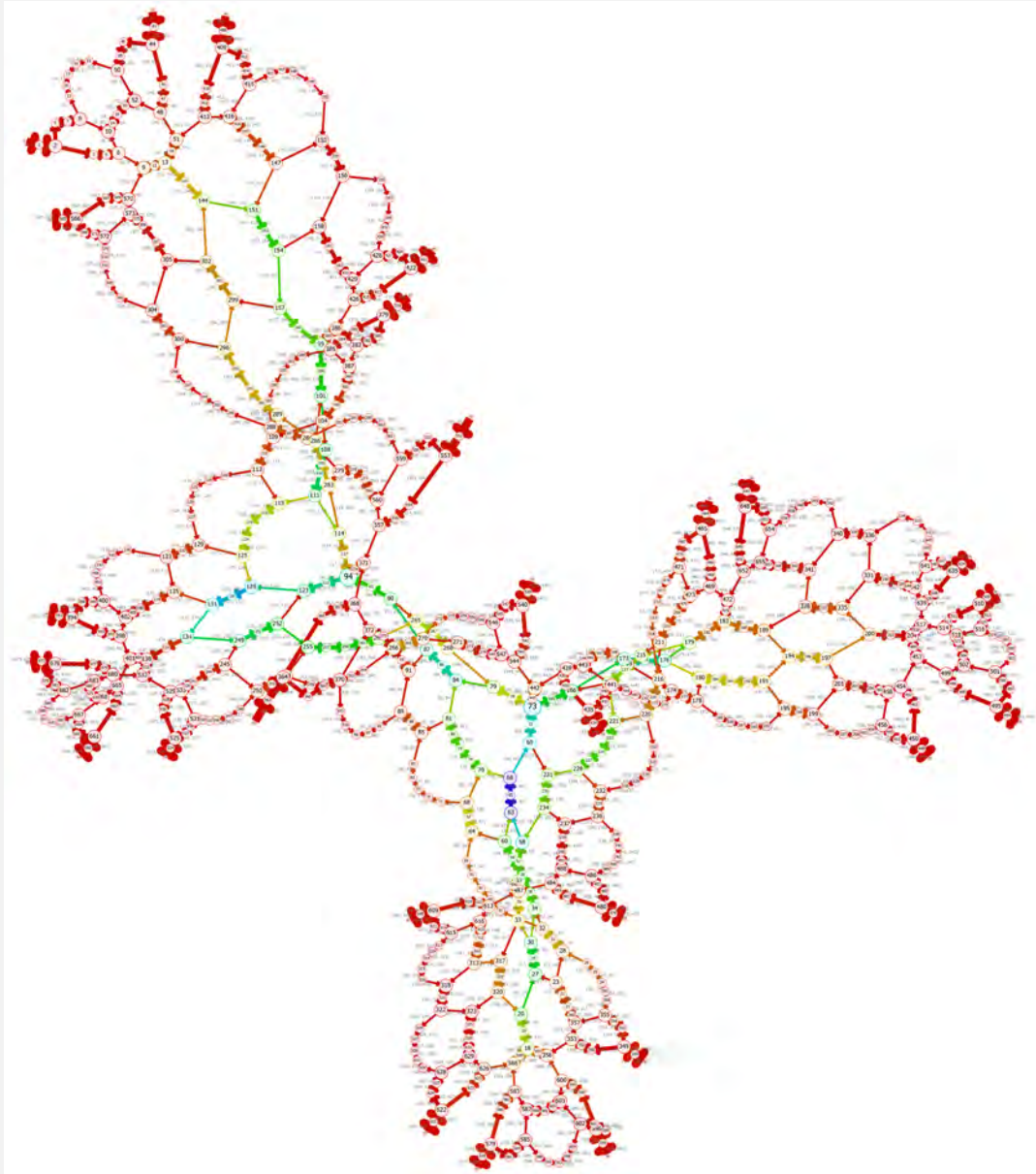


Figure 4.38: A graph-based model of the demonstrator provides a global overview of member characteristics and relationships. Here, the color gradient corresponds to the degree of centrality of each glue-laminated timber component. Also, each bifurcation's directionality is also displayed through an arrow whose thickness relates to the number of laminates contained within each member.

4.2.5.2 Meso-level: extracting the fabrication data from the assembly model

At meso level, the focus is placed on regional assembly strategies and analysis through the subdivision of the entire structure into multiple node “patches”, allowing partial geometrical description, and the extraction of forces from the global abstract network. In a similar way to the modelling experiments in “A2 | Topological Mapping: Modelling Lap Joints”, “A3 | Topological Mapping: Surface/Projection-based modelling” and “A4 | Topological Mapping: Spatial Branching”, the block model partitioning algorithm from the NetworkX Python library was used to fragment the global model into multiple sub-models.

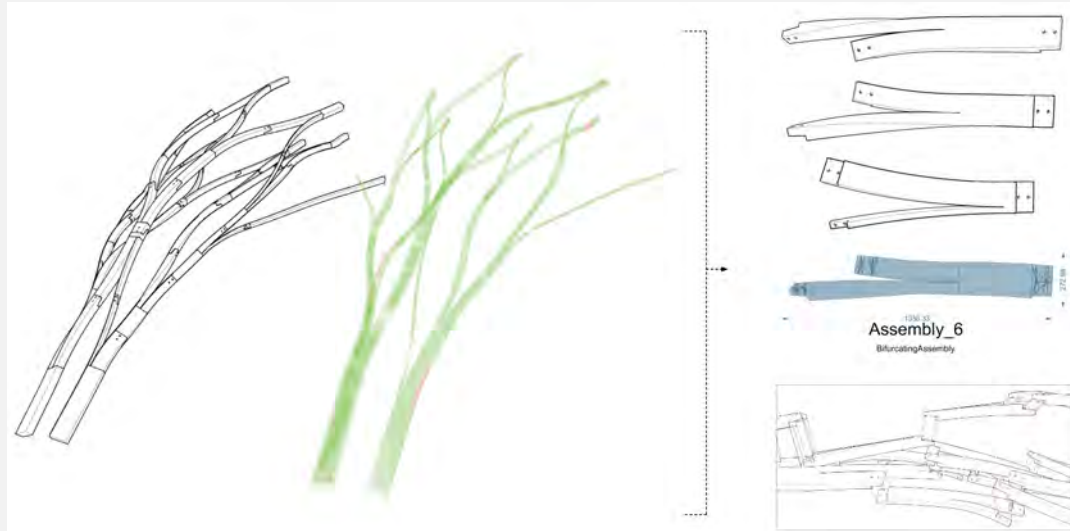


Figure 4.39: A regional design model is extracted from the global design proposal, showing both geometric information from the beam model and material information from the material model. In this case, the model is checked to see if the constituent lamellas of each module are within their bending limits. The modules are then extracted and prepared for further robotic fabrication.

4.2.5.3 Micro-level: molding, laminating processes and physical prototyping

Once all the timber elements were generated, some were extracted for further fabrication and springback behaviour testing. Custom algorithms were developed in order to extract the wooden blanks from the 3D model and generate the suitable formworks that have been further fabricated with 1.5 centimeters Medium-Density Fibreboards that were automatically nested and prepared for milling routines (Figure 4.40). Once the individual formworks fabricated, the wooden laminates were glued together and manually placed on their respective formwork. Additional constraints such as clamps and straps were added gradually until the wooden member takes its final shape (Figure 4.41).

After lamination of each timber branching element, the wooden members are scanned to check whether or not any springback behaviour occurred (Figure 4.42, bottom left). If the springback is very strong, the component might be laminated. If negligible, the neighboring elements might be adapted within the 3D model in order to match the “updated” physical model. Finally, once the wooden element is precise enough to be able to match the digital model, it can be robotically fabricated (Figure 4.42, right) using the integrated workplanes and milling paths (Figure 4.42, top left).

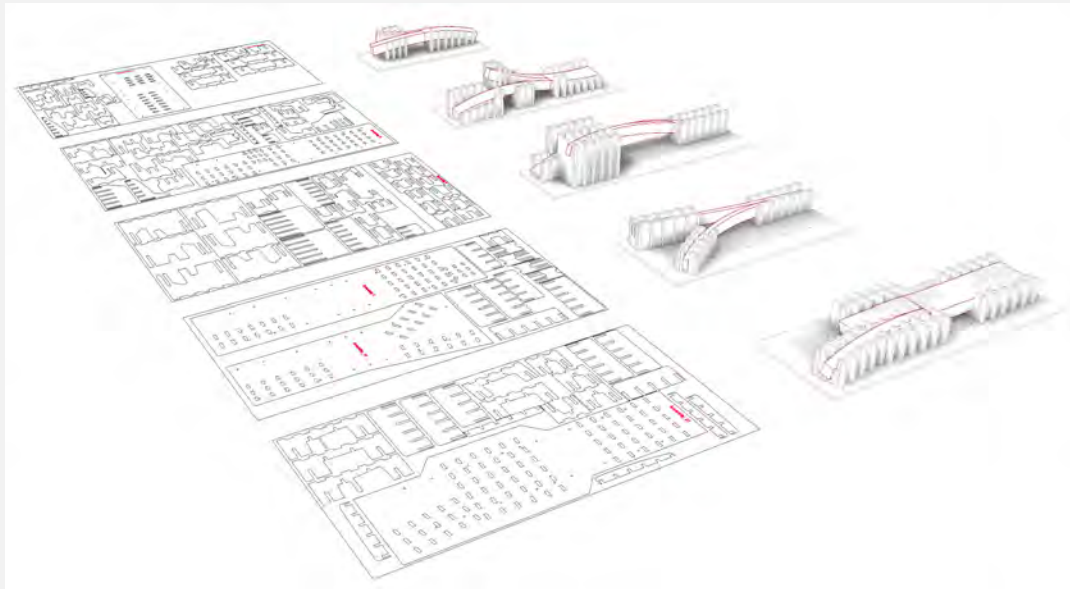


Figure 4.40: Mold generation and boards nesting.

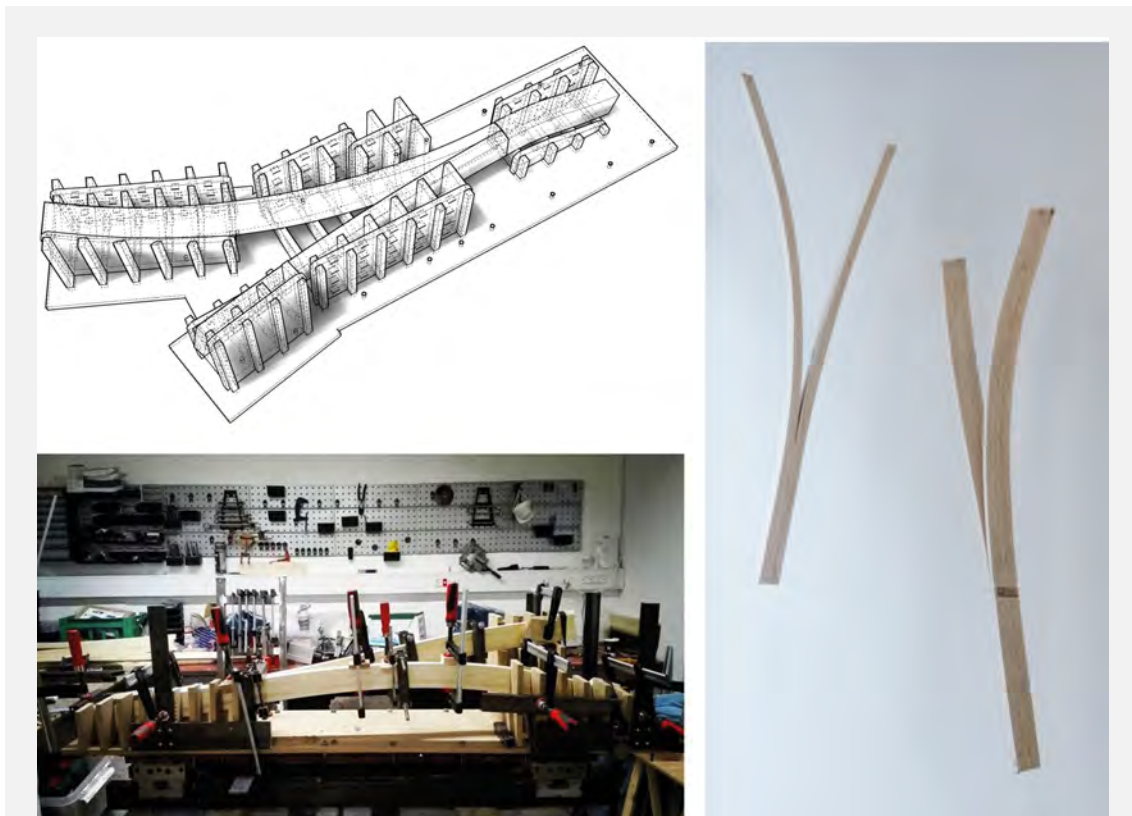


Figure 4.41: Molding and lamination processes.

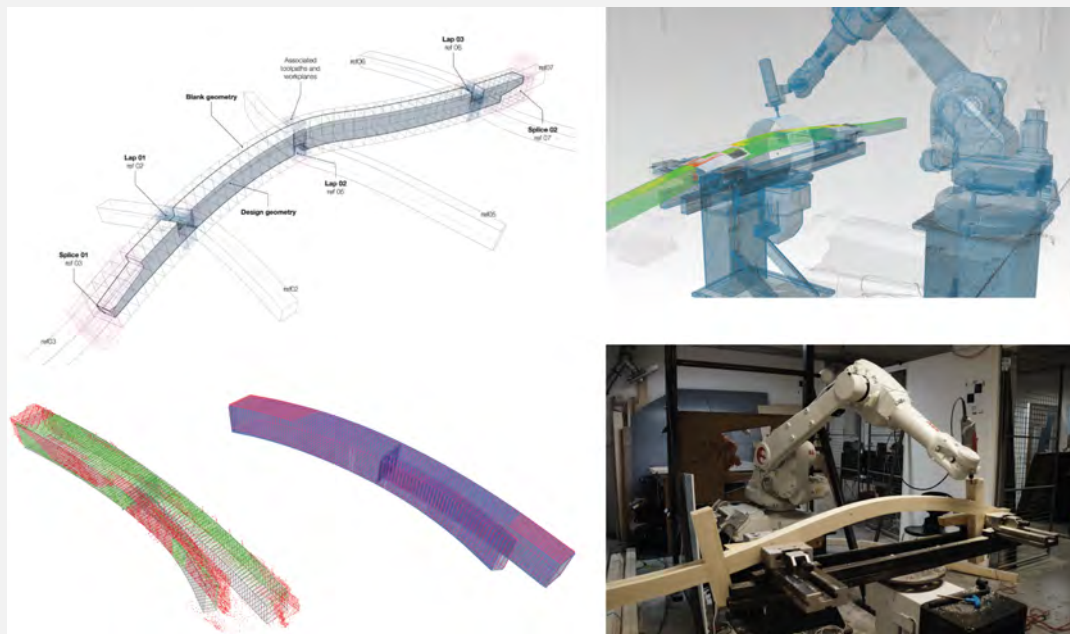


Figure 4.42: Scanning, tolerance testing and robotic fabrication processes.

4.2.5.4 Results and remaining challenges

During the design of the Tallinn Architecture Biennale Installation Competition and the fabrication of the demonstrator, the traditional communication protocols have been redefined. From the ground up, a clear digital infrastructure was agreed between the team members involved. Similarly to the Skeleton and Adapter Models described in section 2.5.3, interface models have been defined to ease data exchange between the different levels of resolution. However, as in *"WS1 / B-MADE Workshop: Fabrication Constraints and Assembly Logistics"*, communication bottlenecks have been observed at late stages during which many manual interventions were necessary to handle late design changes. Indeed, despite an important front-loading (Smith, 2007) strategy implemented by the designers, unforeseen changes still took place at the latest stages of the design process. This resulted into a disorganized collaboration workflow in which residual duplicated models kept circulating between the different team members without referring to the predefined higher level organizational framework. This led to the duplication of multiple residual models which only served for late-stage design purposes.

Here again, it has been discovered that communicating and passing information across external models and teams was a crucial point to in the design process. Therefore, more emphasis would need to be placed on the development of custom workflows in order to handle the described management related problems that appear at late stages.



Figure 4.43: 3D printed model submitted for the Tallinn Architecture Biennale Installation Competition.

4.3 Conclusion: from Modelling to Data Management Concerns

The different modelling-based experiments described in this chapter showcased the potential of Multi-Scalar Modelling strategies to tackle cross-level modelling challenges at the early design stages. Through graph modelling methodologies and graph software libraries, skeleton models and Multi-Scalar Modelling methodologies, the modelling-based experiments “A2 | Topological Mapping: Modelling Lap Joints”, “A3 | Topological Mapping: Surface/Projection-based modelling” and “A4 | Topological Mapping: Spatial Branching” and “A5 | The Tallinn Architecture Biennale Installation Competition” demonstrated how to deploy a sustainable workflow that can pass information across multiple sub-models, levels and resolutions. However, we have seen that issues related to the latest design stages still remain as it is very difficult to predict changes in the design workflow despite a strong front-loading strategy (Smith, 2007). So far, the modelling-based experiments rooted in precedents in academia and speculated on how design workflows could be improved through the developed Multi-Scalar Modelling strategies. Although they enhanced the undertaken design workflows during the undertaken workshops, demonstrators and the Tallinn Architecture Biennale Installation Competition proposal, it has been observed that challenges related to communication bottlenecks and data management concerns remained. Therefore, the next chapter will specifically focus on these challenges. As the research focus is shifting from modelling to data management concerns, the *expansive* research approach described in section 3.4 is used here to transition to the next chapter that will now focus exclusively on data management concerns. The *expansive* mean of composing experiments does not follow “strict successive or linear orders or directions. [...] Rather than deepening [...] knowledge of a domain, this approach widens [the] perspective and extends the concerns [that designers] should include in [their] praxis.” (Krogh et al., 2015). The expansive approach therefore attempts to broaden the research perspective by investigating multiple domains in parallel.

More specifically, the next chapter will assess the importance of common communication protocols and platforms, the need for search interfaces to better curate data-rich models and manual interventions, and the necessity to establish more robust digital infrastructure supporting schema-based workflows and late stage design modifications. Those different strategies are also necessary to fully deploy Multi-Scalar Modelling, which can transform the design process not only at early design stages (as it has been described in this chapter), but throughout the late and fabrication stages as well in order to keep cross-platform design consistency amongst the multiple trades and consultancy practices involved in the design process.

Chapter 5

SCHEMA-BASED EXPERIMENTS AND SEARCH INTERFACES FOR LATE STAGES

From the previous realization that focusing solely on modelling issues at early stages does not necessarily solve or prevent late stage changes in the design process, the present chapter takes a look at the other end of the spectrum, at the very latest stages. The focus is therefore shifted from modelling/simulation to communication concerns that interrogate how data can be better treated and curated, from its visualization and assembly to its query and leverage ([Poinet et al., 2018](#)).

The present chapter is divided into three main sections.

- **From Data Visualization to Adaptive Queries: Inter-linking Scales and Reconfiguring Hierarchies**

Taking Design-to-Production's everyday practice as a precedent, the present chapter investigates the firm's working methodologies for project management at late stages. The first section looks at the data classification strategies used at Design-to-Production and proposes new ways to better understand their intricate data structures that are generated at the latest stages. This would help to improve data queries, navigation interfaces and targeted retrievals of information to better answer the user's concerns and direct needs.

- **Building, Transferring and Receiving Hierarchies**

The second section is focusing on the development of a prototypical application named *SchemaBuilder* allowing the building and assembly of user defined custom schemas containing geometrical information nested from the bottom up.

- **Cross-Practice Collaborations: Workshops and Case Studies**

Finally, the last section describes various case studies and workshops that apply and links the different Multi-Scalar Modelling strategies described in the previous chapter and the schema-based design workflow methodologies developed in this chapter.

Within the present chapter, the *accumulative* research approach and the knowledge gained throughout the series lead to the development of multiple experimental and prototypical web-based applications, such as: *LayerFootprint*, Cross-Practice Collaborations: Workshops and Case Studies, *LayerStalker* and *SchemaBuilder*

5.1 From Data Visualization to Adaptive Queries: Inter-linking Scales and Reconfiguring Hierarchies

This section looks at the data structure that is organized, maintained and deployed within the consultancy practice Design-to-Production to handle data-rich models used at late stages in the design process. Based on the investigation of the current working methodology, different concepts and prototypical applications have been developed in order to both visualize hierarchies, navigate amongst them, unravelling implicit relationships and perform adaptive queries.

Here, the Terminal Pavilions project realized for the Oslo airport is an interesting example to study, especially regarding the methodology that was employed to classify the information throughout the whole design-to-fabrication process. Like most projects of Design-to-Production, the data is organized within the LayerTable from the modeling environment Rhino3d. Parent and intermediate layers act as containers of abstract information (or *classes*), whereas the layers situated at the leaves of the data structure represent geometrical data (see figure 5.2).

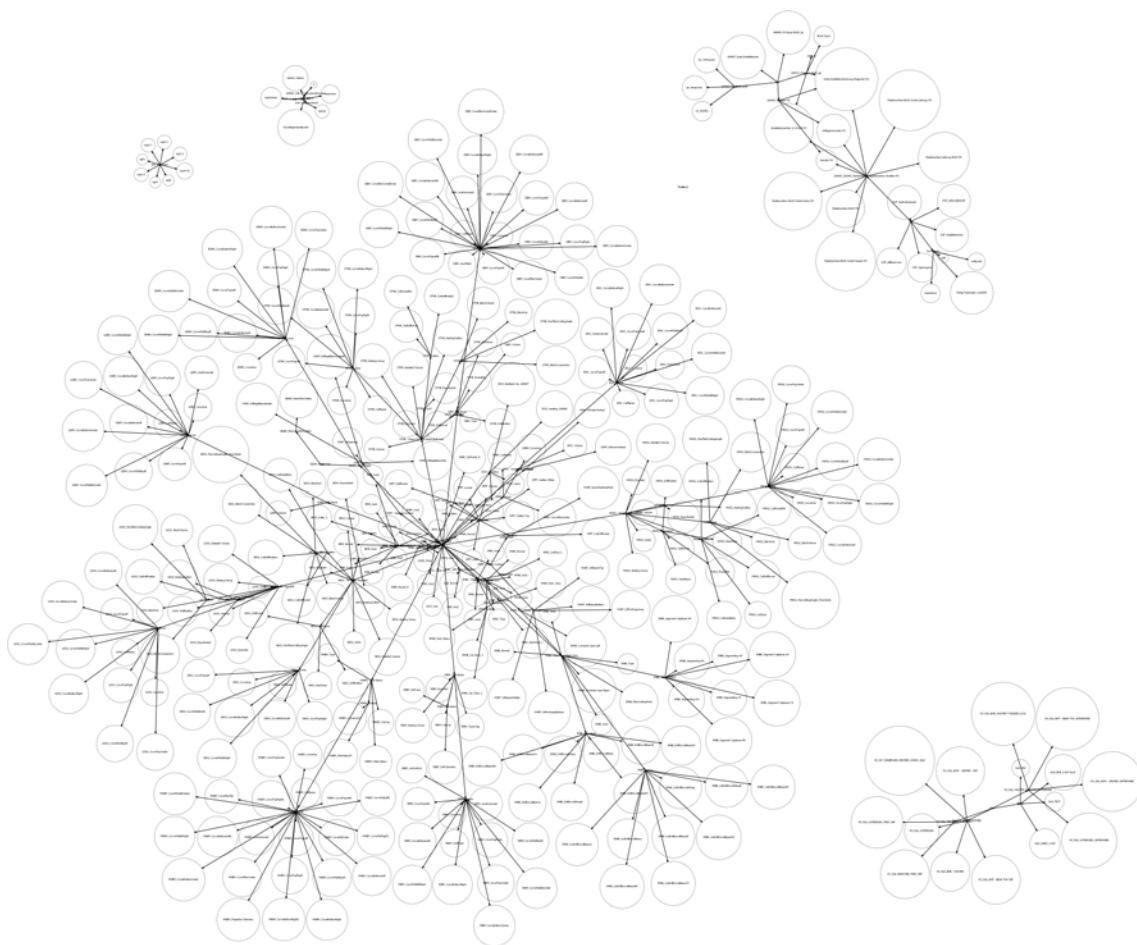


Figure 5.1: Representing complex information: the Rhino3D LayerTable of the Terminal Pavilions project represented as a [DAG](#)

5.1.1 B1 | Visualizing Hierarchies: LayerFootprint

Experiment B1 | Visualizing Hierarchies: LayerFootprint Category: prototype

Related research question: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

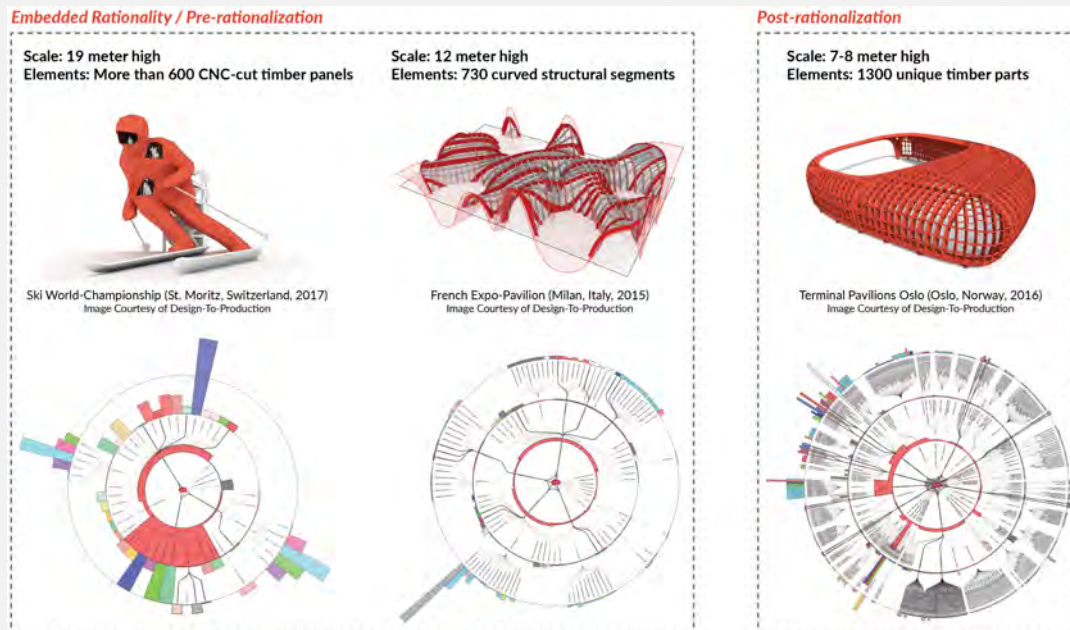


Figure 5.3: Comparison between different 3D-models and their respective data structure. From left-to-right: Terminal Pavilions of the Oslo airport, Ski World-Championship wooden sculpture at St. Moritz, French Pavilion realized for the Milan Expo in 2015. One can notice that the intricacy of the visualized database does not depend on the related project's scale.

Description

In a similar way to the modelling-based experiments “A1 | BlankMachine: Propagating the blanks” and “A2 | Topological Mapping: Modelling Lap Joints”, the present schema-based experiment bases itself on existing practices and precedents — here, the way Design-to-Production organizes its 3D elements within Rhino3D files.

In order to get a better understanding of the data structure contained in the 3D files produced by Design-to-Production, the traditional Rhino3D LayerTable interface GUI has been reinterpreted as a DAG in order to unfold and better represent the full complexity of the data model. However, if the DAG representation is not managed properly and the display remains unordered, it can become quite messy and not very useful from a UX perspective (Figure 5.2, top). Here, the graph shows the relationship between parent and child layers as well as their respective colors. The overall layout (which was generated through the Graph Visualization Software GraphViz) is not as readable as the LayerTable, even though its inherent complexity can be fully unfolded. Therefore, the popular data-visualization “Sunburst Diagram” layout has been chosen (Figure 5.2, bottom) to display the hierarchical relationships existing in the LayerTable. A sunburst diagram is usually represented in a circular manner, in which the child level is represented as a circular offset from the parent level. Each level is subdivided itself by the amount of existing “siblings” layers into many different circular arcs that are here also offset according to the number of objects contained within the corresponding layer.

This data visualization tool, named *LayerFootprint*, has been developed to display the inherent complexities (or “fingerprint”) of the Terminal Pavilions project (Figure 5.2). *LayerFootprint* can also be reapplied and stress tested against any other Rhino3D model which uses the software’s LayerTable to store and classify the data. It was therefore rather straightforward to reapply the same tool for any other project processed at Design-to-Production within the Rhino3D modelling environment. Interestingly, the intricacy of the database visualized by this tool does not depend on the respective project’s scale. Indeed, if one compares the fingerprints between the Terminal Pavilions of the Oslo airport and the sculpture built for the Ski World-Championship at St. Moritz or the French Pavilion realized for the Milan Expo in 2015, it is noticeable that there is no correlation between the size of the project and the complexity of the respective LayerTable (Figure 5.3). On the contrary, the Terminal Pavilions of the Oslo airport show a very intricate database despite their quite modest size (7-8 meters for the tallest) and the sculpture built for the Ski World-Championship at St. Moritz presents a rather simple data structure despite its much bigger scale (19 meters high).

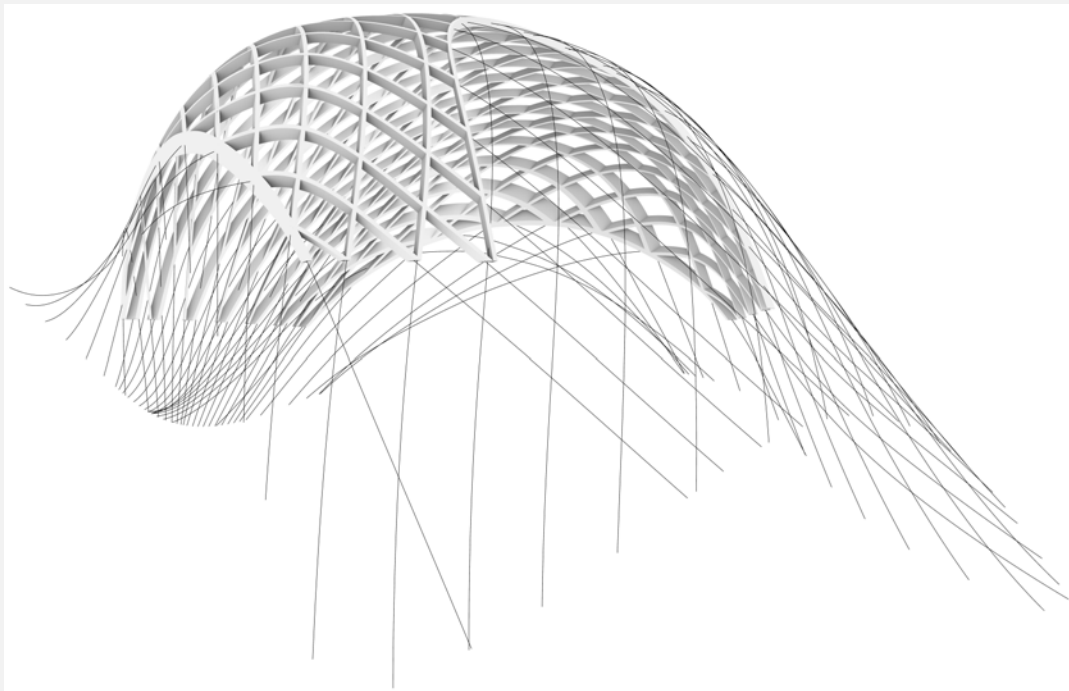


Figure 5.4: Early design study at Design-to-Production for a gridshell project.

Results and limitations

The fingerprint’s complexity depends on whether the architectural project has “pre-rationalized” or “post-rationalized”^a. Following those definitions, it can be suggested that both the French Pavilion realized for the Milan Expo in 2015 or the sculpture built for the Ski World-Championship at St. Moritz have either been “pre-rationalized” or embedded rationality at the very early design stages. In contrast, the Terminal Pavilions of the Oslo airport was “post-rationalized”, explaining therefore the multitude of different object types which had to be defined in order to discretize the entire project, enabling its feasibility. This resulted into a highly intricate data structure – the software’s LayerTable. Although it has been argued in section 2.2.3 that the design effort should not only be placed at early stages but design strategies and tools should be developed as well for late stages, the above observations remind us that the efforts placed at early stages are really considerable and have strong consequences onto the design process afterwards. During a secondment at Design-to-Production, the author had the opportunity to work on the conception phase of a gridshell project 5.4. Its geometry

had to be optimized so that all timber elements were single-curved, easing therefore the fabrication process. Here, the design effort has been placed during the early design phase.

Through the different stress tests carried against different source models, it has been observed that the amount of information displayed by *LayerFootprint* might be too intricate and complex to be grasped at once as a whole. Therefore, the next schema-based experiment attempts to extend the capabilities of *LayerFootprint* by integrating zoomable features, allowing the user to focus on a particular level of the hierarchical data structure.

^aThose terminologies have been introduced by Robert Aish to classify buildings depending on whether or not early-stage design strategies have been deployed to ease the fabrication process at the later stages: *"In the pre-rationalization method, the building geometry is usually predetermined by a number of geometric constraints set in the early design stage, whereas with post-rationalization, the building geometry is retroactively simplified to accommodate realistically constructible components."* (Attar et al., 2010) Stating that *"[...] existing rationalization techniques cannot fully address the design challenges of conflicting configuration and fabrication constraints"*, Aish goes further by introducing the concept of *"embedded rationality"*, which *"[...] in contrast to pre-rationalization and post-rationalization [...] is a method for implicitly combining fabrication constraints into an interactive framework for conceptual design."*

5.1.2 B2 | Navigating across Hierarchies: LayerExplorer

Experiment B2 | Navigating across Hierarchies: LayerExplorer Category: demonstrator



[Video link to the experiment](#)

Related research question: “How would an ideal Multi-Scalar Modelling AEC-model look like and which requirements would it have to fulfill all user’s requests? How could the multi-scalar model be interacted and which User Interface (UI) and User Experience (UX) concepts would be needed?”

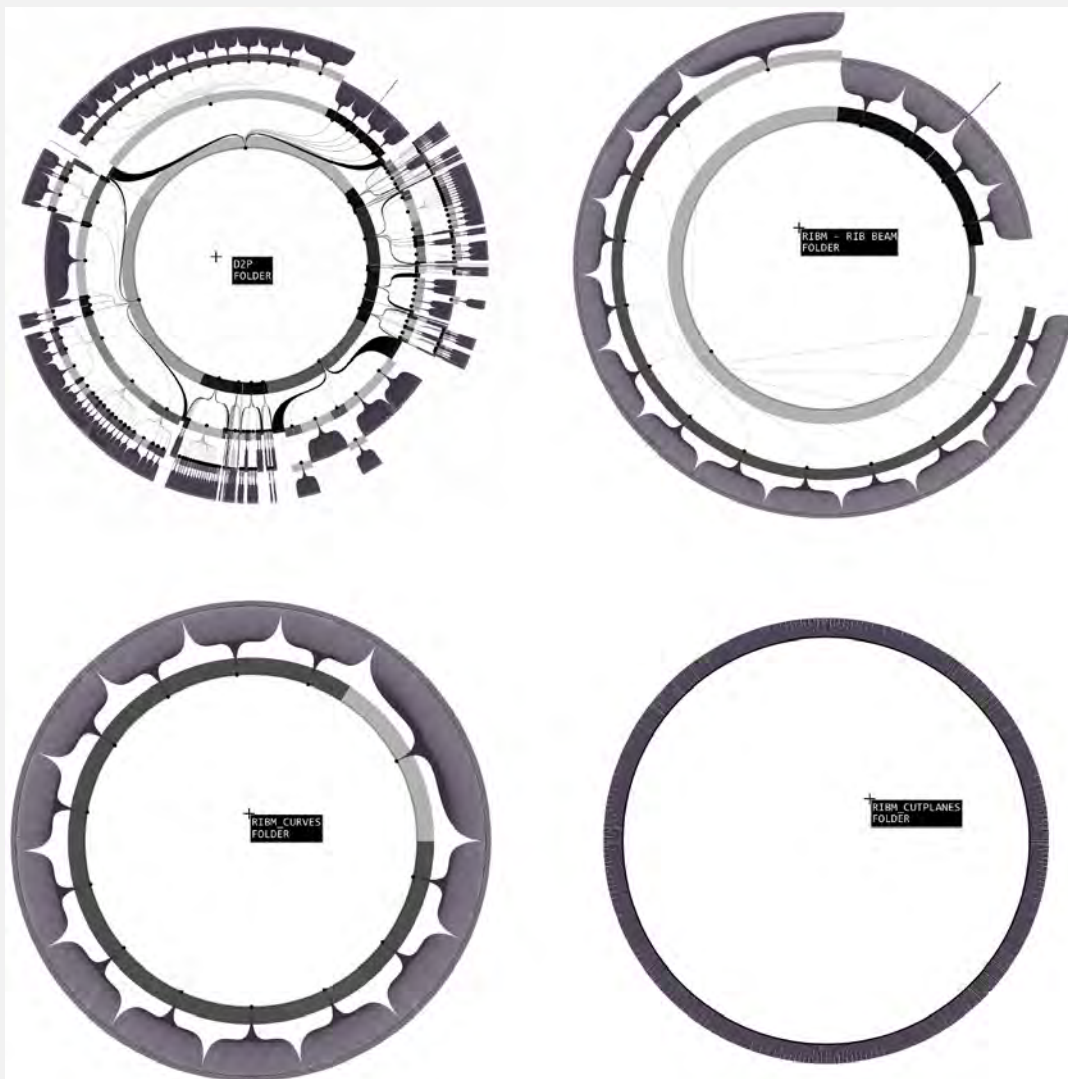


Figure 5.5: *LayerExplorer*, a prototype interface allowing the user to navigate between the different hierarchical levels from the Rhino3d’s LayerTable.

Description

The present schema-based experiment follows the *accumulative* research approach described in section 3.4 and takes into account the knowledge gained in the previous experiment “B1 / Visualizing Hierarchies: LayerFootprint”. As it has been observed that the layer hierarchy unraveled by *LayerFootprint* might be too intricate and complex to be grasped and comprehended as a whole, the author suggests that a zoomable interface would be necessary to let the user navigate between the different levels of resolution that have been predefined. Therefore, the sunburst diagram developed for *LayerFootprint* and described above has been enhanced by UI features allowing the user to get more control on the represented data. During the workshop seminar “Simulating for Design” organized within the InnoChain training network, the author had the opportunity to implement such zoomable feature built within the Processing environment platform, based on an existing Processing sketch ^a and adapted to Design-to-Production’s particular data structure.

LayerExplorer has been written in JavaScript within the Integrated Development Environment (IDE) Eclipse and deployed as an independent prototypical Java application. In order to use this application, a custom C# scripting component has been developed within Rhino-Grasshopper in order to extract the hierarchical data structure from the LayerTable and translate its content as a Directory Tree Structure within the File Explorer GUI in Windows. Through LayerExplorer, the user is asked to input a root of a specific Windows path that needs to be visualized and explored. LayerExplorer has been stress tested against different Rhino3D models by extracting their respective LayerTable.

Once the data structure is rendered within *LayerExplorer*, the expert user is able to zoom-in (by pressing the key “+”), zoom-out (by pressing the key “-”), switch branches (by pressing the key “b”) and focus on a particular level of the hierarchical data structure and explore the different geometrical items situated on the leaves of the Rhino3D’s LayerTable. For the sake of demonstration, hovering on the leaves of the data structure rendered in LayerExplorer would display a tooltip informing the user on the Rhino3D object’s Globally Unique Identifier (GUID). Further interaction features could be implemented within the LayerExplorer interface.

Results and limitations

LayerExplorer enables the possibility to zoom on particular, local levels of the hierarchical data structure. Thus, instead of continuously trying to comprehend the whole database at once through the complete overview of the sunburst diagram, the user is finally able to focus on local levels and navigate between them (Figure 5.5). However, the present experiment operates outside of the 3D modelling environment, as the data structure has been simply externalized to be further manipulated. Therefore, the next schema-based experiment, acting as a design probe, will investigate how custom GUI features can directly interface within the 3D model itself.

^ahttp://www.generative-gestaltung.de/1/M_5_01_TOOL

5.1.3 B3 | Reconfigurable Hierarchies and Adaptive Queries

5.1.3.1 Reconfigurable Hierarchies

As mentioned previously (2.1.1), hierarchies defined in a top-down manner tend to be inflexible especially if one would attempt to reconfigure them based on implicit relationships. In the case of the data structure handled by Design-to-Production, information could be classified in different ways, considering either the geometry type or the parent abstract class (called here “component”) at the root of the data structure (Figure 5.7). Choosing any particular hierarchical order would at some point limit the user’s exploration and experience. If one places the component type at the root, it would be difficult to collect data based on the geometry type, and vice-versa. Therefore, instead of relying exclusively on existing data structures and [UIs](#) integrated, available today in most [CAD](#) environments (such as the LayerTable in Rhino3d), custom solutions need to be developed to display, cross-link information containers in order to navigate from the component to its geometry types (from the root to the leaves) and vice-versa. Indeed, the information hierarchy of the data tree should remain flexible, so one can reverse, transform or simply manipulate the order of the information containers. Then, any type of information can be placed at the center of the data structure, from which further connections can be drawn.

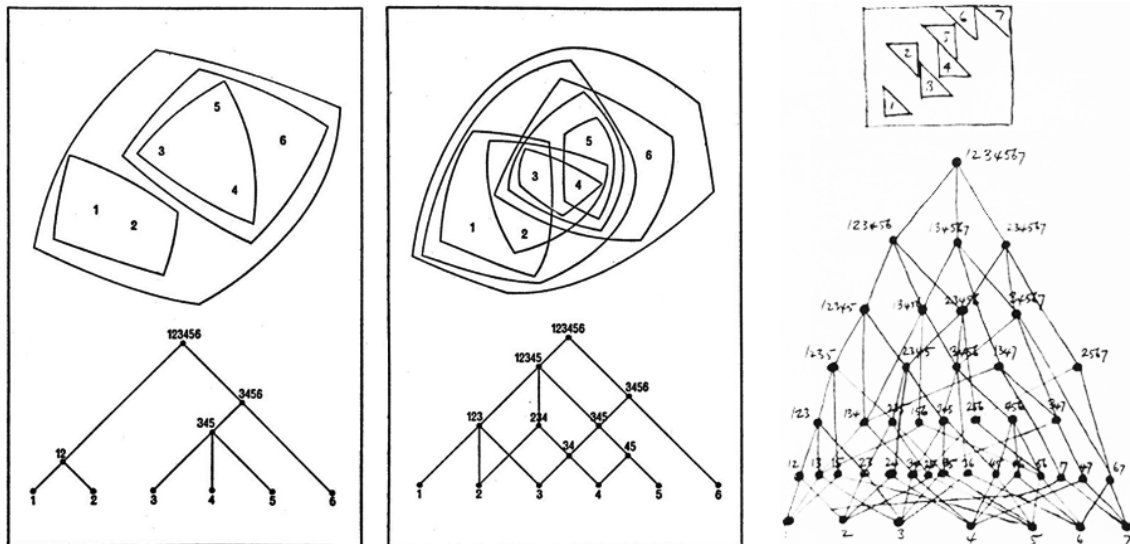


Figure 5.6: Hierarchical, tree-like structure (left) vs. semi-lattice, overlapping subsets (middle and right). ([Alexander, 1965](#)).

This need for reconfiguring hierarchies and adaptive display of existing implicit relationships resonates strongly with a statement from Christopher Alexander, emeritus professor at the University of California, Berkeley and influential theorist, who argued in favor of flexible design systems that can respond to changes affecting the initial hierarchical setup:

“As the relationships between functions change, so the systems which need to overlap in order to receive these, relationships must also change.” ([Alexander, 1965](#))

Indeed, in his famous essay “A City is Not a Tree”, Christopher Alexander warned about the tree structure as the principal medium to represent and rationalize complexity. He demonstrated through the example of urban planning and the organization of cities that trees are too reductive and do not give any insights into the implicit relationships that exist between different isolated subsets:

“The tree – though so neat and beautiful as a mental device, though it offers such a simple and clear way of dividing a complex entity into units – does not describe correctly the actual structure of naturally occurring cities, and does not describe the structure of the cities which we need.” ([Alexander, 1965](#))

Instead of relying on purely tree-like structures, Alexander chose to work with “semi-lattices”, enabling overlapping and the setting up of implicit relationships between different subsets:

“Even before “Notes on the Synthesis of Form” was published in 1964, Alexander had already moved on to a new structure, the semilattice, thanks to the expanded capabilities of HIDECS 3. The new version of the program supported the visualization of more complex information structures because it calculated multiple links and variables that could overlap with one another and link to more than one node, unlike trees, in which relationships could not overlap” (Steenenson, 2017, 34)

The warning from Christopher Alexander about the oversimplification of design systems through the design and visualization of tree-like data structures is very relevant in the context of Multi-Scalar Modelling for Building Design. Indeed, it has been seen in section 2.1.3 that most of the GUIs available today in CAD software to manage complexity and draw relationships between objects take the shape of neatly highly organized tree structure that hardly allow the establishment of adjacent connections between distinct subsets or subgroups of objects. In the modelling context of Design-to-Production, tree-like structures do not answer all needs because adjacent connections exist between objects located at different leaves. Figure 5.7 exemplifies such case in which the hierarchical data structure of two different instances of a specific *MainBeam* object are unfolded. At the leaves of two different distinct *MainBeam* objects, all curves can be collated within one cluster by calling their common geometrical denominator curve type.

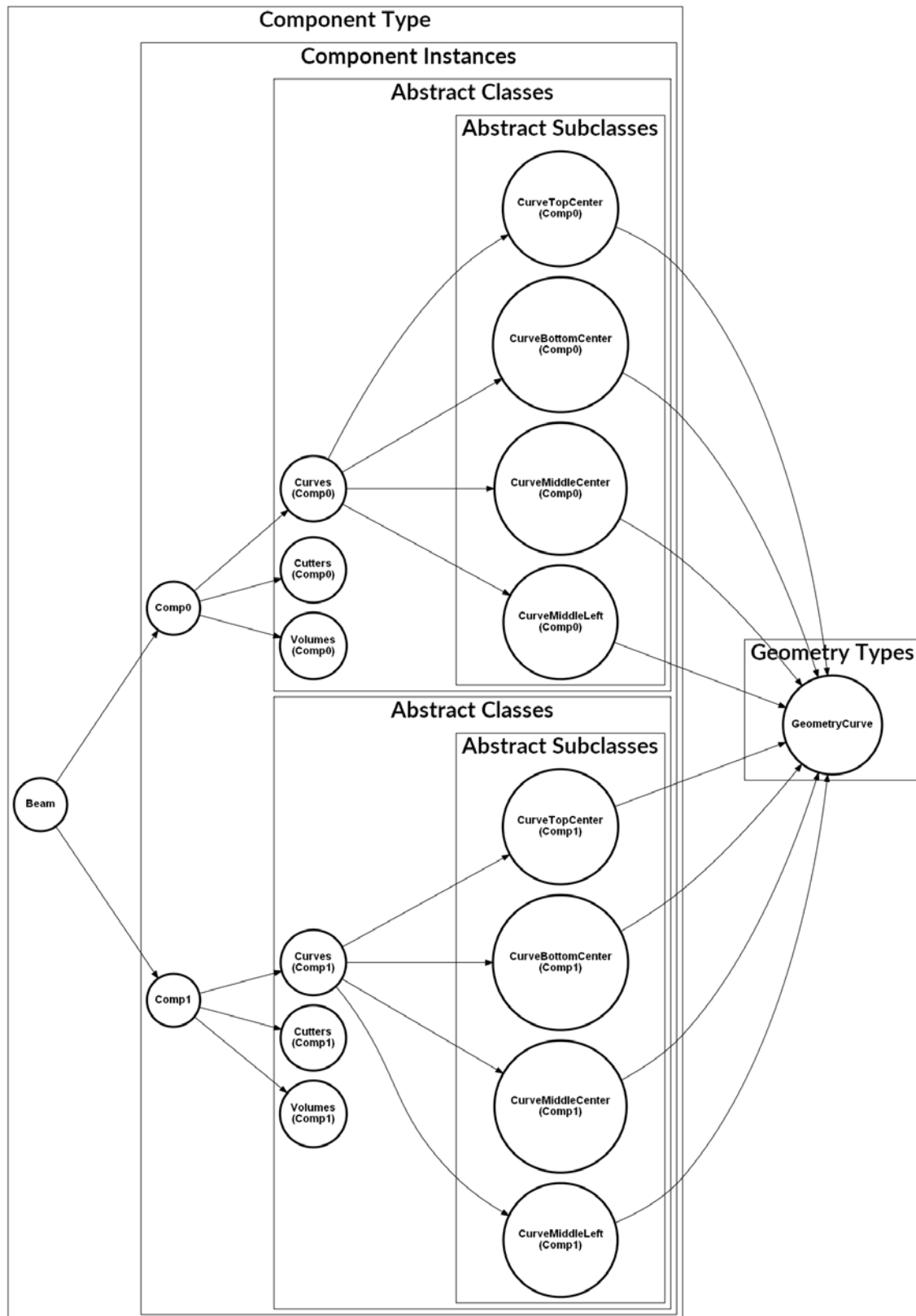


Figure 5.7: This diagram highlights the fact that data can be gathered from different leaves of the original tree structure according to common properties. From left-to-right: component type, component instances, abstract classes, abstract subclasses containing geometrical data. The latter can be gathered and queried according to common geometrical types.

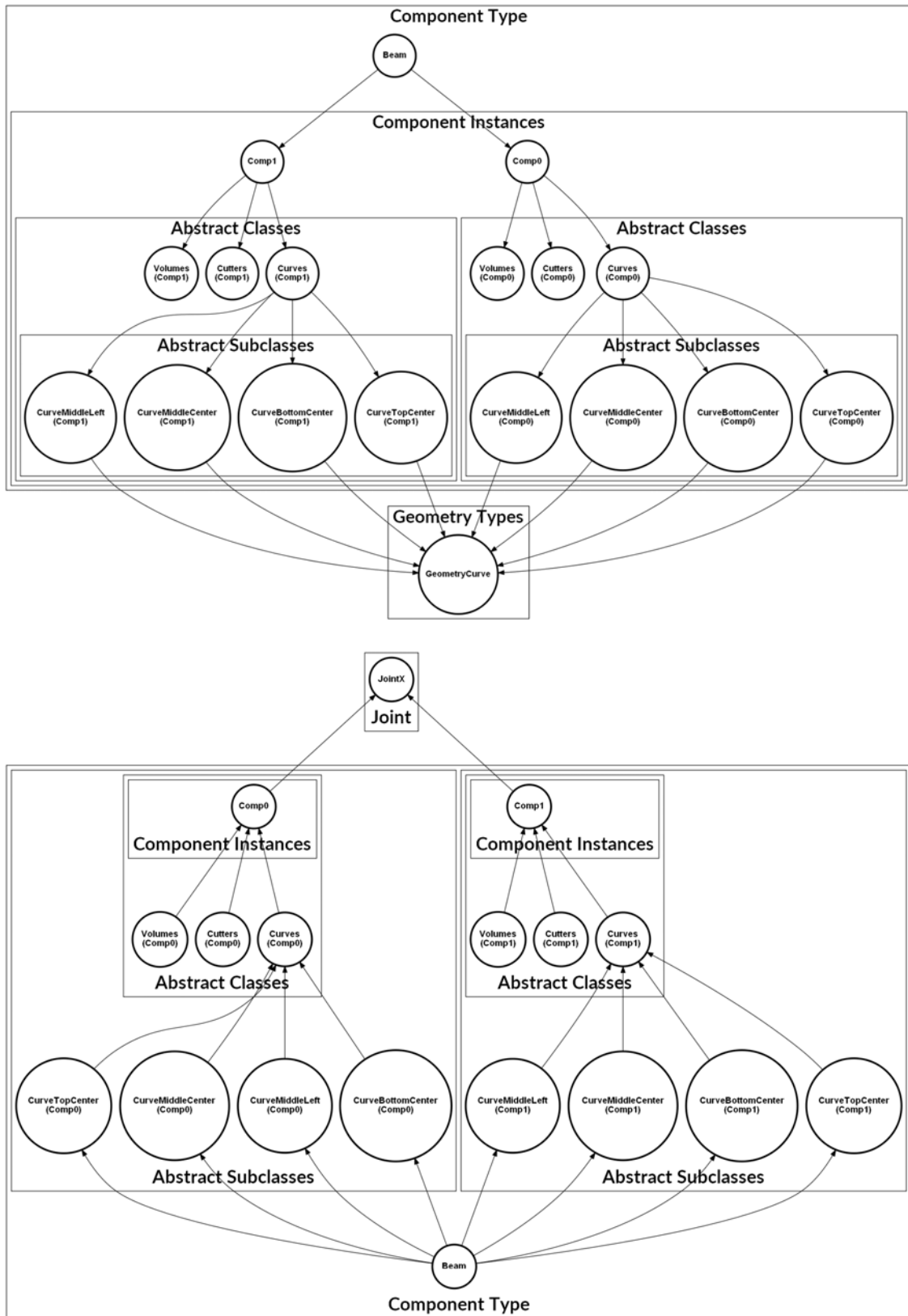


Figure 5.8: Reconfigurable data structure for different types of queries. While the above diagram showcases a query based on the object's geometry type, the bottom diagram presents component instances which are searched based on their common joint intersections.

5.1.3.2 Adaptive Queries

As seen in section 2.1.1, reconfigurable hierarchies and non-hierarchical faceted search (or tag-based search) as an alternative to traditional directory tree structures would improve the UX in its ability to search data-rich objects that do not depend on one fixed hierarchical system. This paragraph therefore describes a speculative search scenario using the organizational systems used at Design-to-Production and proposes new adaptive ways to search, filter and find information:

- **Geometry-based Query**

In figure 5.7, the geometry type is defined on the leaves of the data structure, which allows the collection of similar items based on their common geometry types (as shown in figure 5.9). This particular organizational method strongly depends on the initial Rhino3D's LayerTable that is predefined within the software. This prevents the user to operate other types of queries that would rely on other organizational methodologies. It is therefore important to either allow the hierarchies to be reconfigurable and adaptable in order to allow any kind of targeted queries. Section 2.1.1 describes precedents which might be useful in this case, such as the "Placeless Documents" system which relies on user defined tags instead of a fixed hierarchical folder system.

- **Joint-based Query**

Figure 5.8 reorganizes the hierarchy defined in figure 5.7 and places the component instances on the leaves in order to operate queries based on joint types that unite two components. This would allow the query of multiple components based on a particular joint name. For example, if one calls *Joint 1*, one *MainBeam* and two *RibBeam* objects could be collected by the user. Similarly to figure 5.9, their relationship would be displayed on the GUI and their geometry within the model's viewport.

If it is argued that queries can adapt to any kind of hierarchical systems set by the designer, it could be suggested that instead of relying predefined hierarchical system, one might actually prefer to rely on non-hierarchical and flat data visualization strategies in which all metadata is represented and from which cross-dimensional, cross-hierarchical queries are possible, such as in design explorer (Figure 2.7) developed by Thornton Tomasetti Core Studio.

Experiment B3 | Adaptive Queries
Category: design probe



[Video link to the experiment](#)

Related research question: *“How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”*

Description

The present schema-based experiment acts as a design probe – *“a design-led investigation allowing speculative inquiry, theorization and the setting out of design criteria.”* (Ramsgaard Thomsen and Tamke, 2009, 3) – and continues to follow the *accumulative* research approach described in section 3.4 and takes into account the knowledge gained in the previous experiments *“B1 | Visualizing Hierarchies: LayerFootprint”* and *“B2 | Navigating across Hierarchies: LayerExplorer”*, which explored how the expert user can better navigate through multiple levels of resolution and explore complex data sets of 3D objects.

Here, the present experiment has been focusing on the development of a [GUI](#) from which can be operated diagonal queries across complex data sets. The present interface takes as a starting point the visualization tool described above, extending its original representational role. Different functions from the RhinoCommon framework have been exposed, so that the user is able to navigate across layers by selecting/deselecting objects directly on the Rhino viewport. The different exposed commands (*ShowSelectedLayers*, *ShowHiddenSelectedObjects*, *HideUnselectedObjects* and *HideLayersFromSelectedObjects*) enable the computational designer to navigate across layers and interact with the geometrical objects on the viewport in a more intuitive way. Furthermore, layers can be displayed or hidden by typing single keywords (Figure 5.9). Their respective objects would be also displayed or hidden within the viewport. In parallel, the Sunburst diagram based on the one developed in the previous experiment *“B1 | Visualizing Hierarchies: LayerFootprint”* is overlaid with edge bundles (Holten, 2006), directly informing the user about the specific layers from which the objects have been gathered and collated.

Results

The present experiment demonstrated the possibility of performing adaptive, diagonal queries within a [GUI](#) interface that can interact directly with the 3D environment itself. However, this experiment only acted as a design probe and could therefore hardly be tested against multiple source models. Furthermore the graph itself could be hardly interacted with, as it has been built through the preview features available within the Rhino3D [API](#). The next experiment attempts to develop a similar interface in a more robust manner – with [D3.js](#) – so that it can be tested against multiple source models, interacted with and ultimately shared amongst different stakeholders.

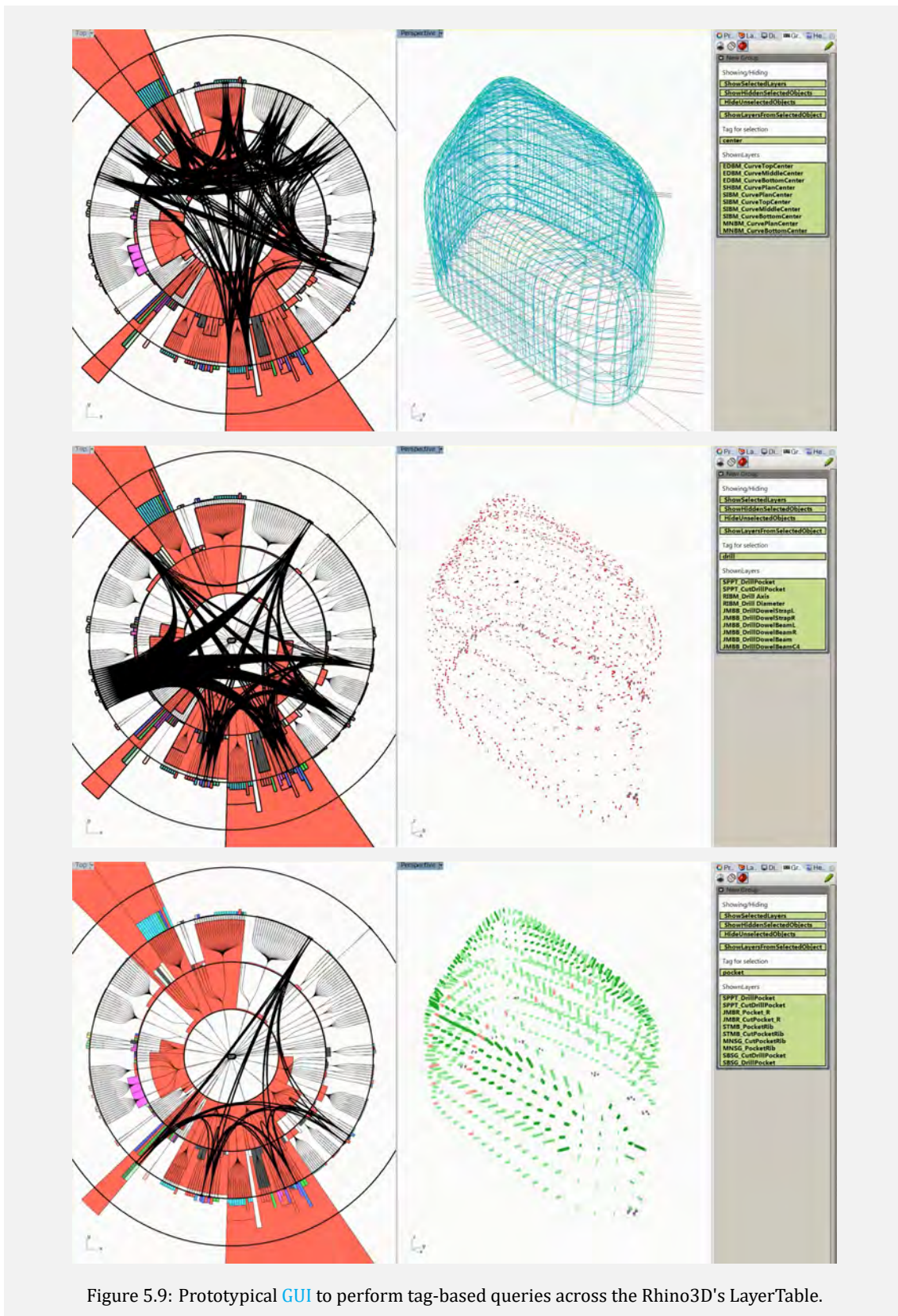


Figure 5.9: Prototypical GUI to perform tag-based queries across the Rhino3D's LayerTable.

5.1.4 B4 | Exploring Hierarchies: LayerStalker

Experiment B4 | Exploring Hierarchies: LayerStalker Category: demonstrator



Video link to the experiment

Related research question: “How would an ideal Multi-Scalar Modelling AEC-model look like and which requirements would it have to fulfill all user’s requests? How could the multi-scalar model be interacted and which User Interface (UI) and User Experience (UX) concepts would be needed?”

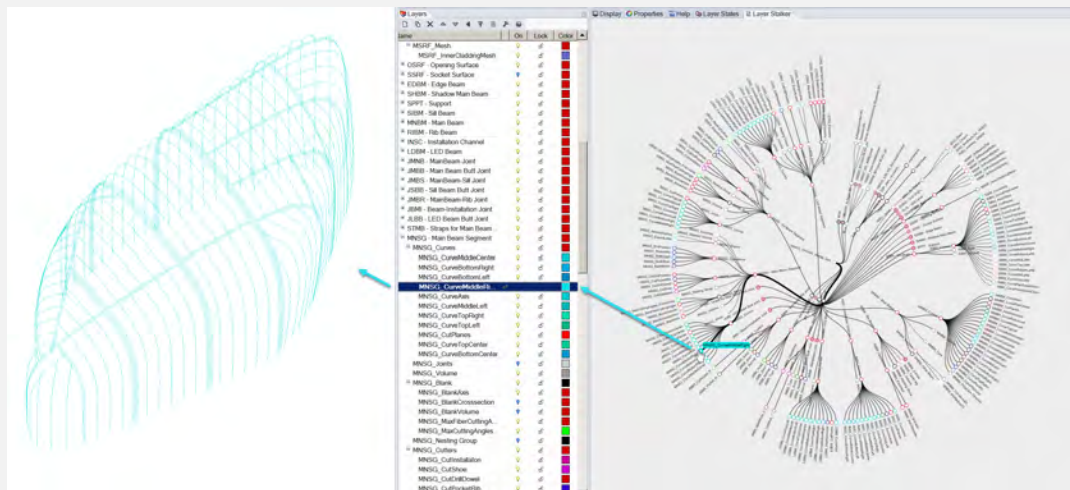


Figure 5.10: The figure represents one of the Terminal Oslo Pavilions modelled by Design-to-Production and its respective interactive LayerTable represented as a Sunburst Diagram. Here, a structured query is performed. The user is able to select one particular sublayer at a leaf of the data structure (the directory path is highlighted here by a thicker line width), displaying the contained objects within the Rhino3D viewport.

Description

The present schema-based experiment acts as a demonstrator – “[...] an application-led investigation allowing interfacing with real world problems and constraints” (Ramsgaard Thomsen and Tamke, 2009, 3) – as it bases itself upon real world data generated by Design-to-Production. The experiment continues to follow the *accumulative* research approach described in section 3.4 by taking into account the knowledge gained within the previous experiments “B1 | Visualizing Hierarchies: LayerFootprint”, “B2 | Navigating across Hierarchies: LayerExplorer” and “B3 | Adaptive Queries”, which explored so far how the expert user can better navigate and explore complex data sets of 3D objects through the custom development of GUI interfaces.

The functionality of the Sunburst Diagram described in the previous subsections acted so far as a pure data visualization tool. Its functionality has been extended by introducing interaction features within the present schema-based experiment called “LayerStalker”: an embedded custom GUI within Rhino3D. Technically, LayerStalker translates the hierarchy of the Rhino3D’s LayerTable into a JSON format that is further utilized to generate a circular

From LayerStalker, it is possible to perform both structured and unstructured queries. Whereas a structured query is operated by inserting a specific path that searches through the branches of a hierarchical directory tree structure, an unstructured query is gathering items regardless of their locations, across the branches of the same directory tree structure:

- **Structured Queries**

Figure 5.10 shows the LayerStalker interface, in which the user is selecting a specific sublayer placed at a leaf of the LayerTable's hierarchical structure by right-clicking on it. The selected sublayer's directory path is highlighted here by a thicker line width. After selection, the objects contained within this sublayer are displayed within the Rhino3D's viewport, hiding all the others.

- **Unstructured Queries**

Besides structured queries, the user is also able to perform unstructured queries, enabling the query of multiple layers based on generic strings or keywords (e.g. "detailed volume", "dowel", "drill", "axis", "connector", etc.). This is exemplified in figure 5.12 where the value "drill" is called, displaying all objects whose respective layer names contain the exact same tag, and hiding all the others. The Sunburst Diagram on the right displays the selected layers and highlights the gathered centralized information within the Rhino3d viewport, on the left. Such example highlights the fact that implicit relationships exist between objects that are situated within the unrelated leaves of the data structure. Indeed, different ways of organizing information exist, and one classification strategy might actually hide another. Therefore, objects need to be properly referenced by meaningful attributes, user strings or User Dictionaries, in order to be able to operate efficient queries later on during the design process.

Besides the possibility of operating both structured and unstructured queries, the user was able to collapse specific nodes in order to hide or show the children layers, such as in the Rhino3D's original LayerTable. As observed through the previous experiment "*B2 / Navigating across Hierarchies: LayerExplorer*", it is beneficial to integrate such feature when visualizing the entire hierarchical dataset of the LayerTable as it is easier to focus on particular branches or leaves than trying to comprehend the whole data structure as a whole.

Results and limitations

The present experiment demonstrated the possibility of performing both structured and unstructured queries from the 3D environment within the GUI interface of *LayerStalker*. This prototypical application could be deployed and stress tested against multiple Rhino3D models. Indeed, any Rhino3D model containing a complex hierarchical LayerTable could benefit from the LayerStalker application, allowing the user to operate both structured and unstructured queries. However, both of these queries depends on the name of the layers exclusively. They do not take into account other meta properties editable by the user (such as the UserData, UserDictionary or the Material properties in Rhino3D). Ultimately, it is preferable to take into account all meta properties when operating such queries. Furthermore, as this interface works exclusively with Rhino3D, the next experiments will investigate alternative strategies to *share* complex data sets across multiple models and software platforms.

5.1.5 Results, limitations and next steps

The interfaces, design probes and digital prototypes described so far in section 5.1 have been developed specifically within Rhino3D, relying exclusively on the respective software's geometrical database. However, one might speculate to generalize such approach by querying objects from a more holistic database which could gather as well data coming from external software platforms. The existing open-source tool and platform [Speckle](#) allows just to do that by storing geometrical objects coming from different software platforms within a centralized server. It has been used here to import of a complete data set generated by Design-to-Production for one of the Terminal Oslo Pavilions (Figure 5.13). It can be therefore speculated that the queries performed above could also be applied from [Speckle](#) as well. Instead of being operated locally, they would be performed directly on a server hosted by [Speckle](#), therefore available to the other trades and stakeholders involved in the design process of a particular architectural project. The concerns of interoperability and data exchange is addressed in the next section that focuses on the development of a prototypical interfaced called *SchemaBuilder*, allowing the building and transfer of nested hierarchical data structures.

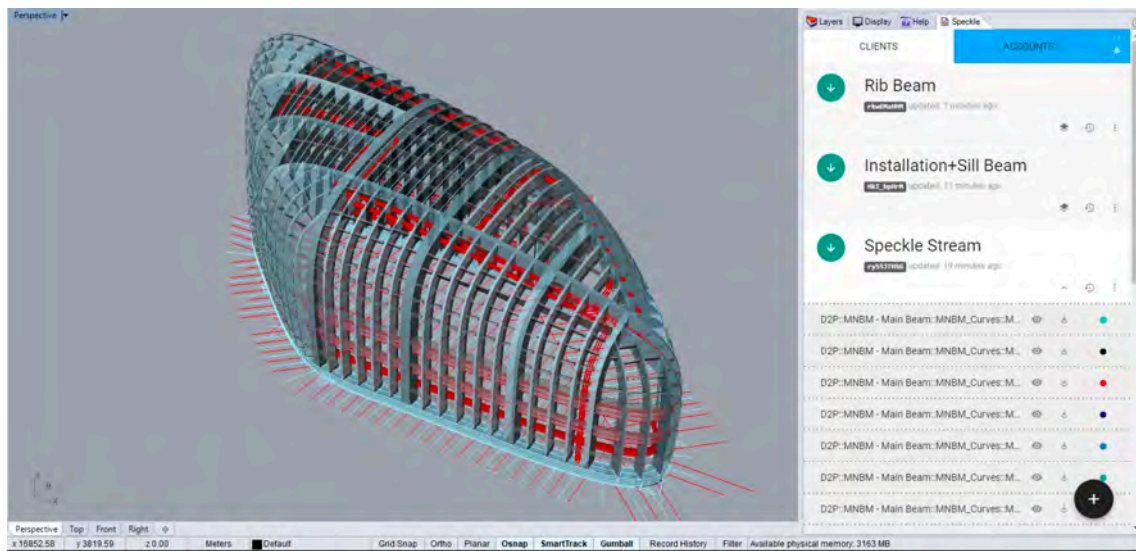


Figure 5.13: One of the Terminal Oslo Pavilions modelled by Design-to-Production is sent to a [Speckle](#) stream, visualized within the Rhino3D viewport. ([Stefanescu, 2018](#))

5.2 Building, Transferring and Receiving Hierarchies

The previous section highlighted the importance of reconfigurable hierarchies and described the potential of structured and tag-based adaptive queries through the development of prototypical search interfaces. However, the latter actually only solves specific design challenges which answer the specific needs of Design-to-Production. Indeed, not all consultancy practices work with the Rhino3D's intricate LayerTable or with the same hierarchies.

The underlying concepts investigated so far (search interfaces, adaptive queries and reconfigurable hierarchies) should be kept in mind for the broader picture of Multi-Scalar Modelling for Building Design. The objective in the present section is to genericize the methods described in the previous section by neutralizing the formats and enabling interoperability between the different trades involved in the design process. This has been tested through the development of a prototypical interface called *SchemaBuilder*.

The present section is divided into two subsection. The first subsection describes the developed prototypical interface *SchemaBuilder* ("*B5 | Building and Transferring Hierarchies: SchemaBuilder*"), and the second subsection explores strategies to receive data-rich objects ("*B6 | Receiving Hierarchies*").

5.2.1 B5 | Building and Transferring Hierarchies: SchemaBuilder

The conceptual development of the *SchemaBuilder* application partly comes from the realization that the most efficient way to exchange data between software platforms need to rely on the most neutral format as possible (Ringley, 2017). As pointed out in section 2.4.1, the Industry Foundation Classes (IFC) standard format contains today more than 1200 different object classes¹, which largely differs from the amount of HTML elements, corresponding today to only 140 different tags². Furthermore, design object flexibility and extensibility are crucial aspects to be enabled during the conception of large-scale digital workflows. This means that any data schema needs to be flexible enough to adapt its representation to any future unknown design change and stakeholder requirements throughout the design process, without however losing its core abstract definition necessary for robust interoperability and sharing across all required software platforms. Here, flexibility and schema adaptability is required over hard coded and predefined object-oriented languages (e.g. the Industry Foundation Classes). It is therefore important to remain as generic as possible by using neutral formats such as *XML* or *JSON* in order to nest and share custom hierarchical data sets. *SchemaBuilder*'s main goal is to seamlessly build custom nested hierarchies of geometries before sharing them through a cloud based application. Here, the *Speckle* communication platform for AEC has been chosen, as its open-source code facilitates the development of applications on top of it, such as *SchemaBuilder*. *Speckle* also offers at its core a database generic enough to host geometrical objects coming from various software platforms. The minimum amount of information necessary for the future reconstruction of those objects is stored along with customized user dictionaries containing parallel information defined by the user. This enables the construction of custom schemas embodying a neat hierarchy of objects and sub-objects, allowing external stakeholders to operate meaningful queries from the database at later stages. *SchemaBuilder* would prove to be extremely useful for the many consultancy practices that interact with data-rich models at late stages in the design process. Ideally, this interface should be able to process both software specific object schemas (e.g. a *RhinoObject* with its respective properties and attributes) as well as the user-defined customized schemas (e.g. the Design-to-Production's *MainBeam* objects) that usually host a multitude of different software specific sub-objects. The next paragraph describes the *SchemaBuilder*'s flowchart, which preceded the final application's implementation. It focuses on the technological Back End that needs to be anticipated before implementation and testing.

¹<http://www.buildingsmart-tech.org/ifc/IFC4/final/html/>

²<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

Experiment B5 | Building and Transferring Hierarchies: SchemaBuilder Category: demonstrator



Video link to the experiment

Related research question: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

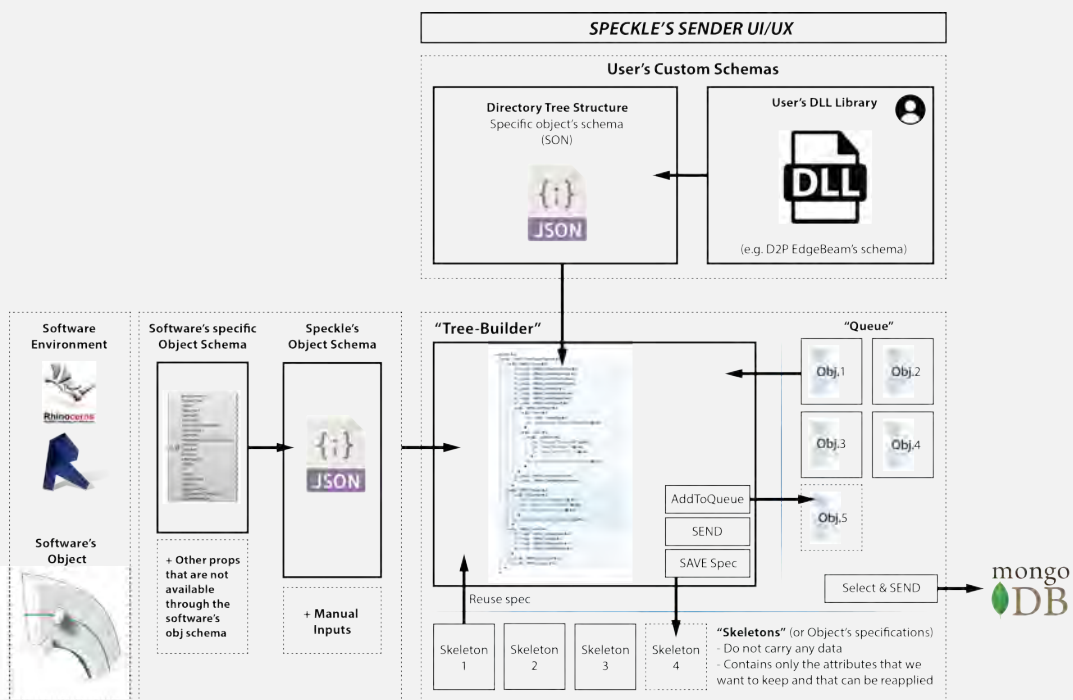


Figure 5.14: Facilitating the assembly of data-rich objects: the SchemaBuilder's flowchart.

Description: SchemaBuilder's flowchart and underlying technologies

The present modelling experiment is conducted through the *comparative* research approach described in section 3.4 and takes at points of comparison the schema-based experiments described in section 5.1. It differs from the *accumulative* logic applied so far as it focuses on the local, object level of the schema rather than on the project level (as in the previous series of experiments “B1 | Visualizing Hierarchies: LayerFootprint”, “B2 | Navigating across Hierarchies: LayerExplorer”, and “B4 | Exploring Hierarchies: LayerStalker” which investigated the overall data structure of a building model).

In the following paragraphs, the SchemaBuilder's flowchart is described, and follows the workflow diagram illustrated in figure 5.14.

First, the user selects an object located within the software environment. Then, after clicking on the “Attach Properties” button, the corresponding object's schema unfolds as a checklist from which it is possible to manually check the properties (or attributes) that need to be kept and attached to the object. The user might encounter a situation in which the attributes that needs to be attached are not directly accessible from the software object's schema. Therefore, the latter is still editable after the first generation its inherent properties, so the user can manually add other

custom ones. The interface could also embed the access to the software's related properties, if attributes non-accessible from the software object's schema reveal to be important ones.

Because specific sets of attributes might need to be reapplied onto a multitude of different objects, the user should be able to save different schema's versions embedding all previously checked/unchecked properties/attributes in order to re-apply them later onto other objects. Therefore, the **UI/UX** should also contain a "Skeleton" section where could be saved the different produced schema versions and load the saved ones to SchemaBuilder.

Finally, one must also consider the scenarios in which the user wants to embed its own custom schemas, classes and subclasses, containing at the leaves of the data structure geometrical, software specific objects. Therefore, the *SchemaBuilder* interface should also allow the user to load its own custom objects. The *SpeckleAbstract* class is allowing to do so (along with the *SpeckleCore* Converter), by serializing and deserializing standard .NET classes. Once the user has finished to build his object with *SchemaBuilder*, it would be possible to choose between two options: either sending it directly to the database or adding it to a queuing system, another part of the **UI** where nested objects would be fully saved. This would allow the user to save "Object A" on the side while building "Object B", before nesting "Object A" within "Object B". The *SchemaBuilder*'s interface constantly serializes the customized object into a *UserDictionary* or a *JSON* string that can be interpreted by the *Speckle* local converter and database, from which can be retrieved any stored custom object by external trades.

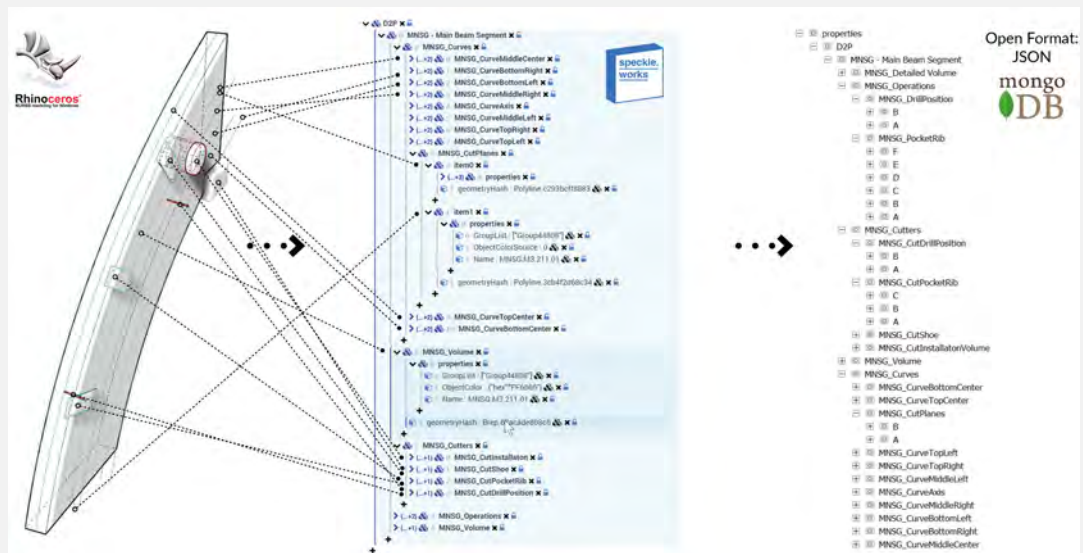


Figure 5.15: A data-rich object modelled in Rhino is assembled through the SchemaBuilder interface, before being sent through *Speckle* database relying on MongoDB.

SchemaBuilder's User Interface

In order to exemplify the above described process, a *MainBeam* object (Figure 5.15, left) created by Design-to-Production is taken as a placeholder or case study. This *MainBeam* is defined by a group of sub-objects stored across many different layers and sub-layers from the software's LayerTable, the latter acting as **UI** feature embodying the object's schema itself. This nested layer hierarchy has been translated into a Rhino3d's *UserDictionary* (Figure 5.15, center) and further converted into a *JSON* format that has been sent to the *Speckle* database, currently relying on MongoDB (Figure 5.15, right), from which specific targeted queries can be operated upon, both structured and unstructured.

To navigate across the objects schemas, the user can use the tree interface (Figure 5.16, center) that keeps track of the parent-child relationships between the objects. **UI/UX** features have been prototyped, such as highlighting the object's geometry while hovering its location within the tree, folding/unfolding branches, and pairing the objects indices between the tree

interface and the 3D environment through preview and graph-based features when selection events are being triggered. Here, the *Cutter* object, represented as a cube, is selected. It's corresponding location within the tree is highlighted as well along with its related index. The directed graph preview in the 3D viewport corresponds to the interface's tree structure at the center, allowing the user to fully understand the hierarchical relationship between the objects.

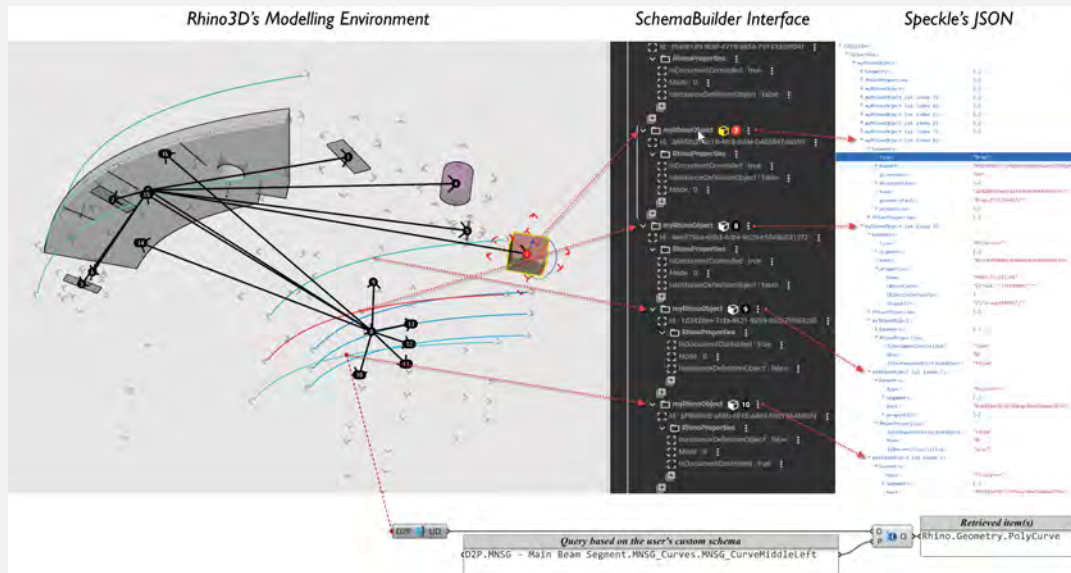


Figure 5.16: SchemaBuilder's current GUI.

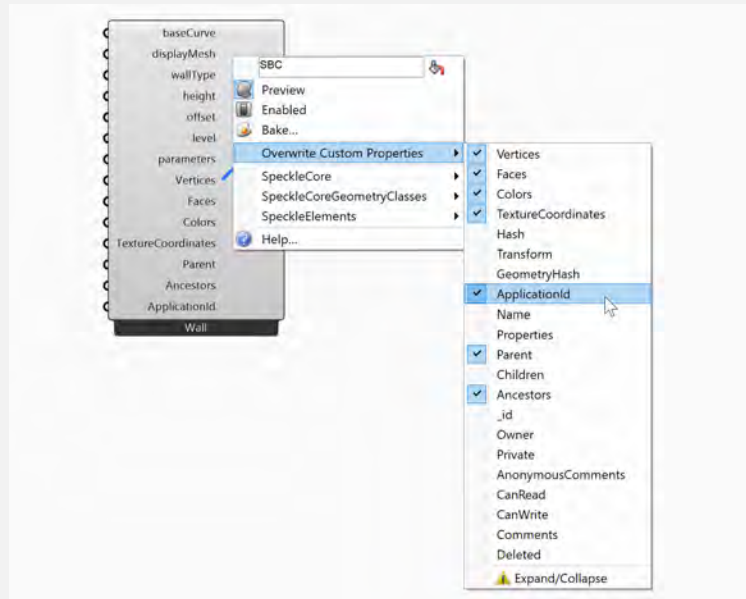


Figure 5.17: The current SchemaBuilder component as part of the [Speckle](#) plug-in for Grasshopper.

The *SchemaBuilder* interface has evolved towards the development of a specific component that is now part of the [Speckle](#) plug-in for Grasshopper (Figure 5.17). It enables the adaptive generation of any existing *SpeckleObject* type through Reflection methods. A *SpeckleObject* can either be native to the [Speckle](#) platform or further defined within a *SpeckleKit* to match specific object models coming from external software such as Revit. From the *SchemaBuilder*

component, similarly to the approach taken by BHoM, the end-user is also able to access through a tree of objects a specific `SpeckleObject` and attach/detach its optional properties.

Results

The *SchemaBuilder* prototypical application demonstrated the possibility to build seamlessly hierarchical relationships between geometrical objects, resulting in a custom schema set by the expert user. This schema could be further sent and received through [Speckle](#) streams in a flawless, file-less manner, without any loss of information. Indeed, the metadata and properties of each object were kept between the two user endpoints. Therefore, interoperability has been improved by enabling the sharing of data-rich objects between different stakeholders.

5.2.2 B6 | Receiving Hierarchies

Experiment B6 | Receiving Hierarchies

Category: design probe

Related research question: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

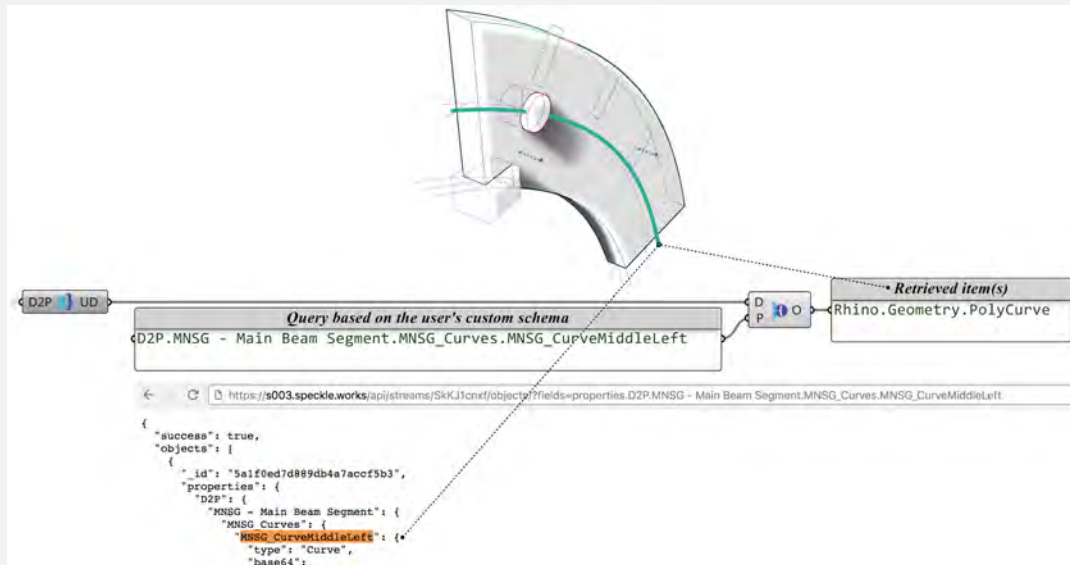


Figure 5.18: Structured query operated upon a *MainBeam* segment modelled by Design-to-Production and received through [Speckle](#).

Description

The present modelling experiment is conducted through both the *comparative* and the *accumulative* research approaches described in section 3.4. It takes at points of comparison the schema-based experiments described in section 5.1, and takes into account the knowledge gained within the previous experiment “B5 | Building and Transferring Hierarchies: *SchemaBuilder*”. Besides the building of custom hierarchies demonstrated within the previous experiment, further strategies need to be deployed in order to also handle the reception of those same hierarchies, or nested data objects. As in section 5.1.4, those data structures can be retrieved both in a structured or an unstructured manner:

- **Receiving data through structured queries**

In the first case, the user can enter a specific path corresponding to the object’s custom schema, and retrieve its corresponding geometry. Figure 5.18 shows a *MainBeam* modelled by Design-to-Production containing many different types of nested objects and sub-objects. This curve can be retrieved from the [Speckle](#) database through the operation of a structured query, such as:

D2P.MNSG - Main Beam Segment.MNSG_Curves.MNSG_CurveMiddleLeft

This type of query is possible as it follows exactly the way the company initially set up the hierarchy, such as in figure 5.7. In a speculative scenario, if Design-to-Production would like to share this specific object with an external trade, it could be possible to simply share this specific path along with the corresponding [Speckle](#) stream. An engineer that would require only the middle left curve from the *MainBeam* object could

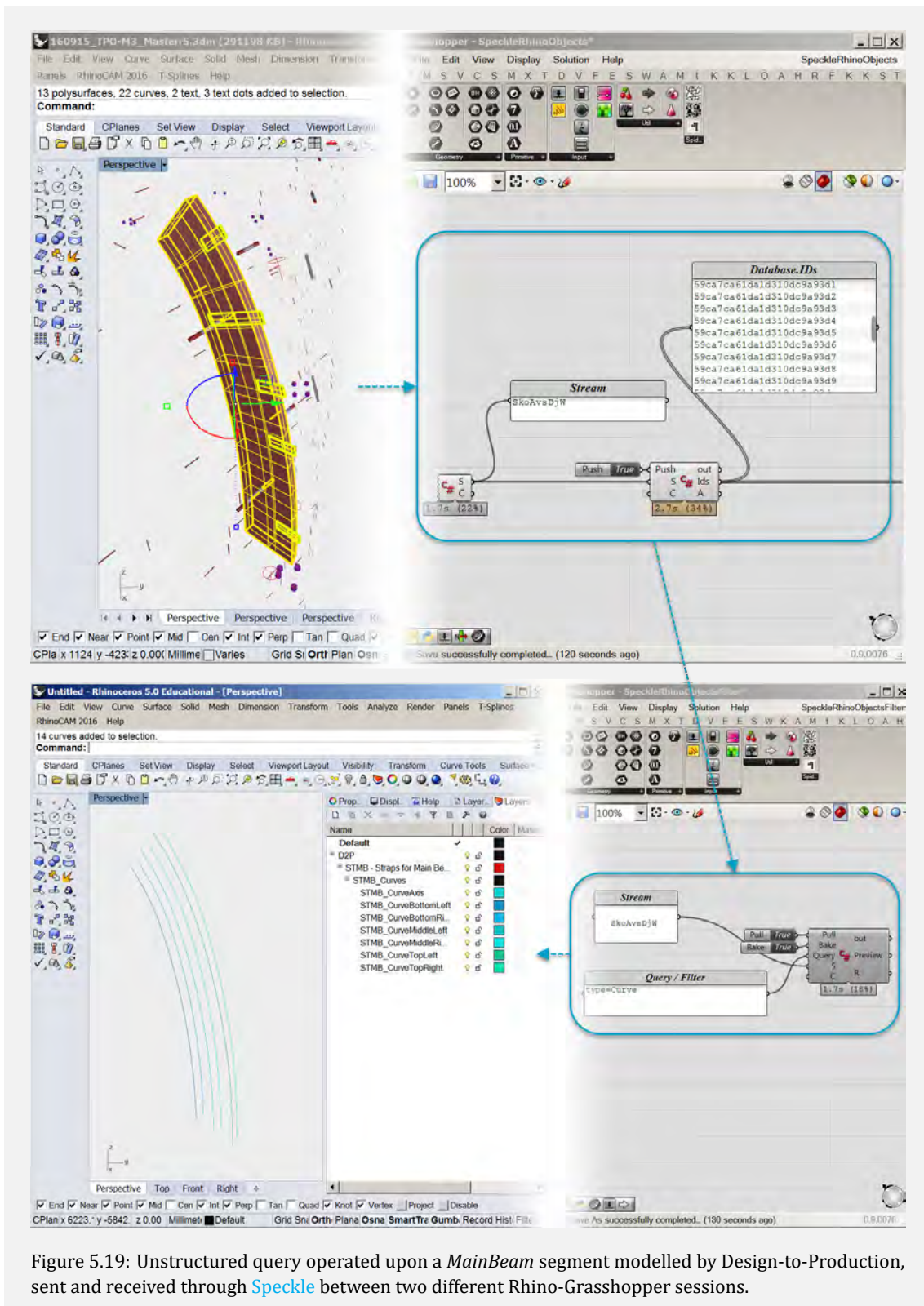
retrieve it just by typing the path above. The next paragraph focuses on unstructured query which focus mainly on the metadata and properties attached to the objects instead of there location within a hierarchical data structure.

- **Receiving data through unstructured queries**

To receive data through unstructured queries, the user can enter a specific object type or attribute to query all concerned objects. For example, figure 5.19 represents a data transfer of the same *MainBeam* segment described above, that is composed from many different data-rich object types (placed on different layers and grouped together), from its original model environment (top) to a new one (bottom) within which can be retrieved filtered information. Through custom C# nodes developed by the author and relying on the [Speckle API](#) (Application Programming Interface), all objects of type *Curve* are transferred through a [Speckle](#) Stream and re-generated along with their respective attributes (*Group*, *Color*, etc.). Furthermore, their hierarchical placement within the original LayerTable has been conserved as well.

Results

The present design probe consisted initially in a custom C# script which evolved towards the development of the *"Gets Nested Value (GNV)"* Grasshopper component allowing targeted structured queries from any object containing nested dictionaries. This can now be achieved by inputting a string of concatenated keys, such as in figure 5.18. Furthermore, the [Speckle](#) interface as since integrated the option to perform unstructured queries directly through the stream's URL. Both these structured and unstructured queries are technically enabled by the [Node.js "query-to-mongo" package](#).



5.2.3 Conclusion and future directions

The different schema-based experiments described so far focused on the local adaptive object schema (“B1 | Visualizing Hierarchies: LayerFootprint”, “B2 | Navigating across Hierarchies: LayerExplorer”, “B4 | Exploring Hierarchies: LayerStalker”) and its transfer between different modelling environments (“B5 | Building and Transferring Hierarchies: SchemaBuilder”, “B6 | Receiving Hierarchies”). Merging together these different concepts would enable cross-practice collaboration at scale, in which multiple involved parties can seek to exchange multiple complex data schemas and (sub-)models throughout an entire project.

After having manipulated complex data sets across platforms, the last section of this chapter attempts to investigate further the potential of the different findings and methodologies developed in this chapter (navigating across hierarchies, operating structured and unstructured queries), by applying and testing them through a series of workshop and case studies. The Multi-Scalar design methods developed and deployed in chapter 4 will also be used and associated with the ones from this chapter, by taking into consideration parametric modelling workflows, adapter and skeleton models as well.

5.3 Cross-Practice Collaborations: Workshops and Case Studies

The previous sections described different prototypical interfaces, methods and workflows to visualize, cross-reference, build, explore, query and share schemas. All those methods could enable more transparency in the design process and therefore improve interoperability between the different trades working together on a same architectural project. Therefore, this last section looks more specifically at cross-practice collaboration and co-creation strategies, enabling communication flows between multiple trades. Contrary to the term “interoperability” which primarily focuses on the technical aspect of sharing information between two software platforms, the term “collaboration” is emphasized here by the authors’ to highlight the important human factor in addition to the technological, in sharing tailored design objects.

The next modelling experiments carried in this section have been conducted following the *probing* research approach described in section 3.4. Generally, the *probing* approach exploits “opportunities and [explores] design ideas as they emerge through [previous] design work.” (Krogh et al., 2015). In this aspect, the different workflows methodologies deployed here borrow concepts investigated both in the previous section (schema-based experiments) and in the previous chapter (Multi-Scalar Modelling strategies and connected pipelines).

Technically, most of the developed experiments described in this section rely on [Speckle](#), the extensible Design & [AEC](#) data communication protocol and platform described in section 2.4.3. The latter has been chosen because it does not enforce any kind of object-oriented paradigm. It is also customizable at will, a very valuable quality enabling an easy way to investigate, test and deploy the different digital demonstrators detailed here.

5.3.1 C1 | Sharing Schemas through **Speckle** and **BHoM**: a Speculative Design Workflow between Design-to-Production and BuroHappold

Experiment C1 | Sharing Schemas through **Speckle** and **BHoM**: a Speculative Design Workflow between Design-to-Production and BuroHappold
Category: demonstrator



Video link to the experiment

Related research question: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

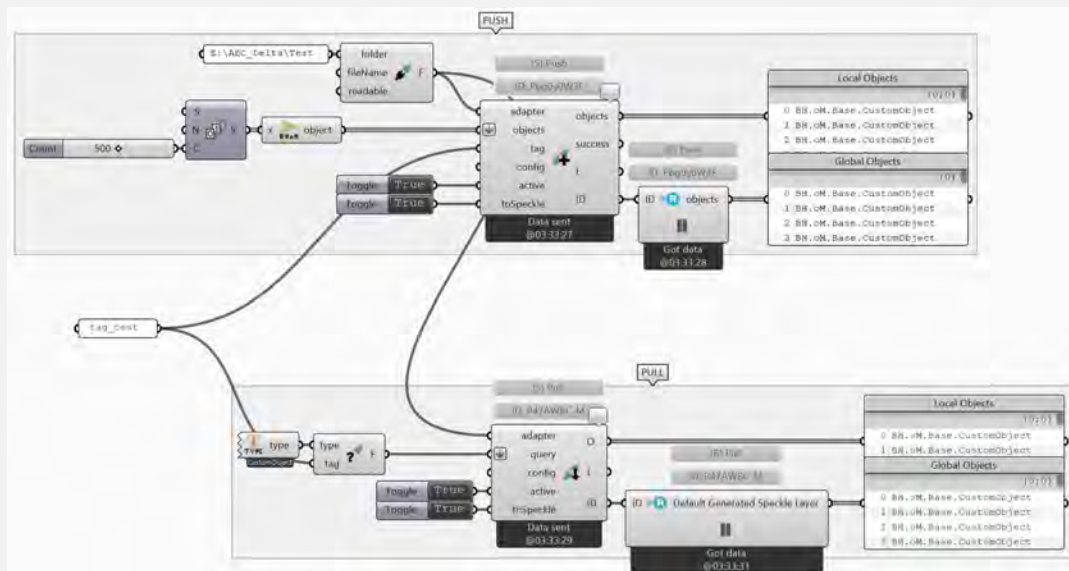


Figure 5.20: **Speckle**-**BHoM** integration prototype.

Description

The present cross-practice collaboration experiment acts as a demonstrator – “[...] an application-led investigation allowing interfacing with real world problems and constraints” (Ramsgaard Thomsen and Tamke, 2009, 3) – and is conducted through the *probing* research approach described in section 3.4, by hybridizing the schema-based strategies developed in section 5.2 and the modelling-based methodologies developed through the modelling-based experiments developed in chap 4. To demonstrate the potential of the previous experiments within a collaborative scenario, the following speculative experiment attempts to integrate the communication and collaboration processes between Design-to-Production and BuroHappold through schema-based workflows, using a specific example of a timber assembly modelled by the first company. Furthermore, this experiment looks at the differences between the **Speckle** and **BHoM** platforms.

One of the main differences between the **Speckle** and **BHoM** platforms lies in the fact that the former provides a server architecture enabling the transfer of object “buckets” through streams and HTTP requests, whereas the latter operates locally through software’s API calls and by storing the objects directly within MongoDB databases. The following developed

prototype described here aims at merging the two paradigms together in order to both benefit from the global server architecture provided by [Speckle](#) to record an history of the design processes and from the frictionless, latency-free local data exchange protocols provided by [BHoM](#). The prototype has been successfully implemented (see figure 5.20) and consists of a custom Grasshopper component which merges the “Push” [BHoM](#) Grasshopper component with the [Speckle](#) Sender Grasshopper component. Such integration enables the end-user to send custom objects both locally through the [BHoM](#) “Push” as well as globally through [Speckle](#) by setting an extra toggle “toSpeckle” to True. Based on this approach, another Grasshopper component has been developed, merging this time the “Pull” [BHoM](#) Grasshopper component with the [Speckle](#) Sender Grasshopper component. In the long term, the general idea is to enable the recording of history of the design processes undertaken by the end-users through any [BHoM](#) adapters (Push, Pull, Execute, Delete, Move) that would all be integrated with an optional [Speckle](#) Sender protocol. End-users would be able to go back in the history of the project’s design process and retrieve crucial information that has been previously manipulated.

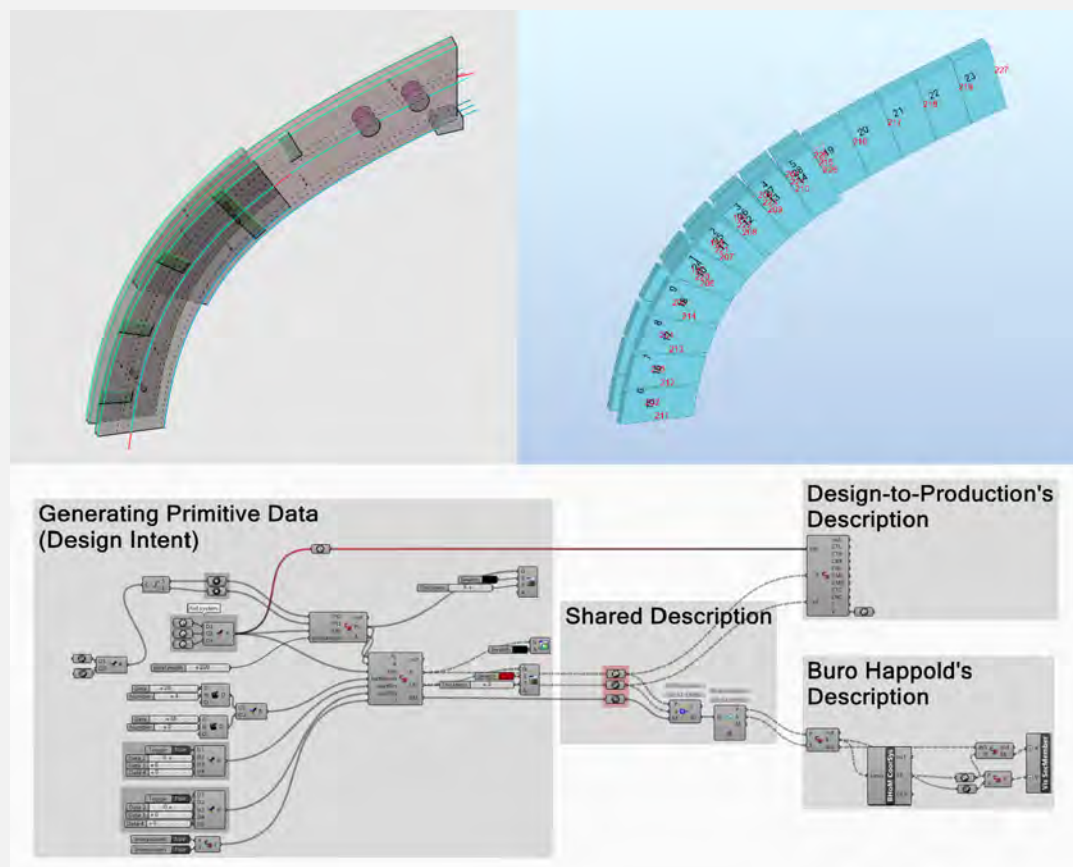


Figure 5.21: A Grasshopper definition demonstrating a digital design workflow between the consultancy practice Design-to-Production and BuroHappold. First, the data is generated by Design-to-Production within Rhino3d (left) and shared through a common schema via [Speckle](#) to BuroHappold, from which can be retrieved just the necessary information to run a structural analysis within Robot (right) using their [BHoM](#) interface.

Case study

In collaboration with the two industrial thesis supervisors (Design-to-Production and BuroHappold), the present speculative experiment has been developed in order to assess how far communication processes can be enhanced between different companies through

schema-based workflows, using as an example a specific timber assembly. In this experiment, a common schema has been shared between the two practices that are working towards different ends: Design-to-Production is generating full geometrical description of each architectural component and BuroHappold is mainly focusing on obtaining precise structural analysis results. To perform the latter, BuroHappold is using the [BHoM](#) (Buildings and Habitats object Model) platform, allowing seamless data transfer from the Grasshopper canvas - considered here as a main [UI](#) interface of the [BHoM](#) - to the Autodesk's Robot Structural Analysis software package. The retained common schema has been discussed by both trades, which agreed on the minimum information necessary to generate both of their respective data: here a series of different planes informing on the directionality of the beam assembly and allowing both its geometrical generation and further structural analysis. This series of planes can be seen as a Skeleton Model (as seen in section 2.5.3) enabling further interoperability. This scenario enables a bi-directional workflow from which data can be transferred seamlessly between the respective working environments of the two companies (Figure 5.21).

Figure 5.22 shows the object schemas unfolding within each company, as the one from Design-to-Production previously described (Figure 5.7). It can be observed that although the modelling structure and logics employed by each company differs, establishing specific points of contact, or Skeleton Models – here, the beam's common end points and its series of planes – enable seamless communication channels from which the two companies can start communicating crucial information for both modelling practice and structural analysis purposes.

Results, limitations and future directions

This cross-practice collaboration experiment combines modelling-based strategies (as described in chapter 4) and schema-based workflows (as described through the previous experiments in the present chapter) to transfer data-rich geometrical information across trades and scales. Figure 5.23 illustrates here a back-propagation behavior that is triggered once the common schema is shared and updated via [Speckle](#). The present experiment hybridizes the modelling-based and schema-based strategies, as part of the overall Multi-Scalar Modelling framework.

Although successful, the multiple schema-based experiments described above in this section focused on the local adaptive object schema (*SchemaBuilder*) and its transfer between different modelling environments (as demonstrated through the present experiment). Merging these concepts with the modelling-based strategies developed in chapter 4 would enable cross-practice collaboration at scale, in which multiple involved parties could seek to exchange multiple complex data schemas and (sub-)models throughout an entire project. This principle was explored during the 2018 Simulation for Architecture + Urban Design (SimAUD) workshop taught with Dimitrie Stefanescu, described in the next section. The workshop aimed at introducing the [Speckle](#) open-source framework and focused on open collaborative design and modelling workflows.

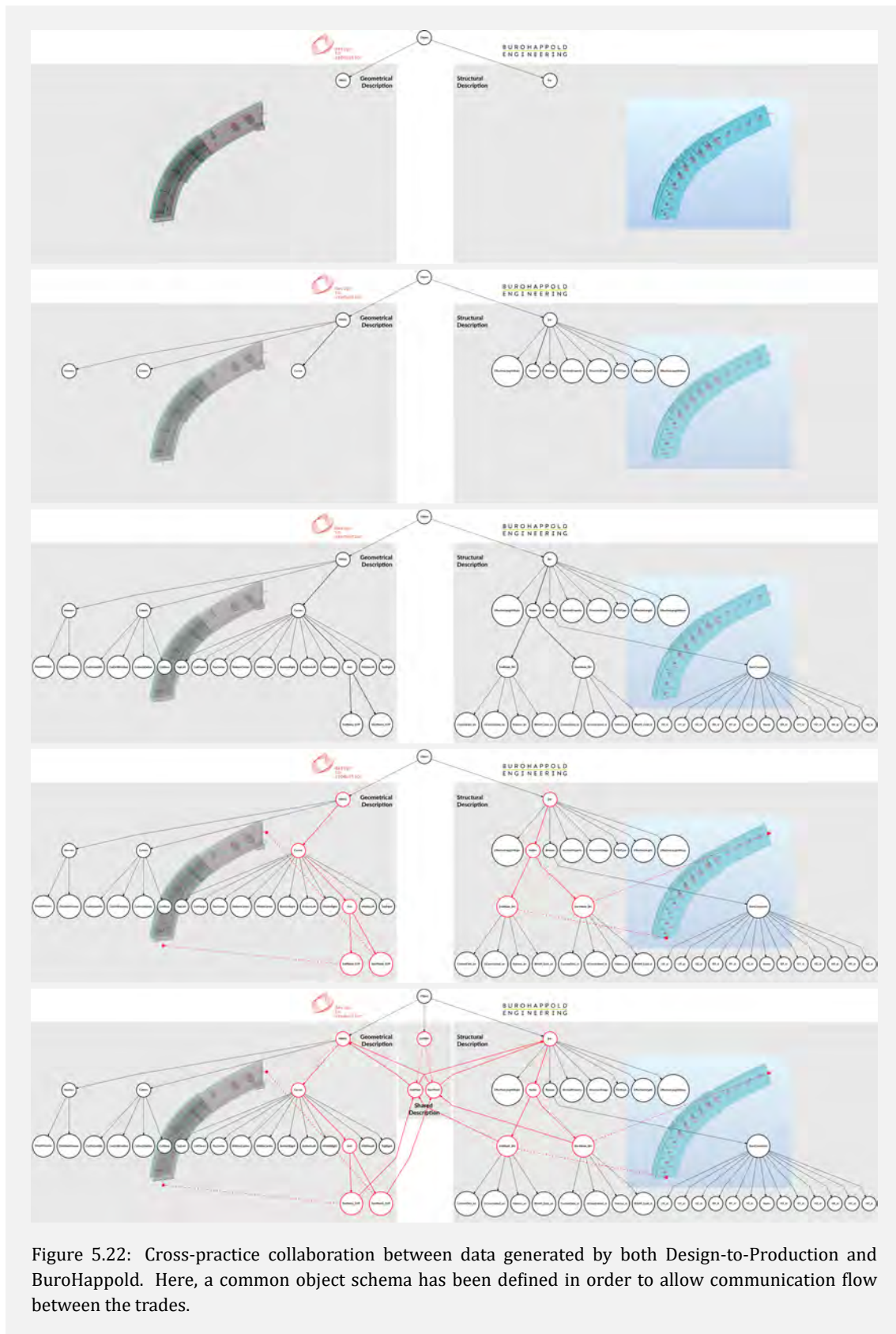


Figure 5.22: Cross-practice collaboration between data generated by both Design-to-Production and BuroHappold. Here, a common object schema has been defined in order to allow communication flow between the trades.

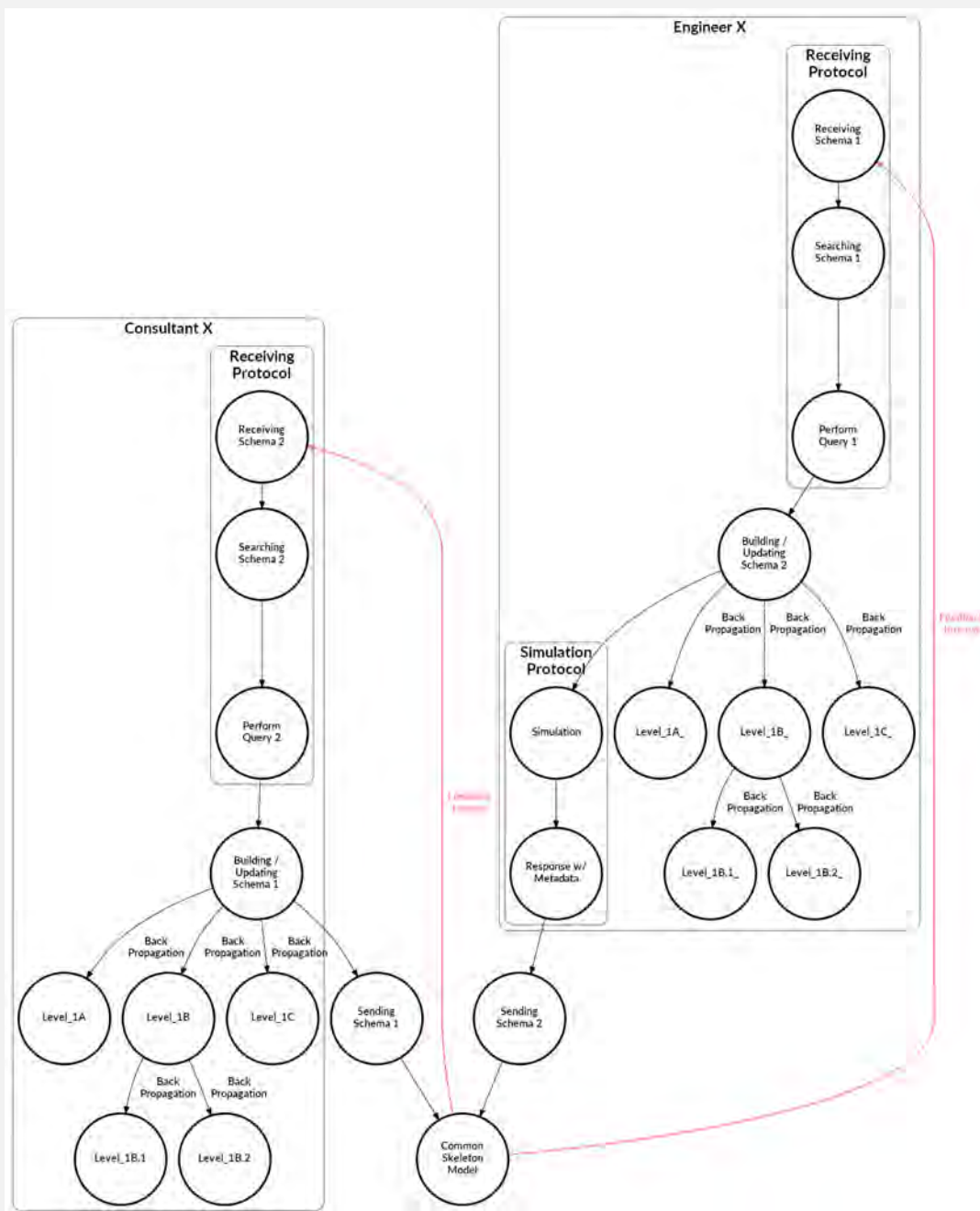


Figure 5.23: Cross-practice collaboration and back-propagation.

5.3.2 C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows

Experiment C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows
Category: demonstrator



Video link to the experiment

Related research question: *“How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”*

Description

The present cross-practice collaboration experiment acts as a demonstrator – “[...] an application-led investigation allowing interfacing with real world problems and constraints” (Ramsgaard Thomsen and Tamke, 2009, 3) – and is conducted through the comparative research approach described in section 3.4. It takes at point of comparison the previous cross-practice collaboration experiment “C1 | Sharing Schemas through *Speckle* and *BHoM*: a Speculative Design Workflow between Design-to-Production and BuroHappold”.

This workshop, based on the *Speckle* Design Data Exchange Protocol developed at UCL within the InnoChain H2020 programme, aims to stage a high frequency iterative digital design workflow between several stakeholders roles in an architectural office: design, analysis and simulation reporting. After an initial introduction to the *Speckle* communication platform, a parametric modelling workflow of a free-form timber structure developed within the Grasshopper interface has been segregated into eight different smaller pipelines (Figure 5.24, top) distributed amongst the eight different participants: (1) Global Network Control, (2) K2 Geometrical Optimization (angle maximization between the members), (3) Radius Control and Maximization, (4) Master Surface Control, (5) Blank Mesh Generation, (6) Volume Mesh Generation, (7) Lamella Mesh Generation, (8) Result Overview. Each participant had control over one of these design spaces. When satisfied with its own design criteria, the user had to communicate the output of its own pipeline to the next design actor, which also had control over its own design space, and so on. This method resulted into a distributed design chain across all the participants, forming the overall design workflow. *Speckle* Senders and Receivers were used to seamlessly share design data between each pipeline. At each of these points of contact and *Speckle* interfaces, the user was able to attach metadata (node index, bending radius, etc.) to the shared geometry’s Skeleton Model, similarly to the shared common descriptions between Design-to-Production and BuroHappold described in the previous section. Furthermore, the geometrical data was gradually streamed on the *Speckle* viewer at each transaction (Figure 5.24, center). The streams could also be aggregated and displayed within a common viewer accessible by all participants. Each time that pipelines were connected through the object streams, the overall workflow’s map (or the connections between the different pipelines) could also be progressively visualized through a more global, higher level graph (Figure 5.24, bottom), in a very similar way to what Woods Bagot developed with its own *Metagraph* viewer (Figure 2.29). The present graph visually informs on the workflow’s complexity by thickening each of its nodes according to the number of Grasshopper components situated within each pipeline. Although the nature of the graph described here is directed and acyclic, it would be theoretically possible to implement feedback loops (based on design criteria or structural analysis inputs), transforming therefore these linear processes into non-linear connected pipelines (or non-linear workflow), as described in Transmissive

Assemblies with Discrete Event Simulation strategies (Figure 3.11).

This *Metagraph* is key concept here, keeping track of both the object, process and crucially human dependencies that exist, adapt and change throughout the evolution of a project.

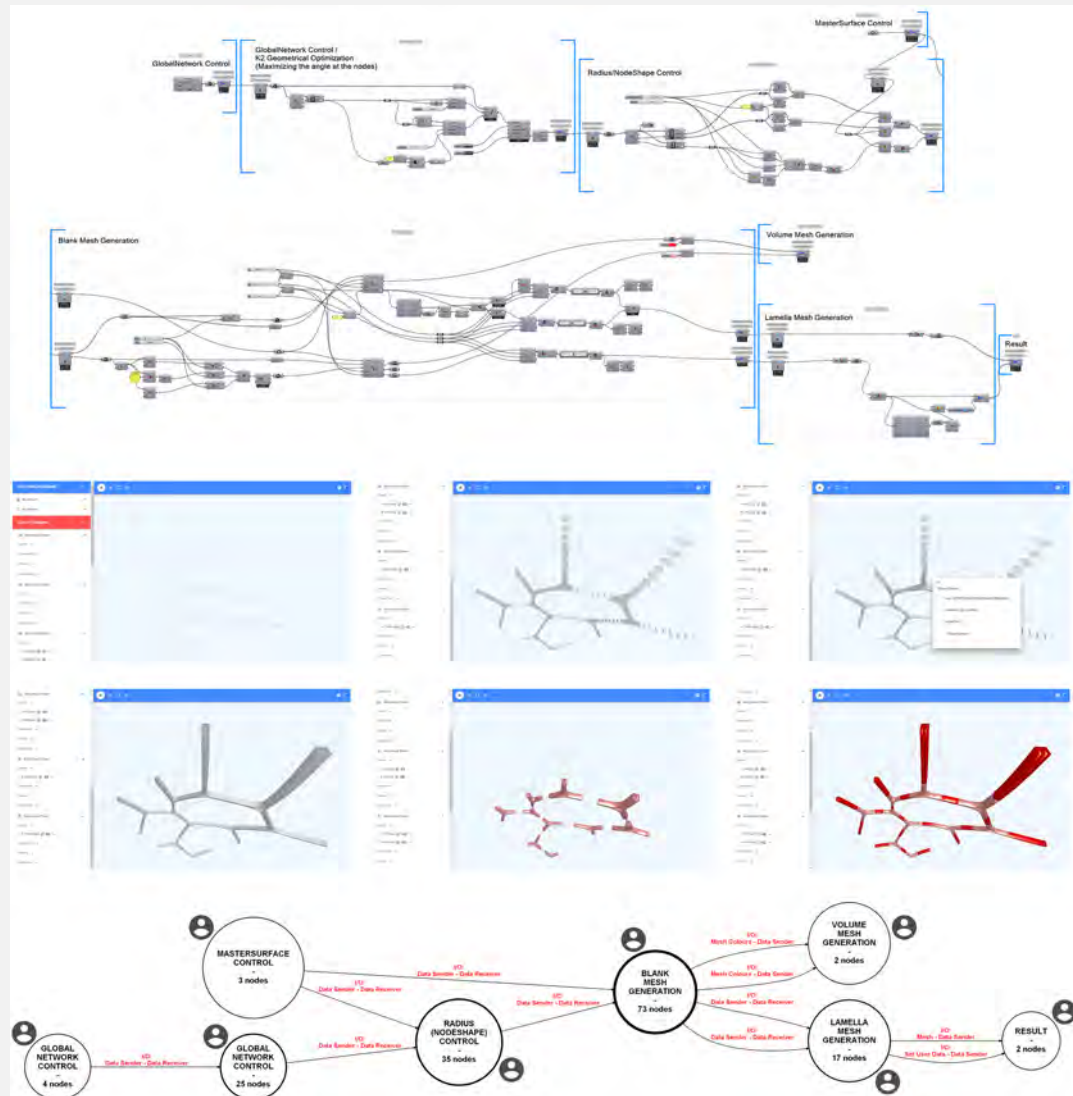


Figure 5.24: This diagram represents the meta-graph of a digital design workflow, and is based on the existing connections between the groups (or modelling pipelines) which communicate through [Speckle](#) Senders and [Speckle](#) Receivers. This allows the user to get a better overview on the overall design process of a building project. In such framework, it would be possible to introduce feedback loop processes that could trigger again parent modelling pipelines, based on different design or structural criteria set by the designer/user.

Results

This cross-practice collaboration experiment demonstrated the possibility of deploying a shared workflow setup across all the workshop's participants, enabling them to communicate in a seamlessly manner across their design spaces. Although segregation proves to be useful to separate the concerns ([Dijkstra, 1982](#)) in architectural design practice ([Pena De Leon, 2014](#)), it does not mean that communication should be disabled. Instead, communication channels should be maintained with the possibility to enable them whenever the design users wish to stream data during the design process.

As in the modelling experiment “WS3 / *Exquisite-Timber-Corpse Workshop: Collaborative Practice*”, this cross-practice collaboration experiment also demonstrated that the human factor cannot be neglected in favour of a purely automated technological paradigm, needing to be strongly considered throughout all aspects of the design process. In order to enable mass participation in design at scale, co-creation and co-authorship behaviours need to be engendered, both in terms of project architecture and team competencies.

Future work

In the context of the AEC Delta Mobility project introduced in section 6.6, future work has been carried in order to implement the *Metagraph* mentioned above. Through such graph (available within the [Speckle](#) project interface), the user is able to access the streams contained within a specific project, navigate across them, select a specific one and visualize it within the [Speckle](#) viewer.



Figure 5.25: Implementation of the metagraph within the [Speckle](#) admin interface. Here the user can have access and visualize all the streams contained within a particular project, and visualize the object contained within specific streams.

5.3.3 C3 | Piped Assemblies Workshop at CEDIM

Experiment C3 | Piped Assemblies Workshop at CEDIM Category: demonstrator

Related research questions: “How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?”

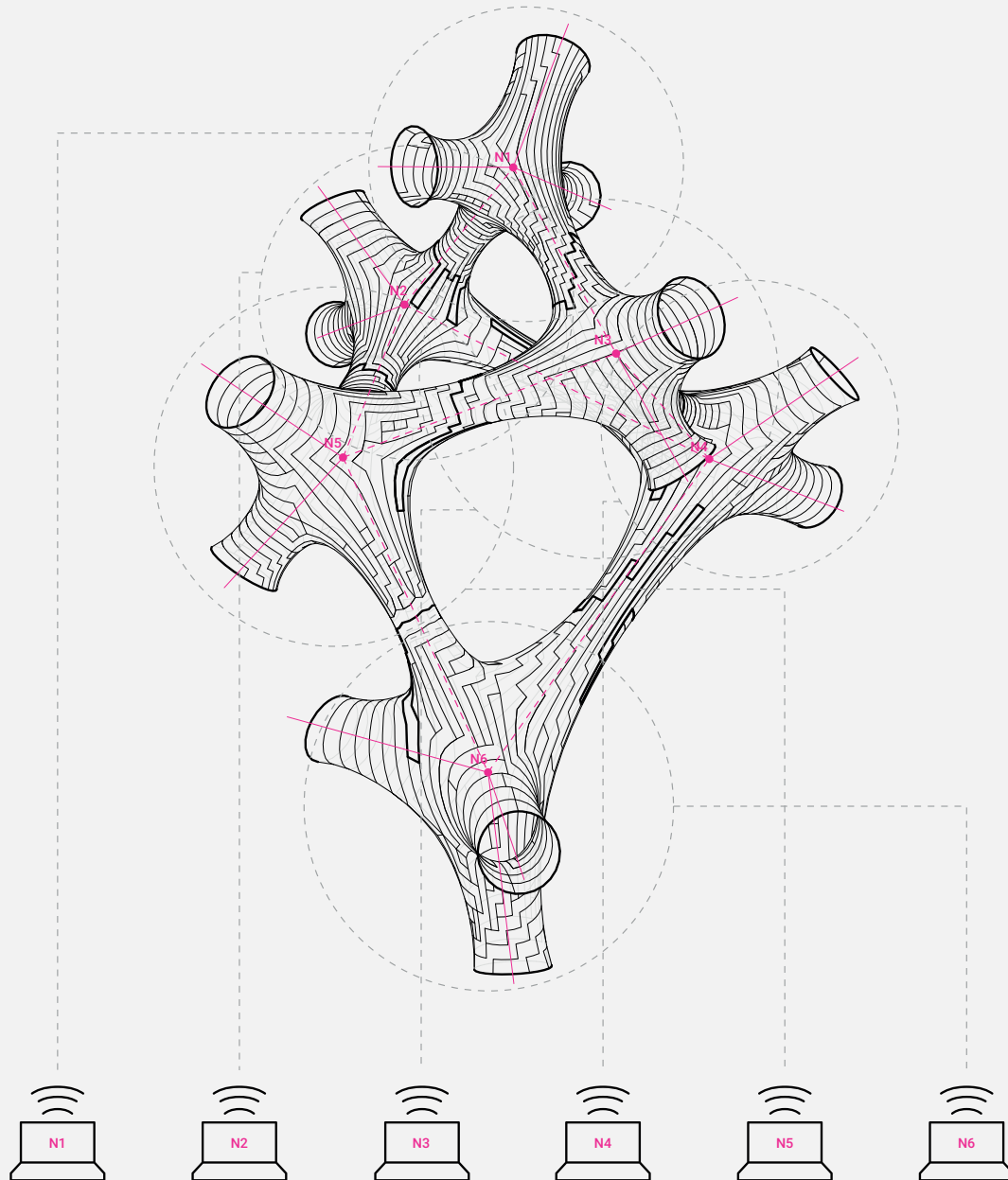


Figure 5.26: After discretization and rationalization, the different nodes of the global structure have been streamed and spread across multiple machines and designers, each one of them focusing on the fabrication workflow of a particular node.

Description

The present cross-practice collaboration experiment acts as a demonstrator – “[...] *an application-led investigation allowing interfacing with real world problems and constraints*” (Ramsgaard Thomsen and Tamke, 2009, 3) – and is conducted through the *comparative* research approach described in section 3.4. It takes at point of comparison the previous cross-practice collaboration experiment “C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows”.

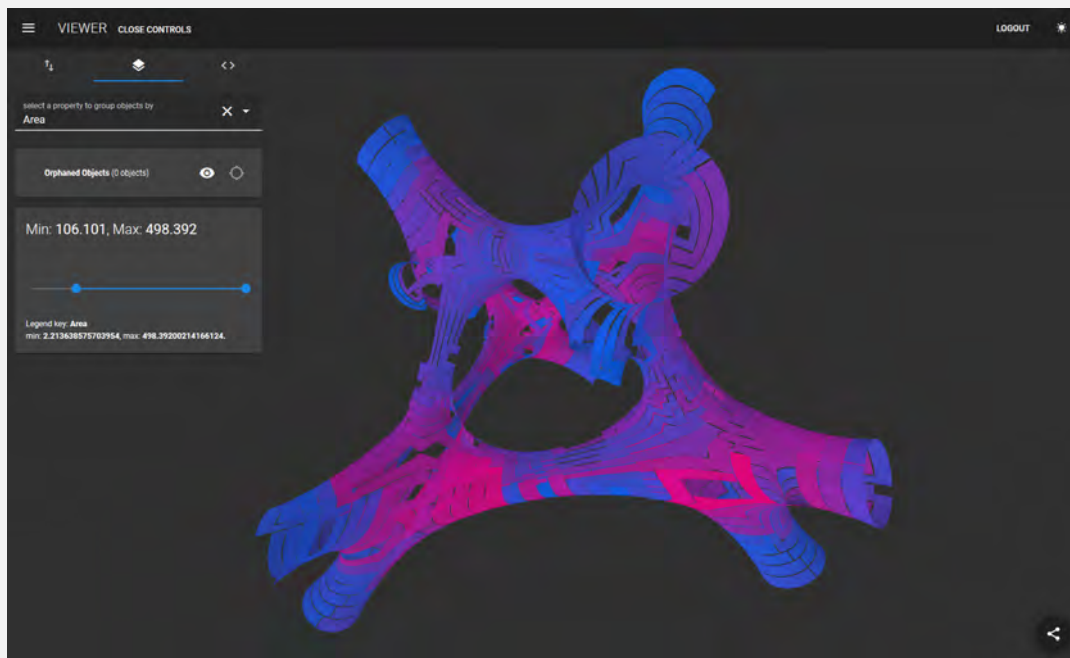


Figure 5.27: Visualization of the Piped Assemblies structure within the [Speckle](#) viewer.

The present cross-practice collaboration experiment describes the Piped Assemblies workshop conducted at CEDIM (Centro de Estudios Superiores de Diseño de Monterrey) from the 26th of November to the 7th of December 2018, which took the challenge of introducing state of the art computational design workflows to undergraduate students in architecture, through the design and fabrication of a free-form structure made of laser-cut polypropylene plastic strips. The first week was dedicated to the prototyping of a single node component and the design of the final installation. During the second week, 2 days were spent on laser-cutting the strips and the 3 remaining days were devoted on the construction, assembly and installation of the final structure. The main goal of the workshop was to set up an integrative, file-less collaborative design workflow using the Rhino-Grasshopper environment as the main modelling platform. The workflow was segregated into different pipelines (or local Grasshopper definitions) shared between students and connected through the open-source [AEC](#) communication platform [Speckle](#) during all phases of both the conception and construction processes of a free-form architectural design installation. [Speckle](#) allows the data to be streamed, shared and visualized within a web server from which data can be both sent and received. In the case of the Piped Assemblies workshop, the [Speckle](#) Client (or plug-in) for Rhino-Grasshopper was used to send and receive data to the server. The overall free-form structure is built from 502 unique developable polypropylene plastic strips connected with rivets, and occupies a volume of 12.5 cubic meters (2.12h*2.51*2.35L). Its aesthetics and morphology (local high curvature) is strongly inspired by the existing state of the art of free-form and organic installations designed and built by both the architectural design practice TheVeryMany (Marc Fornes) and Jens Pedersen’s yearly AA Visiting School workshops in Aarhus. The strip discretization of the structure finds also inspiration in prior research and

graph-based computational design tools developed by both (Nejur and Steinfeld, 2016) and by Anders Holden Deleuran - the implementation of the *NetworkX* library in ghPython and the exposition of its main functionalities.

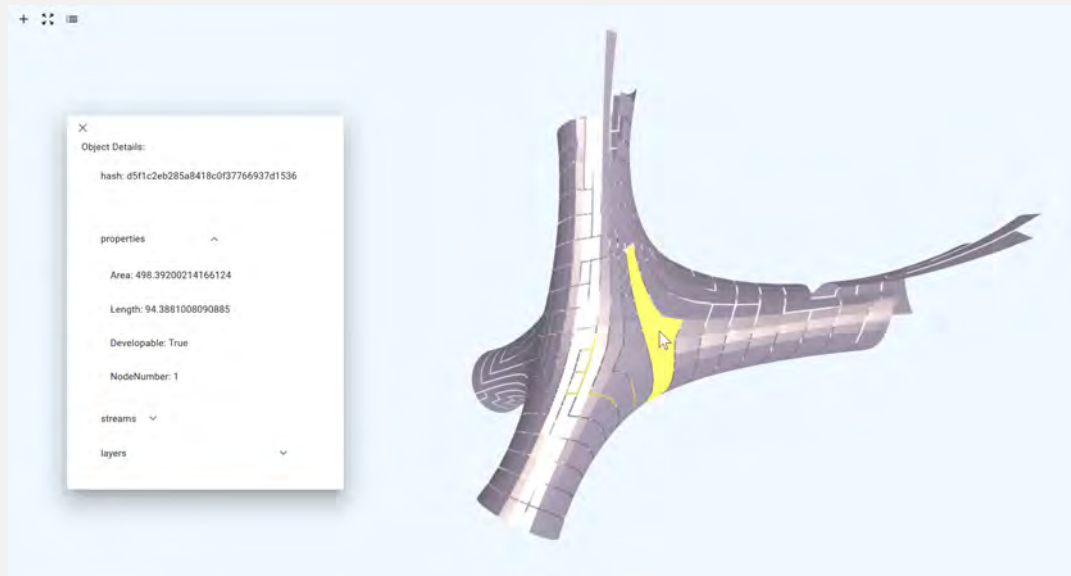


Figure 5.28: A specific node is displayed with its related metadata within the [Speckle](#) viewer.

Design-to-Fabrication workflow

The design-to-fabrication workflow of the Piped Assemblies workshop has been segregated into 7 distinct pipelines (or steps) which are all connected through [Speckle](#) streams: (1) definition of the overall abstract network, (2) base mesh generation and stitching automation between each node, (3) mesh relaxation and planarization (Kangaroo 2), (4) strips generation algorithm based on a mesh-walk method developed with the python library *NetworkX*, (5) labelling, unrolling and sorting of the strips, (6) nesting of the strips (OpenNest) and tab generation, (7) export of the fabrication data to .dxf files for machine-ready deliverables and wireless transfer to laser-cutting (STMRobotics). The steps (1), (2) and (3) were particularly crucial to generate a design which had to be consistent enough to be fabricated and assembled with differentiated polypropylene strips. The mesh resolution, the curvature relaxation and the planarity of each mesh face all had to be taken into account to evaluate the feasibility of the design. Furthermore, as all the nodes were all topologically different (various degree of valence), the number of faces and vertices could strongly differ at the meeting point between two arms, causing mesh topological issues. To solve this problem, stitching design methodologies had to be developed in order to deliver a consistent mesh topology from which the strips could be generated. Step (4) focused on the generation of the mesh strips. Here, a mesh-walk algorithm has been developed, based on a previous implementation of the *NetworkX* library within the RhinoCommon framework by Anders Holden Deleuran. Combined with ghPython, *NetworkX* allowed to convert the global mesh into a global graph, from which various analysis could be performed. The main one used for the fabrication of the structure was the Dijkstra's Shortest Path First algorithm, finding the shortest path between two mesh face center points situated onto the mesh. Although not perfectly geodesic, such path tends to minimize the curvature of the mesh strip. This particular design process was iteratively computed through a feedback loop: after the generation of one strip, the latter was removed from the global mesh, and the Dijkstra's Shortest Path First was reapplied on the remaining global mesh. The feedback loop stopped once the global mesh was completely transformed into individual strips. To unroll each strip in step (5), one has to make sure that the respective geometries are developable. Although the strips are exclusively composed of mesh quads that

have been planarized through the K2 relaxation in step (3), those same quads have also been triangulated to overcome tolerance issues and fully ensure the developability of each one of the strips. After labelling and unrolling the strips, those were nested using the OpenNest plug-in for Grasshopper, and distributed amongst 12 boards of 122*244 centimeters. Finally, the tabs for the rivet connections were generated by means of offsets from the middle point of each strip mesh naked edge.

From the fifth pipeline, the overall workflow grafts into six different parallel branches as the students have to organize themselves into six different teams in order to fabricate separately the six different nodes of the overall structure, thus reducing the complexity of the future assembly process (Separation of Concerns). For each node, the processes described from (5) to (7) also intertwined themselves in order to optimize the timeline of the fabrication process. After being laser cutted at STMRobotics, all the polypropylene plastic strips had to be sorted by their index number to ease the further fabrication process. This above design-to-fabrication workflow allowed the participants to track each piece and detect their neighboring pieces, within and between each node. The six nodes have been constructed parallelly but assembled one after the other, facilitating the reach of the connections from both inside and outside the nodes during the manual placement of the rivets. The final assembled physical prototype has been hanged with nylon threads in the main entrance of CEDIM, between the first and the ground floor through the mezzanine aperture. The tension of the structure was adjusted empirically by calibrating the tensile force and anchor location of each thread.

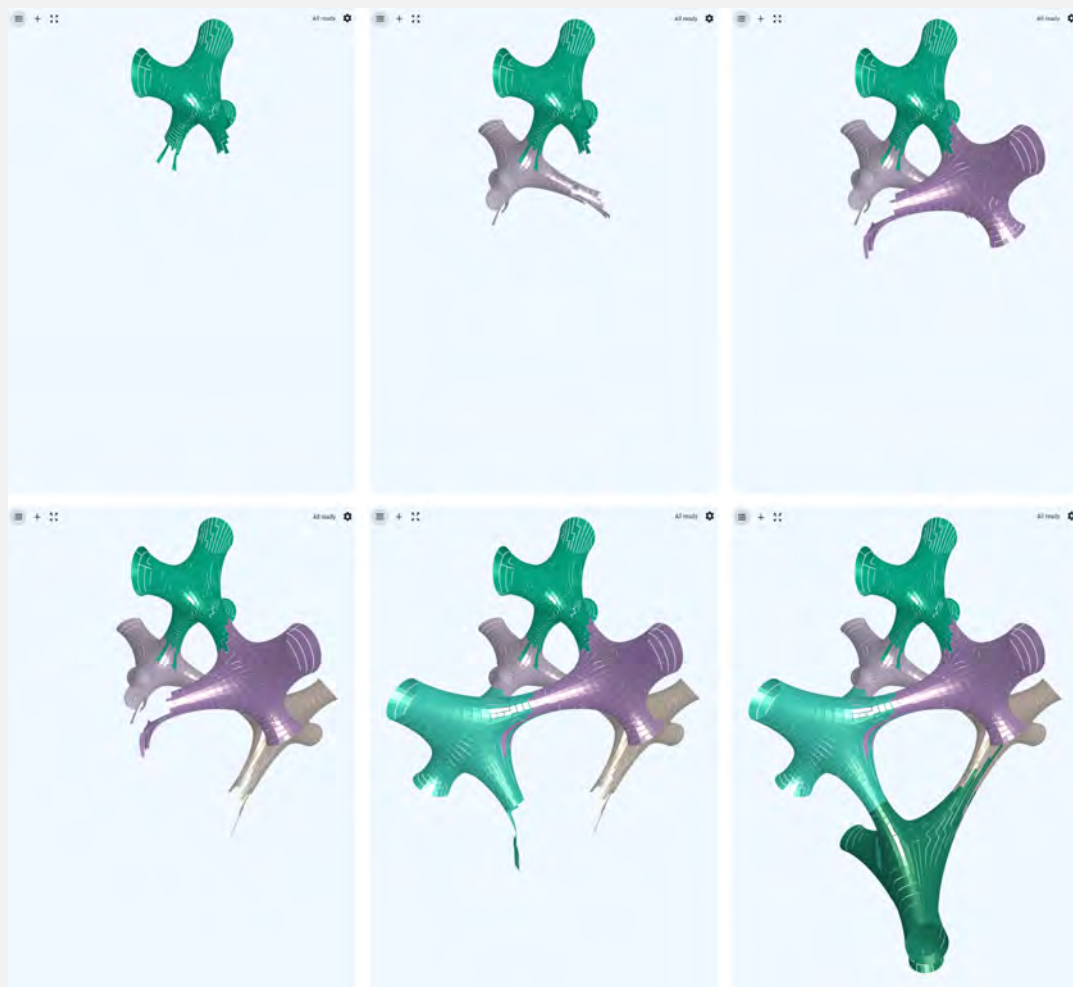


Figure 5.29: The nodes of the overall structure can gradually be streamed within the same [Speckle](#) viewer interface.

Communication strategies through **Speckle**

The aggregation of the **Speckle** streams within the **Speckle** server allowed more flexibility in the design and fabrication process as the team members were not required to work in the same place, but could instead spread themselves across the many different locations in which the design and fabrication of the overall structure took place, and work remotely from those (e.g. the laser cutter at STMRobotics for fabrication, the workshop for assembly, the exhibition space for installation, etc.). Through the **Speckle**'s Grasshopper Client, both Senders and Receivers were used to seamlessly share design data across the pipelines. At each design transaction, the computational designer was able to attach metadata to the shared geometry and the latter could be streamed to the **Speckle** viewer (Figure 5.28). The streams could therefore be aggregated and displayed within a common viewer accessible by all participants (Figure 5.29).

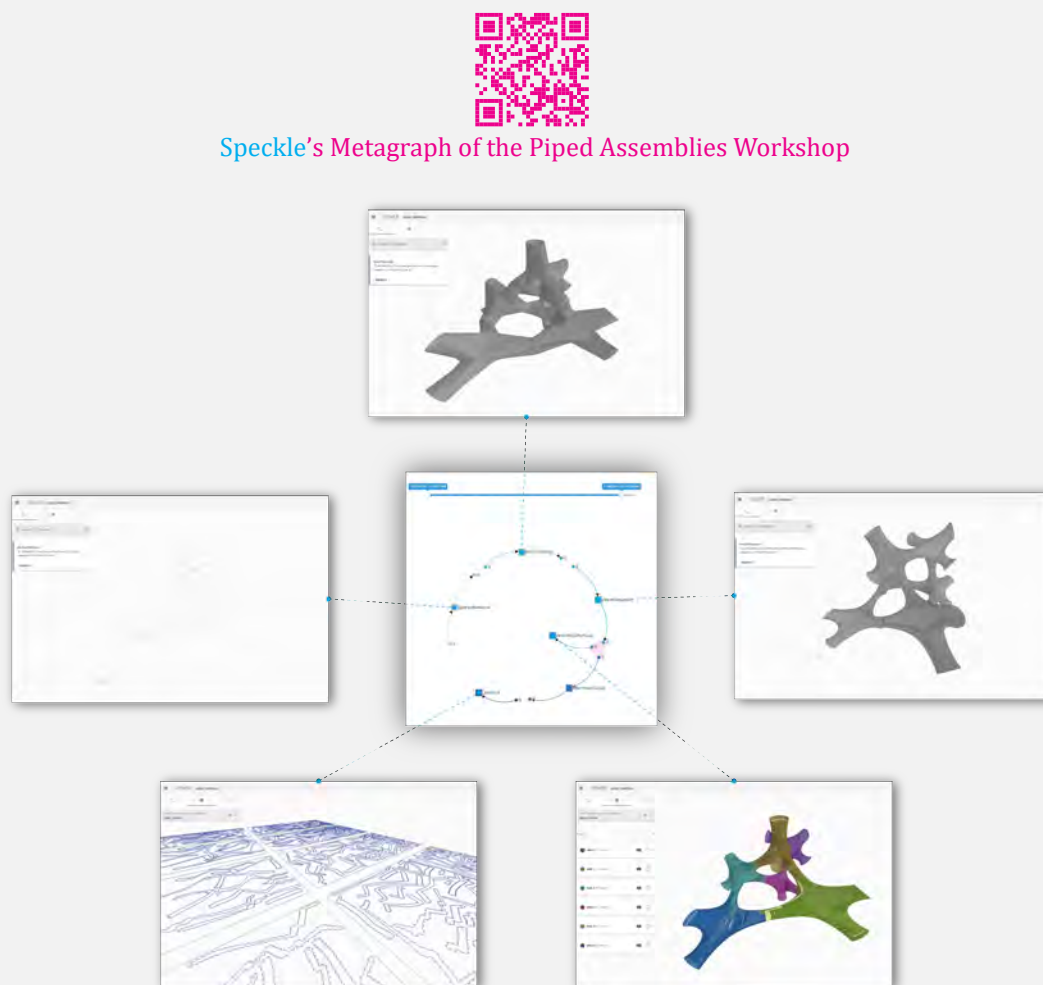


Figure 5.30: **Speckle** metagraph of the Piped Assemblies workshop. The nodes of the overall structure can gradually be streamed within the same **Speckle** viewer interface.

Agile vs. Waterfall Methodologies

In software development, there exist two main different project management methodologies employed in practice: Agile and Waterfall. While the Waterfall method follows a sequential, linear process in which each phase must be complete prior to moving to the next phase (and thus until the delivery of the final software product), the agile method is based on a more flexible, incremental and iterative approach. In-stead of in-depth planning at the beginning of

the project, Agile methodologies are open to changing requirements over time and embraces continuous feedback from the end-users. These two different paradigms resonate with the different methodologies employed during the workshop. The present cross-practice collaboration experiment relied on a Waterfall methodology in which each modelling pipeline depends on its predecessor. Indeed, at any stage in the workflow, the computational designer who locally controls the design process of a particular pipeline is able to communicate issues to a colleague situated upstream in the process in order to refine the design inputs and obtain better results downstream. Such feedback loop could take place at any time during the process until all desired requirements are met. On the other hand, the fabrication phases required a more Agile framework, as certain tasks were not necessarily depending on others, as mentioned earlier. In order to ease the project's design and fabrication processes through both the Waterfall and Agile paradigms, the web-based collaboration platform [Speckle](#) has been used to keep track of the fabrication process, both locally during the assembly of one specific node, and globally when connecting multiple nodes between themselves.

Results

This cross-practice experiment focused on a web-based collaborative workflow for the design and fabrication of a complex structure discretized into polypropylene plastic strips, which has been introduced to undergraduate architecture students at CEDIM during the Piped Assemblies workshop. Although challenging in its complexity, the designed structure has been successfully completed at the end of the workshop, proving therefore the value of the employed collaborative methods during both the design and fabrication stages.

Because it was not the primary focus of the workshop, structural analysis has not been performed to evaluate the lightweight hanging structure described here. However, further investigations could focus on integrating structural evaluation within similar design workflows of such complex discretized structures in order to verify their structural integrity both locally and globally, from the stresses in the joints to the effect of bending during the assembly.

5.4 Conclusion

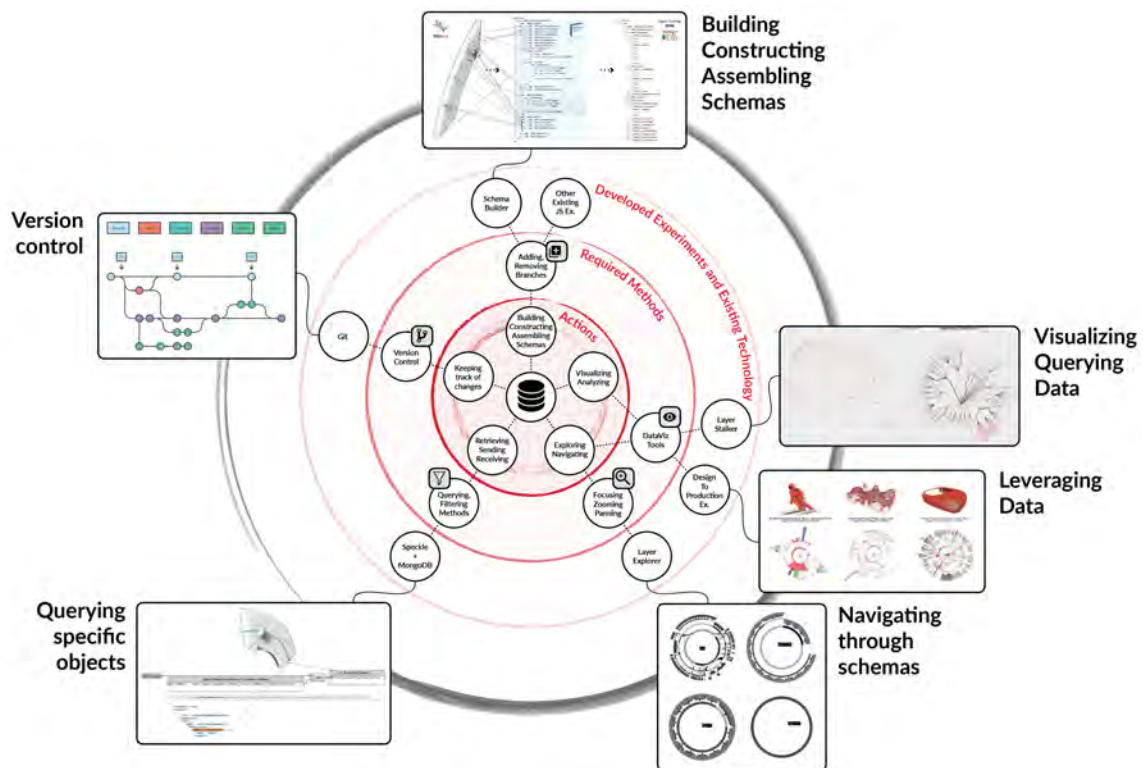


Figure 5.31: Mapping and reflecting upon the experiments: from building, assembling and retrieving schemas, to data-visualization, data-exploration and version control, all those different actions could complete each other to enhance digital workflows for building design.

The present section proposed alternative concepts, tools and methodologies aiming at clarifying and improving the existing building design processes used in the [AEC](#) industry. They are illustrated here by different empirical experiments grounded in the existing everyday practices of BuroHappold and Design-to-Production, both industry partners of the present research project. Those experiments make use of a diverse range of interdisciplinary concepts borrowed from computer science and data visualization: schema-based workflows (Figures 5.15, 5.16 and 5.21), transaction protocols (Figures 5.21 and 5.24) and inter-scalar search interfaces (Figures 5.2, 5.5 and 5.12) are used here to ease and enable the assembly, visualization and query of generated intricate data sets at late stages in the design process. These schema-based workflows (Figure 5.31) for late stages have tackled different points of action:

- **Building schemas**
- **Visualizing schemas**
- **Navigating across schemas**
- **Leveraging late-stage data**
- **Operating adaptive queries**
- **Receiving schemas**

The next chapter will conclude the thesis by reflecting upon the experiments developed so far and by extrapolating their potentials through the speculation of a broader theoretical framework, a Multi-Scalar Modelling paradigm for AEC. This new paradigm aims at facilitating contemporary

5.4. CONCLUSION

digital workflows through the deployment of different visualization, querying, schema-building and exploration strategies across the various scales and trades throughout the whole process of a complex architectural project.

Chapter 6

CONCLUSION AND FUTURE DIRECTIONS: TOWARDS A MULTI-SCALAR MODELLING PARADIGM FOR AEC

The present chapter reflects on the modelling-based experiments, schema-based experiments and search interfaces, and cross-practice collaboration experiments carried so far, and speculates on future potential development and applications to improve current workflows in [AEC](#) through the Multi-Scalar Modelling framework. The chapter is divided into six sections. The first section is reinstating the context of the thesis work and the thesis research goals. The second section summarizes the findings of the modelling and schema-based experiments pursued by the author in chapter 4 and chapter 5. The third section summarizes the main contribution of the thesis. The fourth section attempts to answer the research questions raised in the introduction chapter of the thesis. The fifth section focuses on the limitations of the Multi-Scalar Modelling framework and emphasizes the need for steering software development in [AEC](#) towards open-source. Finally, the sixth and last section provides perspectives regarding future research work that could be further carried to improve the Multi-Scalar Modelling framework for building design.

6.1 Reinstatement of the Context and Research Goals

At the time of writing, 25 years have passed since the introduction of the Industry Foundation Classes ([IFC](#)) in 1994. However, interoperability still remains today a major challenge within the [AEC](#) sector, as observed in chapter 2. In parallel, as the capabilities of 3D modelling software increase, it becomes more and more possible to conceive and build large-scale and complex architectural projects – creating subsequently more complex, richer data during their conception and construction. As observed in section 2.1.3, [CAD](#) modelling software packages have so far neglected interoperability and [BIM](#) concerns to focus instead on improving flexible modelling features. From the other end of the spectrum, buildingSMART – an international organisation formerly known as the International Alliance for Interoperability (IAI) in charge of the development of the [IFC](#) specification – announced the support of [NURBS](#) (enabling precise modelling of complex geometries) only in March 2013 through the release of [IFC4](#). While [CAD](#) modelling software packages need to improve interoperability features, the interoperability standards still need to better support complex modelling features. The current challenge is therefore twofold: enabling better communication between the different existing software platforms, and supporting the translation and management of complex geometries.

In parallel, existing research in academia has looked at setting up fully integrative workflows (as described in section 3.1.3) that take into account different simulation frameworks acting at – and communicating between – the different scales composing the design process of physical demonstrators. Such paradigm is manageable within the context of academia because the tackled scale remains at the level of the pavilion and the involved academic research team usually consists of a small number of persons working within the same institute. In the context of building design and [AEC](#), the design process is much more fragmented – larger array of used software platforms, staff employed at international companies working from different remote locations – and the concerns need to be separated (as observed in section 2.3), limiting the possibilities of setting up similar integrative workflows in [AEC](#), as pointed out in section 2.2.1. Therefore, the present thesis attempts

to embrace these existing fragmented processes in the building industry and interrogates how the existing Multi-Scalar Modelling paradigm used in architectural design research (chapter 3) could help enhancing the current digital design workflows deployed in [AEC](#).

6.2 Summary of Findings

The present thesis has described custom workflows and strategies which tackle different building design problems, both at early design and late stages. Modelling-based and schema-based experiments have been pursued in chapter 4 and chapter 5, through the scope of the theoretical Multi-Scalar Modelling framework introduced and described in chapter 3. In order to support the argument, the thesis has also been grounded through an extensive state of the art of existing modelling practices used in [AEC](#), described in chapter 2.

Chapter 4 focused on the application of Multi-Scalar Modelling design strategies and simulation frameworks formulated in chapter 3 through the investigation of the conducted modelling experiments. Those experiments established different modelling interfaces, from graph-based networks to adapter and skeleton models, and identified the required methods to pass information between multiple scales and across design actors. Those different methods are listed below:

Multi-Scalar Modelling strategies investigated in Chapter 4

- **Approximating the models by defining the required levels and resolutions**
- **Segregating the levels**
- **Defining the coarsening and uncoarsening strategies to navigate across the previously defined levels and resolutions**
- **Setting up connected pipelines, design workflows and handshake techniques to deploy coarsening and uncoarsening strategies**
- **Encapsulating the workflows**
- **Deploying simulation frameworks and optimization strategies across the previously defined pipelines and design workflows**

These different strategies have been successfully implemented within the design probes and prototypes “A1 | *BlankMachine: Propagating the blanks*”, “A2 | *Topological Mapping: Modelling Lap Joints*”, “A3 | *Topological Mapping: Surface/Projection-based modelling*” and “A4 | *Topological Mapping: Spatial Branching*” and “A5 | *The Tallinn Architecture Biennale Installation Competition*”. Especially through the last modelling experiment (A5) which acted as a demonstrator, multiple computational modelling pipelines have been established at different levels of resolutions through multiple grasshopper definitions that communicated via “*Skeleton Models*” defined by the involved designers. High resolution data sets could be accessed through targeted queries from graph models at higher level. It was also demonstrated that the Multi-Scalar Modelling frameworks could host simulation frameworks in “A3 | *Topological Mapping: Surface/Projection-based modelling* (graph relaxation)”, in “A4 | *Topological Mapping: Spatial Branching*” (structural analysis) and in “A5 | *The Tallinn Architecture Biennale Installation Competition*” (mesh relaxation).

The different workshops (“WS1 | *B-MADE Workshop: Fabrication Constraints and Assembly Logistics*”, “WS2 | *LogJam! Workshop: Structural Optimizations and Performance*”, “WS3 | *Exquisite-Timber-Corpse Workshop: Collaborative Practice*” and “WS4 | *Future Wood Seminar: Speculating Design Spaces*”) – carried in section 4.2 – stress tested the Multi-Scalar Modelling strategies listed above. Through those stress tests, it has been observed that despite the effort of deploying these strategies, communication bottlenecks emerged at late stages during which many manual interventions were necessary to handle late design changes. It was therefore

deducted that communicating and passing information across external models and teams were very critical points in the design process, especially at late stages.

Although successful as demonstrating the potential of initial Multi-Scalar Modelling strategies, these modelling experiments investigated in chapter 4 revealed and highlighted that despite an important front-loading strategy (Smith, 2007), the attempt of setting up a continuous digital chain that links multiple scales through Multi-Scalar Modelling and simulation strategies fails at taking into account radical topological changes at late stages. Therefore, the focus and main concerns have been transferred from a modelling to a data management, interoperability and cross-practice collaboration point of view, tackled in chapter 5. More particularly, it has been concluded that a deeper, lower-level infrastructure allowing data exchange between multiple software platforms had to be adopted, alongside to the following strategies:

- **Defining a neutral data format (or schema) to facilitate data-rich object customization, transfer and interoperability**
- **Curating the workflows, improving interoperability and enabling feedback through the development of custom AEC applications**

In this context, chapter 5 demonstrated the benefits of setting up schema-based workflows for late design stages, whose purpose is to complete the Multi-Scalar Modelling strategies tackled in chapter 4 by taking into account scenarios involving late topological changes and manual interventions. The different schema-based workflow strategies tackled chapter 5 are listed below:

Schema-based workflow strategies investigated in Chapter 5

- **Visualizing schemas and leveraging late stage data**
- **Navigating across schemas**
- **Operating adaptive queries and receiving schemas**
- **Building and sharing schemas**

These different strategies have been successfully implemented through the design and development of different prototypical applications:

- The prototypical application developed in “B1 | Visualizing Hierarchies: LayerFootprint” demonstrated alternative data visualization methods to represent intricate datasets from very complex architectural projects. This could help the design technologist to better understand the project complexity “fingerprint” and where data is allocated for different purposes.
- As an extension of the prototypical application “B1 | Visualizing Hierarchies: LayerFootprint”, the demonstrator “B2 - Navigating across Hierarchies: LayerExplorer” showcased the deployment of a meaningful UX within which the expert user was able to navigate across multiple levels of resolutions through complex hierarchical schemas. Indeed, LayerExplorer enabled the user to zoom on particular, local levels of the hierarchical data structure. Thus, instead of continuously trying to comprehend the whole database at once through the complete overview of the sunburst diagram as in “B1 | Visualizing Hierarchies: LayerFootprint” the user was finally able to focus on local levels and navigate between them (Figure 5.5).
- The demonstrator developed in “B4 - Exploring Hierarchies: LayerStalker” validated the possibility of operating adaptive queries – both structured and unstructured – directly from the 3D environment within the GUI interface of LayerStalker. This prototypical

application has been developed successfully and it was possible to deploy and stress test it against multiple Rhino3D models.

- The prototypical application developed in “*B5 - Building and Transferring Hierarchies: SchemaBuilder*” demonstrated the possibility to build seamlessly hierarchical relationships between geometrical objects, resulting in a custom schema set by the expert user. This schema could be further sent and received through [Speckle](#) in a flawless, file-less manner, without any loss of information. Furthermore, the metadata and properties of each object were kept during data transfer across the two user endpoints. Therefore, interoperability has been improved by enabling the sharing of data-rich objects between different stakeholders.

Towards an hybridization of workflows and strategies

Although the above sets of strategies have been investigated within two distinct chapters, one focusing on modelling and simulation frameworks, and the other on late stage concerns, schema-based workflows and search interfaces, it has been argued and demonstrated in section 5.3 – through cross practice collaboration experiments – that those different concepts (simulation frameworks and prototypical search/building schema interfaces) could be hybridized and complete each other, enabling data-rich modelling workflows throughout the whole building design chain, as already suggested in section 2.2.4. Indeed, the first design strategy could be used when the data is *centralized* and can be acted upon with complex simulation frameworks whereas the second strategy is more meaningful when data is *decentralized* and needs to be gathered, curated, analyzed and queried through manual interventions. Figure 6.1 illustrates schematically such hybridization between automated, integrated design strategies at early design stages and segregated manual intervention methodologies at late stages when the data models are more heterogeneous. To pass from one strategy to another without breaking the whole digital chain and preserve the building models, directory tree structures, layer containers and [DAGs](#) could be therefore configured altogether into hybrid data repositories that are able to exchange data-rich schemas between each other, triggering multiple simulation frameworks through both adapter models and search interfaces. This has been partly demonstrated through the cross-practice collaboration experiments in section 5.3.

Cross-practice collaboration experiments investigated in section 5.3

The experiments described in section 5.3 tackled cross-practice collaboration experiments on two different levels:

- **Cross-practice collaboration on the object level**

The experiment “*C1 | Sharing Schemas through [Speckle](#) and [BHoM](#): a Speculative Design Workflow between Design-to-Production and BuroHappold*” demonstrated the possibility of transferring between different trades a data-rich geometrical object whose schema can be adapted depending on the specific local purpose defined by each trade. While Design-to-Production is focusing on the full geometrical description of each architectural component (subsequently working with 3D modelling environments), BuroHappold is mainly focusing on obtaining precise structural analysis results (subsequently working with structural analysis software packages). This particular scenario showed the ability to transfer a single data-rich object whose schema is able to adapt to two different paradigms, from engineering to fabrication concerns.

- **Cross-practice collaboration on the project level**

The experiments “*C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows*” and “*C3 | Piped Assemblies Workshop at CEDIM*” attempted to scale the cross-practice collaboration from the object level to the project level. Instead of focusing on the local level of the schema, these experiments focused on the global level of the project

workflow by linking segregated computational modelling/simulation pipelines through [Speckle](#) senders and receivers, enabling seamless communication across multiple design spaces.

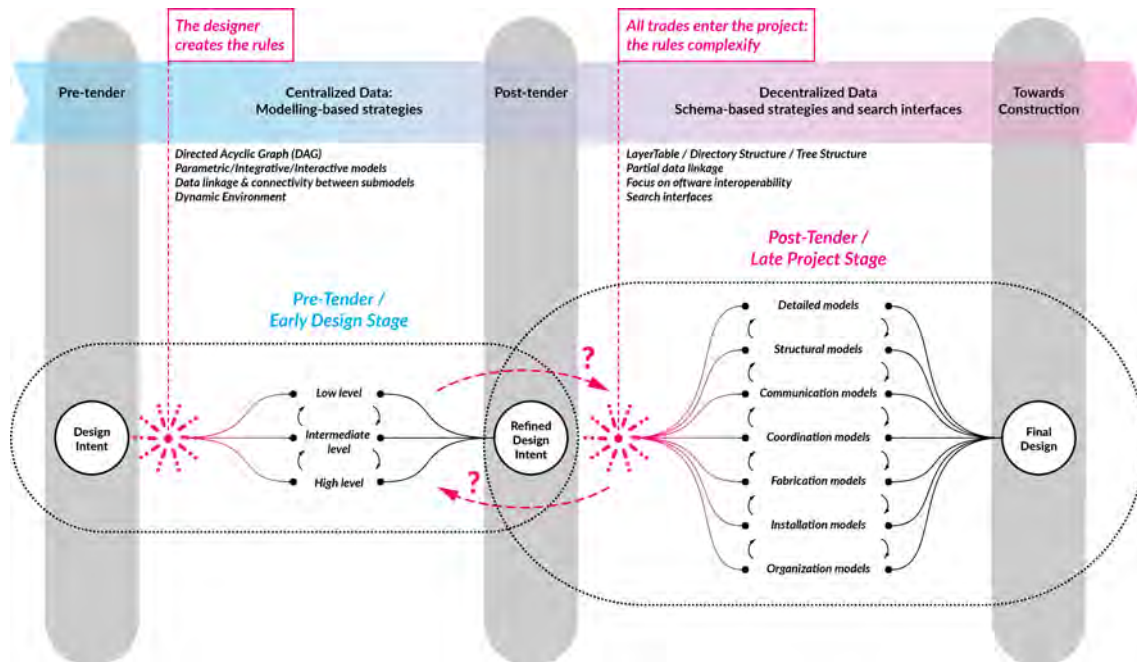


Figure 6.1: This diagram illustrates schematically how integrated and segregated design strategies can coexist and be used in different cases throughout the whole process.

6.3 Contribution of the Thesis

The present thesis has contributed to the body of knowledge of [AEC](#) workflow practices described in chapter 2, by adapting the existing interdisciplinary concept of Multi-Scalar Modelling used within the architectural design research realm and described in chapter 3, so that it can be compatible with the building design domain. The different experiments carried through the thesis work have demonstrated that the current fragmented workflow practices within the [AEC](#) realm can be improved through:

- **the deployment of modelling-based strategies** (graph-modelling and skeleton models) developed and investigated through the first series of modelling-based experiments.
- **the deployment of custom software applications** developed and investigated through the second series of schema-based experiments and search interfaces.

The present thesis contributed in reconciling the academic and industrial realms by converging their respective modes of working towards a same paradigm, a theoretical framework which applies the interdisciplinary concept of Multi-Scalar Modelling to the realm of building design, by both deploying modelling-based strategies investigated in chapter 4 and custom software applications, such as the search interfaces investigated in chapter 5. Both deployments have been synthesized and hybridized in section 5.3 through the series of conducted **cross-practice collaboration** experiments, such as in “C1 | Sharing Schemas through [Speckle](#) and [BHoM](#): a Speculative Design Workflow between Design-to-Production and BuroHappold”. Through this experiment, a common schema (or skeleton model) has been agreed upon by both companies in order to seamlessly share the minimum information required to reconstruct a beam element: here, a simple series of oriented planes. At both ends and within their own modelling software environments, the common schema could be adapted so that each company could separately perform further modelling operations (from Design-to-Production’s perspective) and simulate structural behaviours (from BuroHappold’s perspective). Figure 5.23 illustrates this particular design workflow in which local schema manipulation is combined with more integrative modelling strategies at both ends, through back-propagation (or feedback) behaviours.

As mentioned in the previous section, the **hybridization of workflows and strategies is the key finding and contribution of the present thesis**. Indeed, **hybridizing modelling-based strategies with search interfaces and custom software applications allows seamless data manipulation from the earliest stages to the latest stages**. When different simulations frameworks need to be ran at the start of a building project in order to both optimize and analyze the latter at various scales, modelling-based methods (such as the coarsening and uncoarsening strategies, graph-modelling workflows and skeleton models) can be used and deployed. Once the design is locked on a global level, schema-based workflows and search interfaces can be deployed to manage late stage changes in the design process. These interfaces act on the building data on a more granular level, constantly checking for late topological changes. The user would be able to track, search, modify and curate the data in more meaningful ways. Further design optimizations at later stages (e.g. the discretization of a free-form facade into planar elements) could still require the deployment of modelling-based methods at very local levels. The argument in favour of deploying modelling-based methods in particular situations resonates with Rick Smith’s concluding remarks in his *Technical Notes from experiences and studies in using Parametric and BIM architectural software*:

“We believe parametrically controlled digital modelling has its place in early conceptual design or in the massing stage of a project and also at the back end for manufacturing and fabrication, when warranted or efficient.” (Smith, 2007).

Rick Smith continued his observations by stating that the model should be locked between these two different parametric modelling processes that happen at different ends (at very early stages for the massing stage and at very late stages for manufacturing), which would otherwise have unfortunate consequences in the design process and lead to great liability:

“Once the design settles down you do not want to allow anyone to change the design on their own, especially when the model is issued to the contractors. At that point you want the

geometry of the model to be locked. A mistaken parametric changed later in the process can have a rippling affect that if not noticed could become a great liability.” (Smith, 2007)

Although it would be ideal to lock the geometry once the early stage parametric modelling processes have stopped, it has been already argued in section 2.2.3 that late topological changes in the design process are inevitable and should be taken for granted from the beginning. Therefore, the present thesis extends the parametric modelling design space – that is, according to Smith, taking place locally, usually situated only at very early stage or very late stage – by embedding more intelligence between those two ends through the proposed **custom software applications and search interfaces** investigated and developed in chapter 5. These custom tools enable the architect, computational designer or expert user to build more granular, task-oriented automations that can be combined with direct manipulation – or *Direct Modelling* (Pena De Leon, 2014) – which is more practical and effective than trying to deploy a fully integrative digital workflow across the whole design process chain. Paired with the above mentioned modelling-based strategies, the schema-based experiments and search interfaces enhance cross-practice collaboration at scale, during which the expert user or computational designer can hybridize his/her design practice by alternating between continuous, integrative workflows and more discrete, task-based automation strategies which allow more granular control, in particular at the latest design stages. To pass from modelling-based strategies to schema-based workflows and search interfaces without breaking the whole workflow and maintain consistent building models, directory tree structures (as in “B5 / Building and Transferring Hierarchies: SchemaBuilder” and “B2 / Navigating across Hierarchies: LayerExplorer”), layer containers and DAGs/graph modelling strategies (as in “A5 / The Tallinn Architecture Biennale Installation Competition”) should form an hybrid data infrastructure allowing both integrative design workflows and discrete data manipulation (as in “C1 / Sharing Schemas through Speckle and BHoM: a Speculative Design Workflow between Design-to-Production and BuroHappold”) so that data-rich schemas can be seamlessly built, searched, shared and ultimately trigger further simulation frameworks throughout the design process. This particular hybrid mode of working shapes the Multi-Scalar Modelling framework for building design and AEC.

6.4 Answering the Research Questions

Taking into account the summary of contributions described in the previous section, the present section attempts to answer the three research questions raised in the introduction of the present thesis.

- ***How can a Multi-Scalar Modelling framework allow the designer to work across different scales in order to take into account multiple constraints related to material, fabrication and structural performances during both early design and late stages?***

In the present thesis, the Multi-Scalar Modelling framework takes both into account modelling-based strategies at early stages – as investigated in chapter 4 (“A1 | *BlankMachine: Propagating the blanks*”, “A2 | *Topological Mapping: Modelling Lap Joints*”, “A3 | *Topological Mapping: Surface/Projection-based modelling*”, “A4 | *Topological Mapping: Spatial Branching*” and “A5 | *The Tallinn Architecture Biennale Installation Competition*”) – and schema-based strategies, search interfaces at late stages – as investigated in chapter 5. The modelling-based strategies can be deployed when only a few actors are involved in the design process. The reason for this is that the developed digital workflows and related data need to be highly curated in order to set up simulation and modelling frameworks enabling the passing of information through multiple levels of resolution.

It is then challenging to migrate the Multi-Scalar Modelling framework to the latest stage, from the post-tender phase: multiple parties are involved and the project data becomes decentralized across trades, files and locations. Therefore, the second series of strategies needs here to be deployed through custom software development and Graphical User Interface (GUI) to enable better interoperability and enhance digital workflows. More particularly, the schema-based experiments and search interfaces developed in chapter 5 help the user to visualize (“B1 | *Visualizing Hierarchies: LayerFootprint*”), browse (“B2 | *Navigating across Hierarchies: LayerExplorer*”), explore (“B4 | *Exploring Hierarchies: LayerStalker*”) and collate (“B5 | *Building and Transferring Hierarchies: SchemaBuilder*”) building data in more intuitive ways. The further hybridization of the modelling based strategies and custom software applications (developed through the second series of schema-based experiments and search interfaces) constitutes the final Multi-Scalar Modelling framework for building design allowing for both integrative, automated tasks at early stages and more direct data manipulation at later stages.

As observed in section 2.4.2, existing developments in the AEC industry are often undertaken in isolation. Alternatively and ultimately, these custom developments should be developed on top of a common open-source platform laying the foundation for a common neutral object model and repository. This would enable the AEC community to gather and access the developed resources, preventing the design technology specialists to keep reinventing the wheel.

- ***How can the end-user share, access, track and modify at multiple scales data-rich objects sent and received from different trades within a common directory structure until completion of the building?***

It has been described in chapter 5 that custom Graphical User Interfaces (GUI) can be developed to enable the end-user to share, access, track and modify at multiple scales data-rich objects sent and received from different trades. Developed through the present thesis work, *LayerFootprint* allowed to visualize complex hierarchical data sets, *LayerExplorer* and *LayerStalker* enabled the user to visualize, navigate through these same schemas in a better and more sustainable way, and *SchemaBuilder* enabled to build data-rich, custom geometrical data sets that could be directly shared and queried through the open-source *Speckle* platform. Although the mentioned “common directory structure” has not been developed independently through the present research work, the theoretical Multi-Scalar Modelling framework has been demonstrated through the use of the open-source communication platform *Speckle*. It is suggested that the underlying object infrastructure of the Multi-Scalar Modelling framework should be open-source, as argued in section 2.5. From an open-source platform, the manipulated data of a particular project could be centralized, and further accessed by anyone who has been given read and/or write access permissions. The cross-practice collaboration

experiments described in chapter 5.3 (“C1 | Sharing Schemas through [Speckle](#) and [BHoM](#): a Speculative Design Workflow between Design-to-Production and BuroHappold”, “C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows” and “C3 | Piped Assemblies Workshop at CEDIM”) illustrate cross-practice collaboration at scale through the open-source platforms [Speckle](#) and [BHoM](#).

- **How would an ideal Multi-Scalar Modelling AEC-model look like and which requirements would it have to fulfil all user’s requests? How could the multi-scalar model be interacted with and which User Interface (UI) and User Experience (UX) concepts would be needed?**

The common directory structure – or the open-source communication platform – suggested in the answer to the previous question, would represent here the basis for further software development of specific User Interfaces ([UI](#)), Graphical User Interfaces ([GUI](#)) and search interfaces: the different schema-based experiments and prototypical applications developed through the thesis work (such as “B1 | Visualizing Hierarchies: [LayerFootprint](#)”, “B2 | Navigating across Hierarchies: [LayerExplorer](#)”, “B4 | Exploring Hierarchies: [LayerStalker](#)” and “B5 | Building and Transferring Hierarchies: [SchemaBuilder](#)”) could be deployed as add-ins to the open-source platform. In this context, those experiments would not be developed and maintained in isolation, but deployed through the open-source platform, enabling therefore any interested party to use them.

As stated by Bauke de Vries, “the best way to stimulate the introduction of electronic communication is to create a clear view of the current information exchange process inside and outside the company. From that, the potencies of new technologies can be explored.” ([de Vries, 1996](#)). In other words, if an [AEC](#) company could clearly map visualize its internal information exchange (as illustrated in “C3 | Piped Assemblies Workshop at CEDIM”), the appropriate communication technologies – as argued here, an open-source communication platform – could be deployed, offering an infrastructure to develop third-party interfaces and applications.

The integration of the developed prototypical applications described in chapter 5 through the thesis work in within a common directory structure or open-source communication platform would shape the ideal Multi-Scalar Modelling framework for [AEC](#). However, as user requests might change and evolve over time, it is difficult to give a clear picture of the final developed interface that would be gradually improved before becoming more and more crystallized. Furthermore, enabling expert-users and contributors to participate in the perpetual development of third party open-source applications is primordial and should be encouraged by promoting and fostering a community of [AEC](#) software developers willing to contribute to the open-source platform.

To summarize, the present thesis is contributing to the field of [AEC](#) by setting and specifying the design criteria to enhance collaborative practices across multiple scales from the early design phases to the latest stages. While the early phases can be tackled through modelling-based practices (as exemplified and demonstrated in chapter 4), the latest stages could benefit of schema-based, custom software applications (as exemplified and demonstrated in chapter 5). These two different ends should be hybridized in order to pass from one strategy to the other (as exemplified and demonstrated in section 5.3). Furthermore, it is suggested that the mentioned custom software application should be built upon a larger open source communication platform in order to benefit the larger [AEC](#) community, enabling the reuse of these same applications for future architectural projects.

6.5 Limitations

As aforementioned in section 5.3 and observed through the conducted cross-collaborative modelling experiments, the human factor cannot be overlooked in favour of a purely automated technological paradigm, needing to be strongly considered throughout all aspects of the design process. In order to enable Multi-Scalar Modelling, co-creation and co-authorship behaviours need to be encouraged, both in terms of project architecture and team competencies. Although the Multi-Scalar Modelling framework can be enhanced by custom User Interfaces (UI), Graphical User Interfaces (GUI) and/or User Experiences (UX) as investigated through the schema-based workflow experiments in chapter 5, the act of curating the data and modelling workflows remains to the end-user. The “end-user” is qualified here as an advanced or expert user that performs complex modelling tasks within wider collaborative workflows involving external stakeholders and other advanced end-users. The theoretical Multi-Scalar Modelling framework proposed in this thesis cannot solve all the technological problems faced by the different AEC practitioners on an everyday basis. It gives concepts and tools that can help the end-users to better navigate across scales and resolutions through the design process but the responsibility of curating and setting up the project modelling workflow is left to them.

Open-source Frameworks and Project Management Platforms as Enablers of Transparent Collaborative Design Workflows in AEC

To deploy a Multi-Scalar Modelling framework across multiple trades, models and scales, it has been argued in 2.4.3 that the AEC industry needs to steer software development towards open-source frameworks which would further enable transparent collaborative design workflows. It has also been observed in chapter 2 that many different companies are developing their own custom tools and workflows, as the currently available software platforms for AEC do not meet their specific needs. Even though the currently developed custom solutions (as described in section 2.4.2) are very valuable in solving immediate and critical issues related to the conception and construction phases of different building projects, they are being developed in isolation. These approaches are indeed naturally not focused on – and cannot address – the wider related challenges of data interchange, collaboration and inefficiency across the complete supply chain. There is no common agreement for setting up an underlying data-exchange infrastructure that would enable better linkage between the different practices, which constantly need to build custom application and workflows for unrevealed design problems. Therefore, the AEC industry needs to transition from the current state of the art of design and code experimentation to solve local problems through to development and then deployment and reuse; and importantly with increased transparency and participation at every stage. This means a paradigm shift towards collaborative open-source development to improve the current workflows. Ultimately, this would lead to reconsider the current pyramid scheme of the global software development in the AEC industry.

Reversing the pyramid scheme of the global software development in the AEC industry

It has been highlighted in the state of the art and demonstrated through the carried experiments in the last chapters that the AEC industry currently needs digital infrastructures and communication platforms at lower levels (such as Speckle or Konstru) to optimize processes and workflows across software environments and external stakeholders. These communication platforms would also enable full deployment of the Multi-Scalar Modelling concepts investigated throughout the thesis. Currently, these digital infrastructures need to be intensively maintained and a lot of effort is spent on keeping track of last Application Programming Interface technical updates coming from the different software vendors. If those updates would be disregarded, the communication platforms would take the risk of becoming obsolete and therefore unusable by its users. Such dependency would not be sustainable in the long term, as the effort would be mainly placed on keeping the links and adapter plug-ins operational, rather than on developing query and search interfaces, visualization tools and other Multi-Scalar Modelling techniques described through this thesis. It is suggested here that the pyramid dependency scheme model should be reversed: the software vendors should deliver their latest software releases with the appropriate adapter plug-ins enabling the data transfer to the main communication platform(s), as illustrated in figure 6.2. This would

allow those platforms to focus their development on developing and optimizing third party software applications to improve project management for building design. Realistically, if such speculative scenario happens, the software vendors won't invest substantial development efforts to match their data conversion logics with many of these communication platforms for AEC. Only a few might be able to benefit from this.

Robert McNeel & Associates has proved to be very well aware of this upcoming paradigm shift, through the development of Rhino.Inside and Rhino Compute (introduced in section 2.1.3). While the former allows Rhino and Grasshopper to be embedded within other products (such as Revit and AutoCAD) by calling directly the host's native API from a Grasshopper or Rhino plug-in, the latter allows access to the Rhino's geometry library through a stateless REST API via the McNeel cloud. Therefore, McNeel & Associates has placed more concerns and development effort to standardize their API and interoperate with other software platforms.

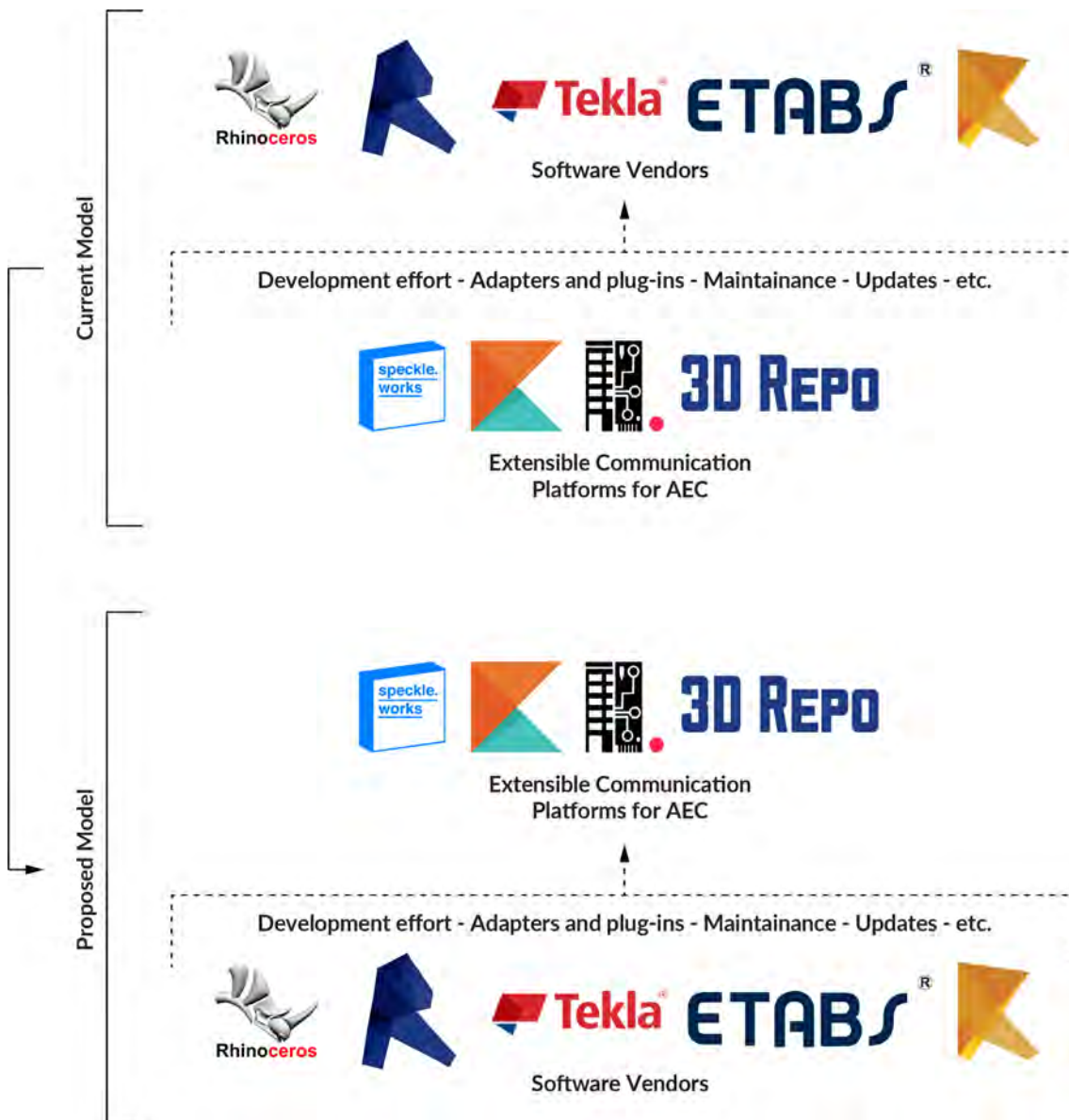


Figure 6.2: The traditional pyramid scheme — in which the development of the generic communication platforms needs to adapt to the software industry — is here reversed: the software vendors should make the effort to develop adapter plug-ins, maintain and update them in order to adapt to the latest versions of the communication platforms.

6.6 Perspectives

Custom project management and version control interfaces

The natural implication of a Multi-Scalar Modelling framework and distributed development approach such as that proposed by this thesis is the ability to exploit network effects and tends towards not only an increasing collective intelligence, but through improved means of communication, participation and access, the means to utilize this as a collective know-how. Through the previously described series of modelling and schema-based experiments and prototypical applications, the present thesis work presented here the progress made in establishing such an approach. However, an architectural project can be understood not only through the multiplicity of its different scales, but also through the time dimension and its different design versions and iterations. This means that a project management and a version control interface should also be thought to complete the wider scope of Multi-Scalar Modelling for Building Design. Such interfaces would enhance the [UI](#) during the design process. The computational designers and end-users would be able to track changes, retrieve models that have been previously transferred at specific times and obtain a better understanding through the project's corresponding global *Metagraph* as illustrated in figure 2.29. From those same interfaces can unfold different levels of granularity that can be aggregated, analyzed and shared between the different involved trades.



Demo of the [Speckle's Metagraph](#)

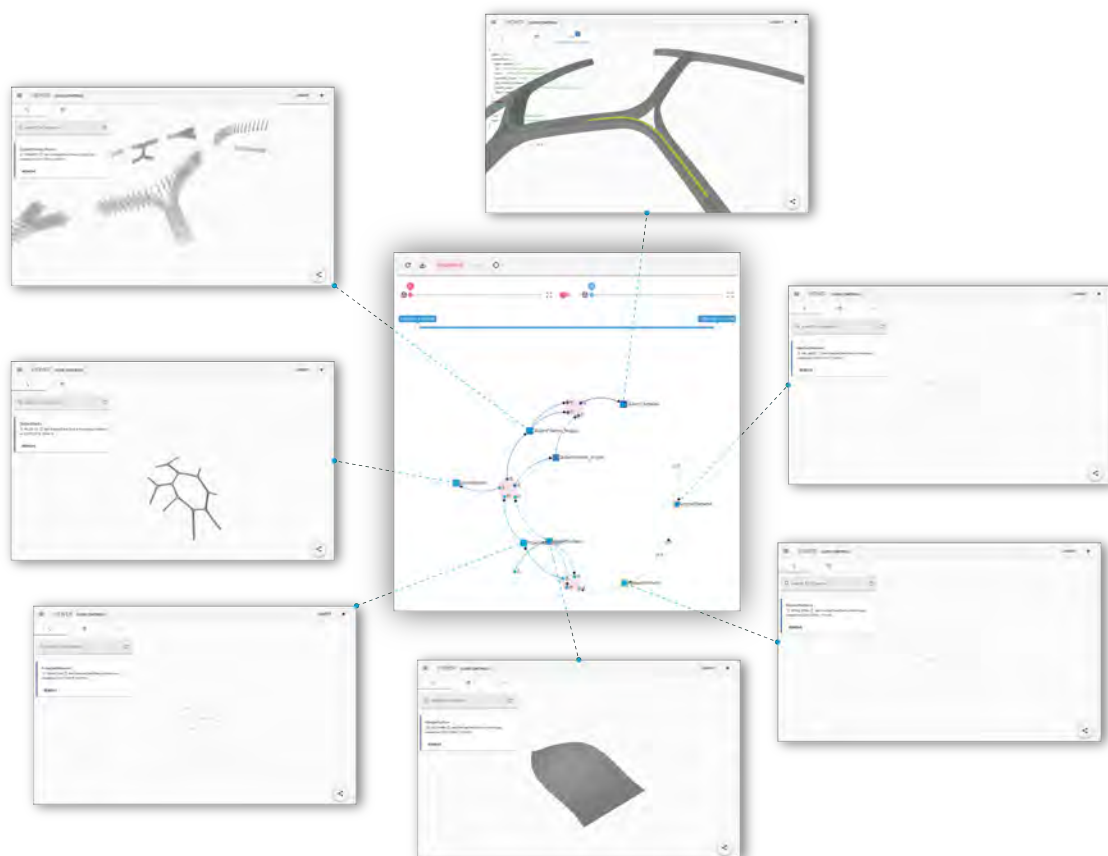
The AEC Delta Mobility project

Since April 2019, the author has been involved within the AEC Delta Mobility project, funded by the InnovateUK from February 2019 to July 2020 to iron out the data exchange processes in the Architecture, Engineering, Construction ([AEC](#)) sector. The project addresses data exchange in an open-source manner and gathers multiple industry partners and collaborators in a consortium, including BuroHappold, 3D Repo, [Speckle](#) and Rhomberg Sersa Rail Group, with external support from HOK, Atkins (SNC Lavalin) and Arup. The team directs its research and development efforts towards solving the inefficient processes during which the software end-users (architects, engineers and fabricators) manually exchange large files and 3D models to communicate design changes (deltas) among themselves.

During the project, the author developed a *Client Graph* within the [Speckle](#) admin interface. This *Client Graph* is a custom Graphical User Interface [GUI](#) built within the [Speckle](#) Project panel. For a selected project, the end-user is able to visualize and track in real-time the created streams, as well as the respective senders and receivers. The *Client Graph* therefore brings the former concept of the *Metagraph* (Figure 2.29) to reality, by representing in real-time the overall project workflow on a high-level, while offering the possibility to navigate at the lower level of the geometrical object. From this interface, it is possible to select multiple streams, aggregate them and inspect them within the [Speckle](#) viewer. The networked graph can also be filtered with a time range slider from which the end-user can select a specific time-frame. The streams, senders and receivers comprised within the selected time-frame will be highlighted and the other faded. For the sake of illustration, the experiment “C2 | SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows” has been visualized through the [Speckle](#) Client Graph (Figure 6.3).



(a) The [Speckle](#) Metagraph of the modelling experiment C2 / *SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows*.



(b) Implementation of the metagraph within the [Speckle](#) admin interface. Here the user can have access and visualize all the streams contained within a particular project, and visualize the object contained within specific streams. The networked graph can also be filtered with a time range slider from which the end-user can select a specific time-frame.

Figure 6.3: The [Speckle](#) Metagraph of the modelling experiment C2 / *SimAUD Workshop at TU Delft: Open Collaborative Design, Simulation & Analysis Flows*.

CARRIED SECONDMENTS

The author pursued a series of secondment both at Design-to-Production and BuroHappold, as well as at Robert McNeel & Associates (a software company developing the commercial 3D computer graphics and computer-aided design (CAD) application software Rhinoceros 3D) during specific periods in the research project that required more in-depth software development. The visits totalize respectively 7 weeks at Design-to-Production, 5 weeks at BuroHappold and 8 weeks at McNeel:

- **From the 19th of January to the 9th of February 2016**
Secondment at: Design-to-Production
Duration: 3 weeks
- **From the 21st to the 25th of March 2016 and from the 19th-21st of July 2016**
Secondment at: BuroHappold
Duration: 7 days in total
- **From the 15th of March to the 15th of April 2017**
Secondment at: Design-to-Production
Duration: 1 month
- **From the 1st to the 30th of September 2017**
Secondment at: BuroHappold
Duration: 1 month
- **From the 4th of December to the 1st of February 2018**
Secondment at: McNeel
Duration: 2 months

Bibliography

- Ackoff, R. L.
1989. From data to wisdom. *Journal of Applied Systems Analysis*, 16(1):3–9.
- Acton Ostry Architects Inc.
2016. Design and Preconstruction of a Tall Wood Building - Brock Commons Phase 1: Design Modelling. *Naturally:Wood*, Pp. 1–16.
- Aish, R.
2013. DesignScript: Scalable Tools for Design Computation. *Computation and Performance - Proceedings of the 31st eCAADe Conference*, 2:87–95.
- Aish, R., A. Fisher, S. Joyce, and A. Marsh
2012. Progress towards Multi-Criteria Design Optimisation using DesignScript with SMART Form, Robot Structural analysis and Ecotect building performance analysis. *Acadia*, II(2007):47–56.
- Alexander, C.
1965. A City is Not a Tree. *Architectural Forum*, 122(1):58–62.
- Alexander, C.
1973. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.
- Amor, R. and I. Faraj
2001. Misconceptions about integrated project databases. *Electronic Journal of Information Technology in Construction*, 6(July):57–68.
- Aranda-mena, G., P. R. Wakefield, and D. G. Aranda-Mena
2006. Interoperability of Building Information: Myth or Reality? *eWork and eBusiness in Architecture, Engineering and Construction*, (2002):127–133.
- Attar, R., R. Aish, J. Stam, D. Brinsmead, A. Tessier, M. Glueck, and A. Khan
2010. Embedded Rationality: A Unified Simulation Framework for Interactive Form Finding. *International Journal of Architectural Computing*, 8(4):399–418.
- Band, J. and M. Katoh
1995. *Interfaces On Trial*. Boulder, CO, USA: Westview Press.
- Band, J. and M. Katoh
2011. *Interfaces On Trial 2.0*. The MIT Press.
- Baran, P.
1964. On Distributed Communications Networks. In *IEEE Transactions of the Professional Technical Group on Communications Systems*.
- Barison, M. B. and E. T. Santos
2010. An overview of BIM specialists. *Proceedings of the International Conference on Computing in Civil and Building Engineering icccbe2010*, (August):141.
- Barnes, S. B.
1995. User Friendly: A Short History of the Graphical User Interface. *Sacred Heart University Review*, 16(1).

- Barth, T. J., M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, eds.
2009. *Multiscale Modeling and Simulation in Science*. Berlin Heidelberg: Springer.
- Bernstein, P.
2012. Intention to artifact. In *Digital Workflows in Architecture: Design-Assembly-Industry*, Pp. 64–71.
- Boehm, B.
1976. Software Engineering. *IEEE Transactions on Computers*, C-25(12):1226–1241.
- Boeykens, S.
2012. Bridging Building Information Modeling and Parametric Design. *eWork and eBusiness in Architecture, Engineering and Construction, 9th ECPPM Conference Proceedings*, (March):453–458.
- Brandt, J.
2012. The Death of Determinism. In *Persistent Modelling: Extending the role of architectural representation*, P. Ayres, ed., chapter Chapter 9, Pp. 105–116. New York, NY: Routledge.
- Burger, S.
2014. South Australian Health and Medical Research Institute (SAHMRI). In *ACADIA 2014, Design Agency*, Pp. 201–204.
- Burphy, J. R.
2007. Mindful Spaces : Computational Geometry and the Conceptual Spaces in which Designers Mindful Spaces : Computational Geometry. *International Journal of Architectural Computing*, 05(04):611–624.
- Burphy, M.
1996. Parametric design and the Sagrada Familia. *Architectural Research Quarterly*, 1(4):70–81.
- Burphy, M.
2011. *Scripting Cultures: Architectural Design and Programming*, first edition. Chichester, West Sussex: John Wiley & Sons Ltd.
- Bush, V.
1945. As We May Think. *Atlantic*, (July):35–46.
- Carpio, M.
2017. *The Second Digital Turn: Design Beyond Intelligence*. Cambridge, Massachusetts, USA: MIT Press, Writing Architecture Series.
- Davies, A.
2019. Project management for large, complex projects. Technical report, The Bartlett School of Construction and Project Management, London.
- Davis, D.
2013. *Modelled on Software Engineering : Flexible Parametric Models in the Practice of Architecture*. PhD thesis, RMIT University.
- de Vries, B.
1995. Message Development in the Building Process. In *Modeling of Buildings through their Life-Cycle. Proceedings of the CIB w78 Conference*, Pp. 467–479, Stanford.
- de Vries, B.
1996. *Communication in the building industry: a strategy for implementing electronic information exchange*. PhD thesis, Eindhoven.
- Deleuran, A., M. Pauly, M. Tamke, and I. Friis
2016. Exploratory Topology Modelling of Form-Active Hybrid Structures.

- Deleuran, A. H.
2015. Synthesizing a Nonlinear Modelling Pipeline for the Design of Masonry Arch Networks. Pp. 65–81.
- Denari, N.
2014. Precise Form for an Imprecise World. In *Digital Workflows in Architecture: Design-Assembly-Industry*, S. Marble, ed., Pp. 28–45. Basel: Birkhäuser.
- Desourdis, R. I., P. J. Rosamilia, C. P. Jacobson, J. E. Sinclair, and J. R. McClure
2009. *Achieving Interoperability in Critical IT and Communication Systems*. Boston | London: Artech House.
- Deutsch, R.
2015. *Data-Driven Design and Construction - 25 strategies for capturing, analyzing and applying building data*. Hoboken, New Jersey: John Wiley & Sons Ltd.
- Deutsch, R.
2017. *Convergence: The Redesign of Design*. Wiley.
- Dijkstra, E. W.
1972. Notes on structured programming. Technical Report 1970.
- Dijkstra, E. W.
1982. On the Role of Scientific Thought. *Selected Writings on Computing: A personal Perspective*, Pp. 60–66.
- Dourish, P., W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton
2000. Extending document management systems with user-specific active properties. *ACM Transactions on Information Systems*, 18(2):140–170.
- E, W.
2011. *Principles of Multiscale Modeling*. New Jersey: Princeton University.
- ElNimr, A., M. Fagiar, and Y. Mohamed
2016. Two-way integration of 3D visualization and discrete event simulation for modeling mobile crane movement under dynamically changing site layout. *Automation in Construction*, 68:235–248.
- Engelbrecht, F. A., W. A. Landman, C. J. Engelbrecht, S. Landman, M. M. Bopape, B. Roux, J. L. McGregor, and T. M
2011. Multi-scale climate modelling over Southern Africa using a variable-resolution global model. *Water SA*, 37(5):647–658.
- Esau, K.
1960. *Anatomy of Seed Plants*. Wiley.
- Faircloth, B., R. Welch, M. Tamke, P. Nicholas, P. Ayres, Y. Sinke, B. Cuffy, and M. R. Thomsen
2018. Multiscale modeling frameworks for architecture : Designing the unseen and invisible with phase change materials. *International Journal of Architectural Computing*, 16(2):104–122.
- Frayling, C.
1993. Research in Art and Design. *Royal College of Art - Research Papers*, 1(1):1–5.
- Hauberg, J.
2012. Research by Design - a research strategy. *Revista Lusófona de Arquitectura e Educação*, 5:46–56.
- Helmut, P., A. Asperl, M. Hofer, and K. Axel
2007. *Architectural Geometry*, first edit edition. Exton, Pennsylvania USA: Bentley Institute Press.

- Hirz, M., W. Dietrich, A. Gfrerrer, and J. Lang
2013. Integrated computer-aided design in automotive development: Development processes, geometric fundamentals, methods of CAD, knowledge-based engineering data management. *Integrated Computer-Aided Design in Automotive Development: Development Processes, Geometric Fundamentals, Methods of CAD, Knowledge-Based Engineering Data Management*, 9783642119:1–466.
- Holten, D.
2006. Hierarchical Edge Bundles: Visualizaiton of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748.
- Holzer, D.
2007. Are you talking to me? Why BIM alone is not the answer. *BIM-legal*, Pp. 108–114.
- Holzer, D.
2016. *The BIM Manager's Handbook - Guidance for Professionals in Architecture, Engineering, and Construction*. Chichester, West Sussex: John Wiley & Sons Ltd.
- Hornby, G. S., H. Lipson, and J. B. Pollack
2001. Evolution of generative design systems for modular physical robots. *Proceedings - IEEE International Conference on Robotics and Automation*, 4:4146–4151.
- Howard, H. C., R. E. Levitt, B. C. Paulson, J. G. Pohl, and C. B. Tatum
1989. Computer Integration: Reducing Fragmentation in AEC Industry. *Journal of the Japan Society of Air Pollution*, 3(1):18–32.
- Inselberg, A. and B. Dimsdale
1990. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the First IEEE Conference on Visualization: Visualization '90*. IEEE.
- Jabi, W., R. Aish, S. Lannon, A. Chatzivasileiadi, and N. M. Wardhana
2018. Topologic: A toolkit for spatial and topological modelling. *Computing for a Better Tomorrow, Proceedings of the 36th eCAADe conference*, 2:449–458.
- Jernigan, F.
2008. *BIG BIM little bim*. 4Site Press.
- Kavitha, T., C. Liebchen, K. Mehlhorn, M. Dimitrios, R. Rizzi, T. Ueckerdt, and K. Zweig
2009. Cycle Bases in Graphs Characterization, Algorithms, Complexity, and Applications. Pp. 1–89.
- Kelly, K.
1998. New rules for the new economy. P. 178.
- Kilian, A.
2006. Design Exploration through Bidirectional Modeling of Constraints. *PhD Thesis*, P. 325.
- Kilian, A.
2013. Design Exploration and Steering of Design. In *Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design*, B. Peters and T. Peters, eds., Pp. 122–129. London: John Wiley & Sons Ltd.
- Knippers, J., A. Menges, H. Dahy, N. Frueh, A. Groenewolt, A. Koerner, K. Rinderspacher, S. Saffarian, E. Slabbinck, J. Solly, L. Vasey, and D. Wood
2018. The ITECH approach: Building(s) to learn. In *Proceedings of the IASS Symposium 2018: Creativity in Structural Design*.
- Knuth, D. E.
1976. Structured Programming With and Without GO TO Statements. *Computing Surveys*, 6(4):261–301.
- Krogh, P. G., T. Markussen, and A. Louise
2015. ICoRD'15 – Research into Design Across Boundaries Volume 1. 34(January).

- Krzywinski, M., J. Schein, I. Birol, J. Connors, R. Gascoyne, H. Doug, S. J. Jones, and M. A. Marra
2009. Circos: an Information Aesthetic for Comparative Genomics. *Genome Res*, 19:1639–1645.
- Kunz, A., T. Maigne, and A. Witt
2014. 3D: an avatar of drawing. In *The Fondation Louis Vuitton by Frank Gehry: A Building for the Twenty-First Century*, A.-L. Roccati, ed., Pp. 123–128. Paris: Flammarion.
- Li, J.-M. and J. Knippers
2015. Segmental Timber Plate Shell for the Landesgartenschau Exhibition Hall in Schwäbisch Gmünd—the Application of Finger Joints in Plate Structures. *International Journal of Space Structures*, 30(2):123–139.
- Linkwitz, K. and D. Veenendaal
2014. Nonlinear force density method: Constraints on force and geometry. In *Shell Structures for Architecture: Form Finding and Optimization*, S. Adriaenssens, P. Block, D. Veenendaal, and C. Williams, eds., chapter Twelve, Pp. 143–157. New York, NY: Routledge.
- Ludolph, F. and R. Perkins
1998. The Lisa User Interface. *CHI 98 Conference Summary on Human Factors in Computing Systems, ACM*, Pp. 18–19.
- Marble, S. and J. Kotronis
2012. Workflow Consultancy. In *Digital Workflows in Architecture: Design-Assembly-Industry*, Pp. 150–169. Birkhäuser.
- Matloff, N.
2008. Introduction to discrete-event simulation and the simpy language. Pp. 1–33.
- McCabe, T. J.
1976. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320.
- McLuhan, M. and Q. Fiore
1967. *The Medium is the Massage - An Inventory of Effects*. London: Penguin Books.
- Menges, A.
2012. Morphospaces of Robotic Fabrication. In *Robotic Fabrication in Architecture, Art and Design*, S. Brell-Çokcan and J. Braumann, eds., Pp. 28–61, Wien, New York. Springer.
- Menges, A. and S. Ahlquist, eds.
2011. *Computational Design Thinking*. John Wiley & Sons Ltd.
- Meyer, M., T. Munzner, and H. Pfister
2009. MizBee: A multiscale synteny browsers. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):897–904.
- Miller, N.
2010. [make]SHIFT: Information Exchange and Collaborative Design Workflows. In *ACADIA 2010: life in:formation*, Pp. 139–144.
- Miller, N. and D. Stasiuk
2017. A Novel Mesh-Based Workflow for Complex Geometry in BIM. In *ACADIA 2017: Disciplines & Disruption*, Pp. 404–413.
- Miller, N., D. Stasiuk, and M. Glodsberry
2019a. Prove It - Episode 2 - Big Ship Computation with Matt Goldsberry.
- Miller, N., D. Stasiuk, and K. Schulte
2019b. Prove It - Episode 1: Generative What?
- Mirtschin, J.
2011. Engaging Generative BIM Workflows. *Collaborative Design of Lightweight Structures - LSAA 2011*, P. 8.

- Mollaert, M.
1989. A kernel reference model for Computer Aided Architectural Design. In *CAAD: Education - Research and Practice: eCAADe Conference Proceedings*, number conversion, Pp. 1–29, Aarhus, Denmark.
- Nejur, A. and K. Steinfeld
2016. Ivy : Bringing a Weighted-Mesh Representation to Bear on Generative Architectural Design Applications. *Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, Pp. 140–151.
- Nicholas, P.
2016. Concepts and Methodologies for Multi-scale Modelling: a Mesh-based Approach for Bi-directional Information Flows. *ACADIA2016*.
- Nicholas, P., D. Stasiuk, E. Clausen Nørgaard, C. Hutchinson, and M. Ramsgaard Thomsen
2015. A Multiscale Adaptive Mesh Refinement Approach to Architected Steel Specification in the Design of a Frameless Stressed Skin Structure. *Design Modelling Symposium*, Pp. 17–34.
- Nicholas, P. and M. Tamke
2013. Computational Strategies for the Architectural Design of Bending Active Structures. *International Journal of Space Structures*, 28(3-4):215–228.
- Nicholas, P., M. Zwierzycki, E. C. Nørgaard, S. Leinweber, D. Stasiuk, C. Hutchinson, and M. Thomsen
2017. Adaptive Robotic Fabrication for Conditions of Material Inconsistency: Increasing the geometric accuracy of incrementally formed metal panels. *Proceedings of the Fabricate 2017*, 2:114–121.
- Nicholas, P., M. Zwierzycki, D. Stasiuk, E. Nørgaard, S. Leinweber, and M. Ramsgaard Thomsen
2016. Adaptive Meshing for Bi-directional Information Flows. In *Advances in Architectural Geometry*, S. Adriaenssens, F. Gramazio, M. Kohler, A. Menges, and M. Pauly, eds., Pp. 260–273. Zürich: vdf Hochschulverlag AG.
- Nikiforakis, N.
2003. AMR for global atmospheric modelling. In *Adaptive Mesh Refinement - Theory and Applications - Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods*, T. Plewa, T. Linde, and V. G. Weirs, eds., Pp. 505–525, Chicago. Springer.
- Nobel, P., C. Sharples, K. Holden, and G. Pasquarelli
2012. *SHoP: Out of Practice*. New York, NY: The Monacelli Press.
- Nolte, T. and A. Witt
2014. Gehry partners' fondation louis vuitton: Crowdsourcing embedded intelligence. *Architectural Design*, 81(1):82–89.
- Paulson, B.
1976. Designing to Reduce Construction Costs. *International Journal of Sustainable Engineering*, 102(4):587–592.
- Pena De Leon, A.
2014. Separation of concerns: strategies for complex parametric design modelling. (March).
- Perkins, R., D. Smith Keller, and F. Ludolph
1997. Inventing the Lisa user interface. *Interactions*, 4(1):40–53.
- Peters, B. and T. Peters
2018. *Computing the Environment: Digital Design Tools for Simulation and Visualisation of Sustainable Architecture*. Wiley.
- Petric, J. and M. Lindsay
2009. Digital Prototyping. *CAADRIA 2003 [Proceedings of the 8th International Conference on Computer Aided Architectural Design Research in Asia]*, Pp. 837–852.

- Piermarini, E., H. Nuttall, R. May, and V. M. Janssens
2016. City of Dreams, Macau - Making the vision viable. *Structural Engineer*, 94(3):56–67.
- Poinet, P., P. Nicholas, M. Tamke, and M. R. Thomsen
2016. Multi-Scalar Modelling for Free-form Timber Structures. *Proceedings of the IASS Annual Symposium 2016 "Spatial Structures in the 21st Century"*.
- Poinet, P., M. Tamke, M. Ramsgaard Thomsen, F. Scheurer, and A. Fisher
2018. Schema-Based Workflows and Inter-Scalar Search Interfaces for Building Design. In *eWork and eBusiness in Architecture, Engineering and Construction : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2018)*.
- Rafsanjani, A., D. Derome, F. K. Wittel, and J. Carmeliet
2012. Computational up-scaling of anisotropic swelling and mechanical behavior of hierarchical cellular materials. *Composites Science and Technology*, 72(6):744–751.
- Ramsgaard Thomsen, M.
2016. Complex Modelling: Questioning the infrastructures of information modelling. *International Journal of Architectural Computing*, 1:33–42.
- Ramsgaard Thomsen, M. and M. Tamke
2009. Narratives of Making: thinking practice led research in architecture. *Communicating (by) Design*, (January 2009):1–8.
- Ramsgaard Thomsen, M. and M. Tamke
2015. Prototyping Practice : Merging Digital and Physical Enquiries. In *Rethink! Prototyping : Transdisciplinary Concepts of Prototyping*, Pp. 49–62. Cham: Springer.
- Ramsgaard Thomsen, M., M. Tamke, A. H. Deleuran, I. Katrine, F. Tinning, H. Leander, E. The, C. Gengnagel, and M. Schmeck
2015. Hybrid Tower - Designing Soft Structures. Pp. 1–10.
- Ramsgaard Thomsen, M., M. Tamke, P. Nicholas, A. Holden, P. Ayres, R. La Magna, and C. Gengnagel
2017. Simulation in Complex Modelling. *Symposium on Simulation for Architecture and Urban Design (SIMAUD)*, (May).
- Rasmussen, M. H., C. A. Hviid, and J. Karlshøj
2017a. Web-based topology queries on a BIM model.
- Rasmussen, M. H., P. Pauwels, C. A. Hviid, and J. Karlshøj
2017b. Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction*, (July):237–244.
- Reimer, J.
2005. A History of the GUI.
- Ringley, B.
2017. SFDUG Jun 2017: What is the point of using Dynamo?
- Samuelson, P. and R. M. Sherman
2006. Open Source and Competition in the Software Industry.
- Sattler, L.
2018. L'ère post-BIM – Pour une obsolescence déprogrammée : une étude de cas de deux projets de Frank Gehry en France, de 2008 à 2016.
- Scheurer, F.
2012. Digital Craftmanship: From Thinking to Modeling to Building. In *Digital Workflows in Architecture: Design-Assembly-Industry*, S. Marble, ed., Pp. 110–131. Basel: Birkhäuser.

- Scheurer, F., L. Simonin, and H. Stehling
2015. "Energy for life" – the timber structure of the French Pavilion at the EXPO 2015. *Proceedings of the International Association for Shell and Spatial Structures (IASS)*.
- Scheurer, F. and H. Stehling
2011. Lost in Parameter Space? *Architectural Design*, 81(4):70–79.
- Scheurer, F., H. Stehling, F. Tschümperlin, and M. Antemann
2013. Design for Assembly – Digital Prefabrication of Complex Timber Structures. *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2013*, Pp. 1–7.
- Schön, D. A.
1983. *The Reflective Practitioner - How Professionals Think in Action*. Basic Books.
- Schwerdtfeger, E.
2018. AULON98261: Custom Computational Workflows for BIM Design Implementation. Technical report, Autodesk Univeristy.
- Schwinn, T., O. D. Krieg, and A. Menges
2014. Behavioral Strategies: Synthesizing Design Computation and Robotic Fabrication of Lightweight Timber Plate Structures. *ACADIA 14: Design Agency [Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)]*, Pp. 177–188.
- Schwinn, T. and A. Menges
2015. Fabrication agency: Landesgartenschau exhibition hall. *Architectural Design*, 85(5):92–99.
- Secchi, S. and L. Simoni
2003. An improved procedure for 2D unstructured Delaunay mesh generation. *Advances in Engineering Software*, 34(4):217–234.
- Shelden, D. R.
2006. Tectonics, Economics and the Reconfiguration of Practice: The Case for Process Change by Digital Means. *Architectural Design: Programming Cultures*, 76(4):82–87.
- Smith, R.
2007. Technical Notes from experiences and studies in using Parametric and BIM architectural software . Virtual Build Technologies.
- Smith, R.
2017. *Fabricating the Frank Gehry Legacy: The Story of the Evolution of Digital Practice in Frank Gehry's office*. Rick Smith; 1 edition (February 1, 2017).
- Stasiuk, D.
2018. Design Modeling Terminology. Pp. 1–6.
- Steenson, M. W.
2017. *Architectural Intelligence: How Designers and Architects Created the Digital Landscape*. The MIT Press.
- Stefanescu, D.
2018. Reimagining the Design Process from the Internet Up. Technical report.
- Stefanescu, D., J. Edgan, S. Matthew, Y. van Havre, and W. Pearson
2018. speckleworks/SpeckleServer: 1.0.0-beta.
- Stehling, H., F. Scheurer, H. Geglo, and M. Hofmann
2017. From Lamination to Assembly: Modelling the Seine Musicale. In *Fabricate 2017: Rethinking Design and Construction*, A. Menges, B. Sheil, R. Glynn, and M. Skavara, eds., Pp. 258–263. London: UCL Press.
- Stephen, C.
2018. Q&A: Eckart Schwerdtfeger, ZHA | Custom scripts push BIM boundaries. *BIM+*.

- Sutherland, I. E., A. Blackwell, and K. Rodden
1963. Sketchpad: A man-machine graphical communication system.
- Svilans, T., P. Poinet, M. Tamke, and M. Ramsgaard Thomsen
2017. A Multi-scalar Approach for the Modelling and Fabrication of Free-Form Glue-Laminated Timber Structures. In *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, K. De Rycke, C. Gengnagel, O. Baverel, J. Burry, C. Mueller, M. M. Nguyen, P. Rahm, and M. Ramsgaard Thomsen, eds., Pp. 247–257. Springer.
- Tamke, M., P. Nicholas, and J. Riiber
2014. The Agency of Event: Event Based Simulation for Architectural Design. *Acadia*, (Figure 1):63–74.
- Tamke, M., D. Stasiuk, and M. Ramsgaard Thomsen
2013. The Rise - Material Behaviour in Generative Design. *ACADIA2013 - Adaptive Architecture*, Pp. 379–388.
- Tamke, M., M. Zwierzycki, A. H. Deleuran, Y. Sinke Baranovskaya, I. Tinning Friis, and M. Ramsgaard Thomsen
2017. Lace Wall: Extending Design Intuition through Machine Learning. UCL Press.
- Terzidis, K.
2003. *Expressive form: A conceptual approach to computational design*.
- Tessmann, O.
2008. *Collaborative Design Procedures for Architects and Engineers*. PhD thesis.
- Tombesi, P.
2012. What do we mean by building design? In *Digital Workflows in Architecture: Design-Assembly-Industry*, S. Marble, ed., chapter 11, Pp. 184–201. Basel: Birkhäuser.
- Twidale, M. B. and I. Floyd
2008. Infrastructures from the bottom-up and the top-down: Can they meet in the middle? *Proceedings of the Anniversary Conference on Participatory Design (PDC)*, Pp. 238–241.
- Usai, S. and H. Stehling
2017. La Seine Musicale: A Case Study on Design for Manufacture and Assembly. In *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, K. De Rycke, C. Gengnagel, O. Baverel, J. Burry, C. Mueller, M. M. Nguyen, P. Rahm, and M. Ramsgaard Thomsen, eds. Springer Singapore.
- Van Berlo, L., J. Beetz, P. Bos, H. Hendriks, and R. Van Tongeren
2012. Collaborative engineering with IFC : new insights and technology. *eWork and eBusiness in Architecture, Engineering and Construction, 9th ECPPM Conference Proceedings*, (December):811–818.
- Van Der Heijden, R.
2017. Parametric Building Information Generation for Design and Construction - Talk @ "BIM 2017 | WHAT'S NEXT?".
- Van Der Heijden, R., E. Levelle, and M. Reise
2015. Parametric Building Information Generation for Design and Construction. *Acadia*, Pp. 417–430.
- Vasey, L., E. Baharlou, M. Dörstelmann, V. Koslowski, M. Prado, G. Schieber, A. Menges, and J. Knippers
2015. Behavioral Design and Adaptive Robotic Fabrication of a Fiber Composite Compression Shell With Pneumatic Formwork. *Proceedings of ACADIA 2015*, Pp. 297–309.
- Wang, W., B. Lu, N. Zhang, Z. Shi, and J. Li
2010. International Journal of Multiphase Flow A review of multiscale CFD for gas – solid CFB modeling. 36:109–118.

Weyuker, E. J.

1998. Testing component-based software: A cautionary tale. *IEEE Software*, 15(5):54–59.

Winsberg, E.

2010. *Science in the Age of Computer Simulation*. The University of Chicago Press.

Wong, K.

2006. What Grounded the Airbus A380?

Wood, D., D. Correa, O. Krieg, and A. Menges

2016. Material computation–4D timber construction: Towards building-scale hygroscopic actuated, self-constructing timber surfaces. *International Journal of Architectural Computing*, 14(1):49–62.

Xia, L. and P. Breitkopf

2015. Multiscale structural topology optimization with an approximate constitutive model for local material microstructure. *Computer Methods in Applied Mechanics and Engineering*, 286:147–167.

Yershóv, A. P.

1965. One View of Man-Machine Interaction. *Journal of the ACM*, 12(3):315–325.

Zhao, J., M. Glueck, F. Chevalier, Y. Wu, and A. Khan

2016. Egocentric Analysis of Dynamic Networks with EgoLines. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, Pp. 5003–5014.

Zhao, J., M. Glueck, P. Isenberg, F. Chevalier, and A. Khan

2018. Supporting Handoff in Asynchronous Collaborative Sensemaking Using Knowledge-Transfer Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):340–350.

Illustration Credits

2.1	© Microsoft Corporation	23
2.2	© Apple	24
2.3	Diagram by Robert Desourdis et al. (2009)	26
2.4	© Disney/Pixar (https://graphics.pixar.com/usd/overview.html , accessed March 2019)	28
2.5	Diagram by Wassim Jabi et al. (2018)	32
2.6	Screenshots (Rhino3D, Digital Project, IFC Tree Structure) and diagrams by the author	34
2.7	Thornton Tomasetti's Design Explorer	35
2.8	Top: diagram by Martin Krzywinski et al. (2009), bottom: diagram by Miriah Meyer et al. (2009)	37
2.9	Diagram by Daniel Davis (2013) revisited by the author	42
2.10	Diagram by Ramon Van der Heijden et al. (2015)	45
2.11	Diagram by Oliver Tessmann (2008)	46
2.12	Diagram by Bauke de Vries	47
2.13	Diagrams by the author	49
2.14	Diagram by the author	50
2.15	Top left: diagram by Paolo Tombesi (2012), top right: diagram by Alex Kunz et al. (2014), bottom: diagram by Dimitrie Stefanescu (2016)	53
2.16	Diagram by Marble Scott and James Kotronis (2012)	54
2.17	Diagram by Paul Baran (1964)	56
2.18	Meme by unknown author	57
2.19	Figure by McKinsey Global Institute. Source: AppBrain; Bluewolf; Computer Economics; eMarketer; Gartner; IDC Research; LiveChat; US Bureau of Economic Analysis; US Bureau of Labor Statistics; US Census Bureau; McKinsey Global Institute analysis	62
2.20	Diagram by Paul Wintour (2016)	63
2.21	Top: Diagram by SHoP Architects (2012), bottom: 3DEXPERIENCE® - Dassault Systèmes®	65
2.22	Diagram by Nathan Miller and David Stasiuk (2017)	66
2.23	Diagrams by Eckart Schwerdtfeger (2018)	68
2.24	Top: diagram by Design-to-Production (2012), bottom: Excel management sheet by Design-to-Production (2016)	69
2.25	Diagram by KieranTimberlake (2016)	71
2.26	Database and 3D models screenshots by BuroHappold (2017)	73
2.27	Top and Bottom: diagrams by BuroHappold (2018)	74
2.28	Top and bottom: diagrams by Treegram (2017)	75
2.29	Diagram by Woods Bagot (2017)	76
2.30	Screenshot from Thornton Tomasetti's CORE studio's main applications (accessed March 2019)	77
2.31	Diagram by Dimitrie Stefanescu	80
2.32	Screenshot of the Speckle project management interface (accessed March 2019)	81
2.33	Diagram provided by BuroHappold Engineering (2017)	83
2.34	Diagram provided by BuroHappold Engineering (2017)	84

2.35	Top: diagram by Abby Covert (2016) revisited by the author, bottom: diagram by the author	85
2.36	Diagrams by the author	86
2.37	Top and bottom: diagrams by Mario Hirz et al. (2013)	88
2.38	Screenshot of the EgoLines application by Jian Zhao et al. (2016)	89
2.39	Screenshot of the KTGraph application by Jian Zhao et al. (2018)	90
3.1	Screenshots of Power of Ten, directed by Charles and Ray Eames (1977)	94
3.2	Diagram by the author	97
3.3	Top: diagram by Frederick Kiesler (1937–1941), middle: diagram by Andreï Petrovitch Yershóv (1963), bottom: photograph by unknown of Timothy Johnson	98
3.4	Diagram by Paul Nicholas et al. (2016)	101
3.5	Diagrams by Martin Tamke et al. (2017)	103
3.6	Top: diagram by Katherine Esau (1960), middle: diagram by Tobias Schwinn et al. (2014), bottom: diagram by Jian-Min Li and Jan Knippers (2015)	105
3.7	Diagram by Nikos Nikiforakis (2003)	107
3.8	Top: diagram by Stefano Secchi and Luciano Simoni (2003), bottom: 3D model and screenshot by Mathieu Venot (2018)	108
3.9	Top: diagram by Trevor Steil, bottom: diagram by Paul Nicholas et al. (2016)	111
3.10	Diagram by Martin Tamke et al. (2014)	112
3.11	Diagram by Martin Tamke et al. (2014)	113
3.12	Screenshot of the building's 3D model and photograph by Acton Ostry Architects Inc. (2016)	114
3.13	From left-to-right and top-to-bottom: photograph by CODA - Computational Design Affairs (2014), photograph by ICD/ITKE (2014), photograph by CITA (2016), photograph by ICD/ITKE (2015), photograph by CITA (2016), photograph by ICD/ITKE (2016), 3D model by David Stasiuk (2013), 3D model by Front Inc. (2017), 3D model by Design-to-Production (2015), 3D model by Design-to-Production (2017), 3D model by Gehry Technologies (2017), 3D model by ShoP Architects (2012)	116
3.14	Diagram by the author	118
3.15	Diagram by Tobias Nolte and Andrew Witt (2014)	119
3.16	Diagrams by Emidio Piermarini et al. (2016)	121
3.17	Diagram by the author	122
3.18	Top: diagram by Christopher Alexander (1973), bottom: diagram by the author	123
3.19	Diagram by the author	127
3.20	Diagram by the author	132
4.1	Rhino3D model by the author	137
4.2	Diagram by the author	138
4.3	Screenshot by the author	140
4.4	Diagram by the author	141
4.5	3D models and photographs by Klaus Linkwitz and Diederik Veenendaal (2014)	143
4.6	Diagram by the author	146
4.7	Rhino3D-CAD model (top) and two-dimensional map (bottom) by Design-to-Production (2015)	148
4.8	Diagram by the author	149
4.9	Drawing by the author	150
4.10	Rhino3D model by the author	151
4.11	Graphs by the author	152
4.12	Rhino3D model by the author	154
4.13	Rhino3D model and corresponding 2D graph by the author	154
4.14	Rhino3D model by the author	155
4.15	Rhino3D models by the author	156
4.16	Graph by the author	157
4.17	Rhino3D model by the author	158

4.18	Rhino3D model by the author	159
4.19	Rhino3D model by the author	160
4.20	Rhino3D model by the author	160
4.21	Rhino3D model by the author	161
4.22	From left-to-right and top-to-bottom: fabrication setup, Rhino3D model and photograph by the author, Tom Svilans and Kasper Ax	163
4.23	Rhino3D model by the author, Tom Svilans and Kasper Ax	164
4.24	Photograph by the author, Tom Svilans and Kasper Ax	165
4.25	Dynamo definition and 3D model by the author	166
4.26	3D models and physical prototype by the workshop participants and the author	167
4.27	Rhino3D models and diagrams by Emil Buchwald, Stian Vestly Holte, Daoyuan Zhu and Brian Cheung	168
4.28	Rhino3D model by the author	170
4.29	Rhino3D model by the author	170
4.30	Rhino3D model by the author	171
4.31	Rhino3D model by the author	172
4.32	Rhino3D model by the author and Tom Svilans, rendered by Leonardo Castaman and Enrico Pontello	174
4.33	Diagram by the author and Tom Svilans	175
4.34	Rhino3D models by the author	176
4.35	Rhino3D models by the author and Tom Svilans	177
4.36	Rhino3D model by the author and Tom Svilans	177
4.37	Diagram by the author	178
4.38	Diagram by the author	179
4.39	Diagram by Tom Svilans et al. (2017)	180
4.40	Rhino3D fabrication model by the author (2017)	181
4.41	Fabrication models, photographs and physical prototyping by the author and Tom Svilans	182
4.42	Fabrication models, photographs and physical prototyping by the author and Tom Svilans	182
4.43	3D printed model by the author and Tom Svilans (2017)	183
5.1	Diagram by the author	186
5.2	Diagrams by the author, Rhino3D model provided by Design-to-Production	187
5.3	Diagrams by the author, Rhino3D models provided by Design-to-Production	188
5.4	Rhino3D model by the author	189
5.5	Diagrams by the author	191
5.6	Diagrams by Christopher Alexander (1965)	193
5.7	Diagram by the author	195
5.8	Diagram by the author	196
5.9	UI by the author, 3D model provided by Design-to-Production	199
5.10	Search Interface by the author, 3D model provided by Design-to-Production	200
5.11	Diagram by d3indepth	201
5.12	Placeholder	201
5.13	Screenshot of the Rhino3D viewport and the Speckle UI by Dimitrie Stefanescu (2018), Rhino3D model provided by Design-to-Production	203
5.14	Flowchart by the author	205
5.15	Diagram by the author	206
5.16	SchemaBuilder's GUI by the author	207
5.17	Screenshot by the author	207
5.18	Diagram by the author	209
5.19	Rhino3D-Grasshopper screenshots by the author, 3D model provided by Design-to-Production	211
5.20	Screenshot by the author	213
5.21	Rhino3D-Grasshopper screenshots by the author, 3D model provided by Design-to-Production	214

5.22	Diagram by the author	216
5.23	Diagram by the author	217
5.24	Diagrams and screenshots by the author	219
5.25	Diagrams and screenshots by the author	220
5.26	3D model by the author, diagram by Djordje Stanojevic	221
5.27	3D model by the author	222
5.28	3D model by the author	223
5.29	3D model by the author	224
5.30	GUI by the author	225
5.31	Diagram by the author	227
6.1	Diagram by the author	233
6.2	Diagram by the author	239
6.3	GUI by the author	241

Glossary

Application Programming Interface (API) An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. The Oxford English Dictionary defines an API as follows: *“Computing a set of routines, protocols, and tools designed to allow the development of applications that can utilize or operate in conjunction with a given item of software, set of data, website, etc.”* 28, 30–32, 63–66, 198, 210, 239

Architecture, Engineering and Construction (AEC) Architecture, Engineering and Construction (AEC) is the sector of the construction industry that provides the services on the architectural design, engineering design and construction services. It is a sector which is very active in the adoption of Information, Communication and Technology. 7, 13–15, 17–19, 21, 25, 26, 29–31, 33, 35, 36, 38–43, 51, 52, 54, 55, 58–61, 63, 64, 66, 75, 77–80, 89, 91, 115, 117, 123, 125, 130, 204, 212, 222, 227, 229–231, 234–240

BHoM BHoM (Buildings and Habitats object Model) is an open-source platform developed by BuroHappold Engineering. The platform offers a neutral object model enabling structural engineers and architects to exchange seamlessly structural elements and data between multiple software platforms. 9, 18, 72, 74, 82, 129, 134, 213–215, 218, 232, 234, 235, 237

Building Information Modelling (BIM) BIM (Building Information Modeling) is an intelligent 3D model-based process that gives architecture, engineering, and construction (AEC) professionals the insight and tools to more efficiently plan, design, construct, and manage buildings and infrastructure. 15, 29–31, 38, 40–42, 54–60, 63, 66, 67, 70, 71, 77, 80, 83, 84, 100, 114, 229

CITA CITA (Centre for Information Technology in Architecture) is an innovative research environment exploring the intersections between architecture and digital technologies. Identifying core research questions into how space and technology can be probed, CITA investigates how the current forming of a digital culture impacts on architectural thinking and practice. CITA is based at The Royal Danish Academy of Fine Arts / School of Architecture, Design and Conservation (KADK) and is directed by Professor Mette Ramsgaard Thomsen and Associate Professor Martin Tamke. 11, 14–16, 51, 100–102, 104, 109, 110, 112, 132, 256

Computer-Aided Design (CAD) Computer-Aided Design (CAD) software is used by architects, engineers, drafters, artists, and others to create precision drawings or technical illustrations. CAD software can be used to create two-dimensional (2-D) drawings or three-dimensional (3-D) models. 7, 21–38, 40, 57–59, 63, 65, 81, 144, 148, 193, 194, 229, 256

D3.js D3.js (also known as D3, short for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented Scalable Vector Graphics, HTML5, and Cascading Style Sheets standards. 198, 201

Directed Acyclic Graph (DAG) In mathematics and graph theory, a Directed Acyclic Graph (DAG) is a directed graph with no directed cycles. It consists of vertices (or nodes) and edges that connect those nodes. In computational modelling, DAGs are used to create geometrical relationships

between different objects, or components. 7, 17, 18, 38–42, 53, 59, 110, 116, 122–124, 160, 186, 188, 232, 235

Graphical User Interface (GUI) The Oxford English Dictionary defines a Graphical User Interface as follows: *“A visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems.”* A Graphical User Interface is considered to be a subset of a [UI](#). 7, 18, 21–37, 89, 91, 188, 192, 194, 197–200, 202, 207, 231, 236–238, 240, 257, 258

Industry Foundation Classes (IFC) The Industry Foundation Classes (IFC) is a data model intended to describe architectural, building and construction industry data. 14, 229

JSON (JavaScript Object Notation) JSON (JavaScript Object Notation) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). 200

Non-Uniform Rational Basis Spline (NURBS) A Non-Uniform Rational Basis Spline (NURBS) is a mathematical model commonly used in computer graphics for generating and representing curves and surfaces. 21, 30, 58, 67, 158, 229

Speckle Speckle (<https://speckle.works/>) was originally developed by Dimitrie Stefanescu (ESR 5) at The Bartlett School of Architecture as part of the InnoChain project, a H2020 Marie Curie ETN network. Since its inception in 2016, it has been adopted by a large number of progressive AEC companies as a key piece in their digital transformation efforts. 9, 11, 18, 50, 67, 68, 79–81, 129, 134, 203, 204, 206–215, 218–220, 222–226, 232–238, 240, 241, 255, 257

User Experience (UX) The Oxford English Dictionary defines a Graphical User Interface as follows: *“The overall experience of a person using a product such as a website or computer application, especially in terms of how easy or pleasing it is to use.”* 7, 18, 21, 25, 33, 35, 36, 38, 91, 130, 152, 160, 188, 197, 206, 231, 238

User Interface (UI) The Oxford English Dictionary defines a User Interface as follows: *“The means by which the user and a computer system interact, in particular the use of input devices and software.”* 7, 18, 21, 22, 32, 33, 35, 38, 48, 67, 76, 82, 83, 91, 140, 141, 147, 192, 193, 206, 215, 237, 238, 240, 257